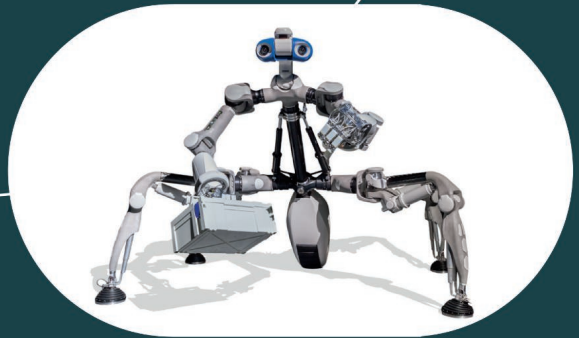
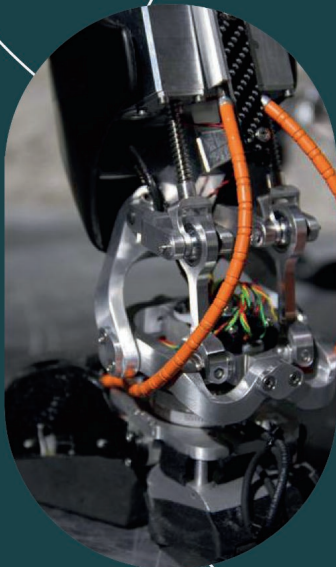


Biologically Inspired Series-Parallel Hybrid Robots

Design, Analysis, and Control

Shivesh Kumar
Andreas Müller
Frank Kirchner



**BIOLOGICALLY
INSPIRED
SERIES-PARALLEL
HYBRID ROBOTS**

This page intentionally left blank

BIOLOGICALLY INSPIRED SERIES-PARALLEL HYBRID ROBOTS

Design, Analysis, and Control

SHIVESH KUMAR

Robotics Innovation Center
German Research Center for Artificial Intelligence
Bremen, Germany

ANDREAS MÜLLER

Institute of Robotics
Johannes Kepler University
Linz, Austria

FRANK KIRCHNER

Robotics Innovation Center
German Research Center for Artificial Intelligence
Bremen, Germany



ACADEMIC PRESS

An imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States

Copyright © 2025 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

Publisher's note: Elsevier takes a neutral position with respect to territorial disputes or jurisdictional claims in its published content, including in maps and institutional affiliations.

MATLAB® is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® software.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-323-88482-2

For information on all Academic Press publications
visit our website at <https://www.elsevier.com/books-and-journals>

Publisher: Mara Conner
Acquisitions Editor: Sonnini Yura
Editorial Project Manager: Tessa Kathryn
Production Project Manager: Surya Narayanan
Jayachandran
Cover Designer: Mark Rogers

Typeset by VTeX



Contents

<i>List of contributors</i>	<i>ix</i>
<i>Preface</i>	<i>xv</i>

PART 1 Introduction

1. Motivation	3
Shivesh Kumar, Andreas Müller, and Frank Kirchner	
2. Modular and decentralized design principles and applications	17
Shivesh Kumar, Hendrik Wöhrle, José de Gea Fernández, Andreas Müller, and Frank Kirchner	

PART 2 Geometric analysis

3. Methods for geometric analysis of parallel mechanisms	51
Shivesh Kumar and Andreas Müller	
4. 2-DOF orientational parallel mechanisms	69
Shivesh Kumar, Christoph Stoeffler, Heiner Peters, Andreas Müller, and Frank Kirchner	
5. 3-DOF orientational parallel mechanism	93
Shivesh Kumar, Bertold Bongardt, and Frank Kirchner	

PART 3 Kinematics, dynamics, and control

6. Kinematics and dynamics of tree type systems	121
Shivesh Kumar and Andreas Müller	
7. Modular algorithms for kinematics and dynamics of series-parallel hybrid robots	145
Shivesh Kumar, Rohit Kumar, Andreas Müller, and Frank Kirchner	
8. Forward dynamics with constraint embedding for dynamic simulation	179
Rohit Kumar, Shivesh Kumar, Andreas Müller, and Frank Kirchner	

9. Whole-body control	195
Dennis Mronga, Shivesh Kumar, and Frank Kirchner	

10. Whole-body trajectory optimization	213
Melya Boukheddimi, Rohit Kumar, Shivesh Kumar, Justin Carpentier, and Frank Kirchner	

PART 4 Case studies on mechatronic system design

11. Charlie, a hominidae walking robot	233
Daniel Kühn, Alexander Dettmann, Moritz Schilling, Tobias Stark, and Sankaranarayanan Natarajan	

12. Multi-legged robot Mantis	261
Wiebke Brinkmann, Alexander Dettmann, Leon C. Danter, Tobias Stark, Christopher Schulz, Holger Sprengel, Marc Manz, and Sebastian Bartsch	

13. Sherpa, a family of wheeled-leg rovers	281
Florian Cordes, Ajish Babu, and Tobias Stark	

14. Recupera exoskeletons	305
Martin Mallwitz, Michael Maurus, Shivesh Kumar, Mathias Trampler, Marc Tabie, Hendrik Wöhrle, Elsa A. Kirchner, Henning Wiedemann, Heiner Peters, Christopher Schulz, Kartik Chari, Ibrahim Tijjani, and Frank Kirchner	

15. RH5 Pedes humanoid	333
Melya Boukheddimi, Ivan Bergonzani, Shivesh Kumar, Heiner Peters, and Frank Kirchner	

16. ARTER: a walking excavator robot	355
Ajish Babu, Pierre Willenbrock, Jannik Tiemann, Felix Bernhard, and Daniel Kühn	

PART 5 Software and outlook

17. PHOBOS: creation and maintenance of complex robot models	381
Henning Wiedemann, Kai A. von Szadkowski, and Malte Langosz	

18. HyRoDyn: Hybrid Robot Dynamics	397
Shivesh Kumar, Rohit Kumar, Mareike Förster, Andreas Müller, and Frank Kirchner	
19. Design of a flexible bio-inspired robot for inspection of pipelines	427
Swaminath Venkateswaran and Damien Chablat	
20. Optimization of parallel mechanisms with joint limits and collision constraints	451
Durgesh H. Salunkhe, Shivesh Kumar, and Damien Chablat	
<i>Index</i>	<i>477</i>

This page intentionally left blank

List of contributors

Ajish Babu

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Sebastian Bartsch

Airbus Defense and Space (ADS), Bremen, Germany

Ivan Bergonzani

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Felix Bernhard

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Bertold Bongardt

Institute for Robotics and Process Informatics, Technical University of Braunschweig, Braunschweig, Germany

Melya Boukheddimi

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Wiebke Brinkmann

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Justin Carpentier

INRIA Paris, France

Damien Chablat

Centre National de la Recherche Scientifique (CNRS), Laboratoire des Sciences du Numérique de Nantes (LS2N), UMR CNRS 6004, Nantes, France

Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, Nantes, France

Kartik Chari

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Florian Cordes

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Leon C. Danter

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

José de Gea Fernández

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Alexander Dettmann

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Mareike Förster

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Elsa A. Kirchner

Institute of Medical Technology Systems, University of Duisburg-Essen, Duisburg, Germany

Frank Kirchner

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Working Group Robotics, University of Bremen, Bremen, Germany

Daniel Kühn

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Rohit Kumar

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Shivesh Kumar

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Malte Langosz

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Martin Mallwitz

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Marc Manz

Airbus Defense and Space (ADS), Bremen, Germany

Michael Maurus

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Dennis Mronga

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Andreas Müller

Institute of Robotics, Johannes Kepler University, Linz, Austria

Sankaranarayanan Natarajan

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Heiner Peters

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Durgesh H. Salunkhe

Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, Nantes, France

Moritz Schilling

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Christopher Schulz

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Holger Sprengel

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Tobias Stark

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Christoph Stoeffler

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Marc Tabie

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Jannik Tiemann

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Ibrahim Tijjani

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Mathias Trampler

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Swaminath Venkateswaran

Léonard de Vinci Pôle Universitaire, Research Center, 92 916 Paris La Défense, Paris, France

Kai A. von Szadkowski

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Henning Wiedemann

Working Group Robotics, University of Bremen, Bremen, Germany

Pierre Willenbrock

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Hendrik Wöhrle

Intelligente autonome Sensor- und Aktor-Systeme, Fachhochschule Dortmund,
Dortmund, Germany

Robotics Innovation Center, German Research Center for Artificial Intelligence
(DFKI GmbH), Bremen, Germany

Institute of Communication Technology, Dortmund University of Applied
Sciences and Arts, Dortmund, Germany

This page intentionally left blank

Preface

Nature has always served as a profound source of inspiration in the field of robotics. Most animals have a parallel arrangement of muscles around their skeletal system, which brings a certain mechanical advantage to them. Taking inspiration from this observation, parallel mechanisms are increasingly being used as modular subsystem units in various robots for their superior stiffness, payload-to-weight ratio and dynamic properties. This leads to series-parallel hybrid robotic systems that combine advantages of both serial and parallel design architectures but also inherit their kinematic complexity.

Although several excellent books have been published on serial and parallel robots, there is currently no comprehensive resource that systematically addresses the unique challenges in the design and control of series-parallel hybrid robots. This book is structured to provide a comprehensive understanding of the design, analysis, and control of biologically inspired series-parallel hybrid robots. While the knowledge presented here can be found scattered across numerous research papers, this book endeavors to collect, organize, and present this information systematically, offering a single, consolidated resource for students, researchers, and practitioners in the field of robotics.

The book is organized into five parts. Part 1 provides a general introduction to series-parallel hybrid robots and presents the state of the art in this area, highlighting the modular and distributed aspects of their mechatronic system design. Part 2 provides an overview of modern methods, such as screw theory and computational algebraic geometry for geometric analysis of parallel mechanisms and their applications for solving geometric models of exemplary parallel joint modules with varying degrees of freedom. Part 3 explains the kinematics, dynamics and control aspects of series-parallel hybrid robots. The methods presented here allow for high-fidelity simulation, motion planning as well as whole-body control directly in the actuation space of the robots allowing the full utilization of their capabilities. Part 4 presents case studies on various series-parallel hybrid robot designs, for example, humanoids, exoskeletons, multi-legged robots, etc. Part 5 deals with some advanced topics such as their software aspects, design optimization, integration of spring elements into such a robot design.

The target audience for this book is intended to be quite broad, since it provides a balanced treatment to the needs of robotics researchers (Part 1, 2, and 3), control engineers (Part 3), mechanical or mechatronics engineers

(Part 4), etc. The case studies presented in Part 4 could be also interesting for system engineers or technologists. The book is also useful for teachers and graduate students in various disciplines such as robotics, mechanical engineering, electrical engineering, computer science, etc.

This book builds upon the PhD thesis of the first author, Shivesh Kumar, under the supervision of the co-authors, Andreas Müller and Frank Kirchner. The authors have together studied the design, modeling and control aspects of series-parallel hybrid robots in detail and have implemented many of these methods on various robots in practice at the DFKI Robotics Innovation Center in Bremen, Germany. When the first author started working in this area, these robots were dealt with ad-hoc approaches which made their mechanical as well as control design cumbersome and challenging. In most cases, only a kinematic control of these systems was possible. The holistic approach presented in this book has enabled their kinematic as well as dynamic control directly in the actuation space enabling full exploitation of their dynamic capabilities. We hope that these methods will help other robotics researchers in bringing the full potential out of their robotic system design.

Finally, we extend our gratitude to the numerous researchers and practitioners whose work has contributed to the knowledge encapsulated in this book. Their dedication and innovative spirit continue to push the boundaries of what is possible in robotics.

Shivesh Kumar, Andreas Müller, and Frank Kirchner
July 2024

PART 1

Introduction

This page intentionally left blank

CHAPTER 1

Motivation

Shivesh Kumar^a, Andreas Müller^b, and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

^cWorking Group Robotics, University of Bremen, Bremen, Germany

1.1 Introduction

With rapid progress in robot technology, our expectations from robotic systems are steadily increasing. Present day robots are expected to be fast, agile, safe, soft, precise, compliant, predictable, and intelligent at the same time which can also sometimes sound contradictory. To address these demands, the robotic systems need to have an intelligent design as the intelligence in behavior and control can not alone meet the growing expectations.

Nature has always captivated the attention of roboticists and inspired the design of robotic systems. Most industrial robotic arms include a shoulder-elbow-wrist architecture similar to humans abstracted with either seven revolute joints grouped in spherical (S) - revolute (R) - spherical (S) joint pairs or six revolute joints grouped in universal (U) - revolute (R) - spherical (S) joint pairs. An example of the former is KUKA iiwa LBR robot and the latter is Staubli RX90 robot. An advantage of this architecture is the principle solvability of its inverse kinematics problem. This kind of architecture is inspired from biology but mostly on the surface. Looking under the skin, one finds various muscle groups actuating a certain joint in rather a series-parallel architecture. For example, the elbow joint in human arm includes biceps and triceps muscles connected with the rigid skeleton with the help of tendons. This allows for intelligent placement of the actuators along the links and improve the payload to weight ratio of the arm. This is becoming the design inspiration in modern robotics and designers around the world are trying to abstract different kinematic joints with the help of parallel mechanisms.

1.2 Series-parallel hybrid robots

Definition 1.1. A serial robot is a mechanical system that is designed as a series of links connected by motor-actuated joints that extend from a base to a single end-effector.

Definition 1.2. A tree-type robot is a mechanical system that is designed as a series of links connected by motor-actuated joints that extend from a base to multiple end-effectors.

Serial and tree-type robots are well known for their versatility in applications, large workspace, simple modeling, and control. Hence, they often represent the state-of-the-art in robotic systems. However, they generally feature only limited precision, low structural stiffness, and poor dynamic characteristics. For these reasons, robots based on a pure tree type topology suffer from speed and torque limitations. Fig. 1.1a and Fig. 1.1b show schematics of exemplary serial and tree-type systems, respectively.

Definition 1.3. A parallel robot is a mechanical system that uses several serial chains to support a single platform or end-effector.

In contrast to serial robots, parallel robots as shown in Fig. 1.1c can provide higher stiffness, speed, accuracy, and payload capacity. On the downside, they possess a reduced workspace and a more complex geometry that requires careful analysis and control. Parallel robots have been traditionally used in more tailored use cases such as fast pick-and-place applications [1], driving simulators [2], fast orientation devices [3], etc. Table 1.1 presents an overview of comparison between serial and parallel robots. It can be noticed that advantages and disadvantages of two designs are almost complementary.

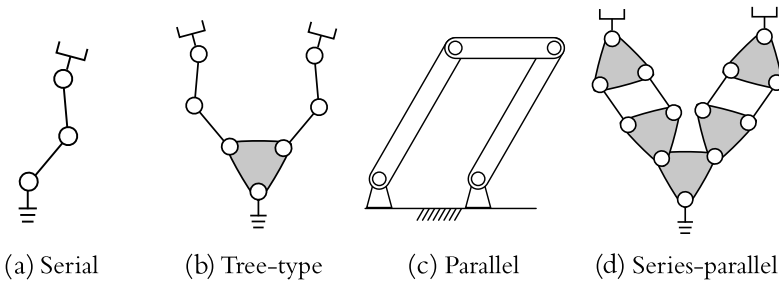


Figure 1.1 Serial, tree-type, parallel, and series-parallel hybrid mechanisms [4] (CC BY 4.0).

Table 1.1 Comparison of serial / tree-type robots and parallel robots.

Serial / Tree-type Robots	Parallel Robots
+ State of the art, easier to control	- Complex geometry, difficult to control
+ Large workspace	- Limited workspace, singularities
- Inherent low stiffness	± High stiffness (inhomogeneous)
- Lower precision	+ Higher precision
- Lower payload capacity	+ Higher payload capacity
- Acceleration limits	+ High accelerations (up to 100 g)

Definition 1.4. A series-parallel hybrid robot is defined as a robot that is built from a serial or tree-type combination of serial and parallel mechanisms.

Series-parallel hybrid designs (see Fig. 1.1d), combining the advantages of serial and parallel topologies, have long been common in the field of heavy machinery, e.g., cranes, excavator arms, bulldozers, etc., for a long time. Such designs have also recently caught the attention of robotics researchers from industry and academia. For instance, the stiffness of an industrial manipulator can be significantly improved by including a simple parallelogram mechanism. In particular, industrial robots as ABB’s IRB4400, IRB6660, KUKA’s KR 40-PA., KR 50-PA., KR 700-PA robots, and Comau’s Smart NJ series, SR400 utilize this design concept [5], [6]. Logabex’s LX4 robot piles four Stewart platform modules in series to achieve a redundant series-parallel hybrid manipulator with large workspace and good payload (75 kg) to weight (120 kg) ratio [7]. In academia and R&D, the idea to use closed-loop mechanisms and parallel kinematic manipulators (PKM) has been utilized more liberally, giving rise to a number of biologically inspired lightweight robotic systems with good dynamic characteristics. Some prominent examples, including Stewart platform, 2SPU+1U¹ [9], double parallelogram linkage [10], etc., have been used in various hybrid robots like hominid CHARLIE [11], multi-legged robot MANTIS [12], AXO-SUIT [13], and humanoid robot LOLA [14], THOR [15], etc. The design motivation of such hybrid robots is evident: use of PKM-based submechanisms helps designers to achieve lightweight, modular, and compact design while enhancing the stiffness and dynamic characteristics of the robot.

¹ In mechanism theory, it is typical to identify mechanisms using their type. For details, see [8].

1.3 Complexity in their modeling and control

On one hand series-parallel hybrid designs combine the advantages of serial and parallel architectures, on the other, they inherit the kinematic complexity of both designs. Observing this development in robotics, it can be noticed that robots are becoming increasingly complex for modeling and control. It is well known that the inverse kinematics problems for serial chains are difficult to solve and forward kinematics problems are difficult to solve for the parallel robots [16]. Hence, both forward and inverse kinematics problems are difficult to solve for series-parallel designs. While numerical tools for solving such problems exist, they often provide only a *local* view to their kinematic behavior. Tailored analytical solutions to these problems can provide deeper and *global* insights into their kinematics. Hence, it is interesting for geometers and kinematics researchers, to study the analytical solutions to geometric problems associated with a specific type of parallel mechanisms using methods from screw theory, computational algebraic geometry, distance geometry, etc. Their importance over numerical solutions is irrefutable.

Due to difficulties in modeling of series-parallel hybrid robots, their full dynamic model is not exploited. For example, in most cases these robots are often limited to position control (e.g. MANTIS, CHARLIE, SHERPATT [17]) where a joint to actuator mapping is utilized to achieve a kinematic control of the robot. When equipped with real time dynamic control, an inverse dynamic model in generalized coordinates (neglecting the contribution from the closed loops) is often combined with an inverse static model in actuation space to compute the actuator forces [18], [19], [20]. This approach is used in torque controlled series-parallel hybrid humanoids such as THOR, Valkyrie, Lola, etc. An obvious advantage of this simplification is the reduced CPU time needed to solve the inverse dynamic model but it leads to unnecessary increase of PD gains for achieving the desired controller tracking performances. The trade offs of this model simplification have been studied in [21]. Similarly, solving forward dynamics of such robots is challenging which makes the simulation of these robots very difficult. Numerical handling of loop closure constraints while solving forward dynamics often lead to numerical errors.

Since, the kinematic and dynamic modeling of series-parallel hybrid robots is challenging, whole body trajectory optimization [22] and whole body control [23] becomes even more challenging. As a workaround, parallel mechanisms are abstracted as serial chains for the purpose of whole body trajectory optimization and whole body control and the solution is

mapped to the full system using specialized functions which leads to various limitations. Some of these limitations include:

- Box constraints used to model the physical limitations of the abstracted serial mechanism either overestimate or underestimate the effective workspace of the real robot.
- Parallel mechanisms may be subject to singularities or poorly conditioned areas in the workspace that are not taken into account in the optimization problem while working with serial models.
- The optimization formulation is inaccurate since it does not take into account the full dynamics of the closed-loop mechanisms of the system.
- It leads to custom and complicated robot control software stacks, which may be hard to maintain and reuse.

1.4 Structure

The general objective of this book is to provide a systematic treatment of modeling and control of series-parallel hybrid robotic systems to help designers and control engineers to develop and control the complex robotic systems of the future.

This book is organized in a total of 20 chapters which are categorized into 5 parts namely, [1. Introduction](#), [2. Geometric Analysis](#), [3. Kinematics, Dynamics, and Control](#), [4. Case Studies on Mechatronic System Design](#), and [5. Software and Outlook](#). Fig. 1.2 shows the overall outline of this book. In the following, a summary of the each chapter is provided which lets the readers easily navigate through the book document.

Part 1: Introduction and SOTA

This part collects the Chapters [1](#) & [2](#) of this book which provide an introduction and study of state of the art in this area.

Chapter 1 (Motivation)

This chapter serves as an entry point to the book and presents the motivation, some key definitions and challenges this book aims to address. It also provides details about the structure of the book.

Chapter 2 (Modular and decentralized design principles and applications)

This chapter provides a state of the art survey on various series-parallel hybrid robots in various application domains. Further, it studies the modular and decentralized aspect of their design in both hardware and software

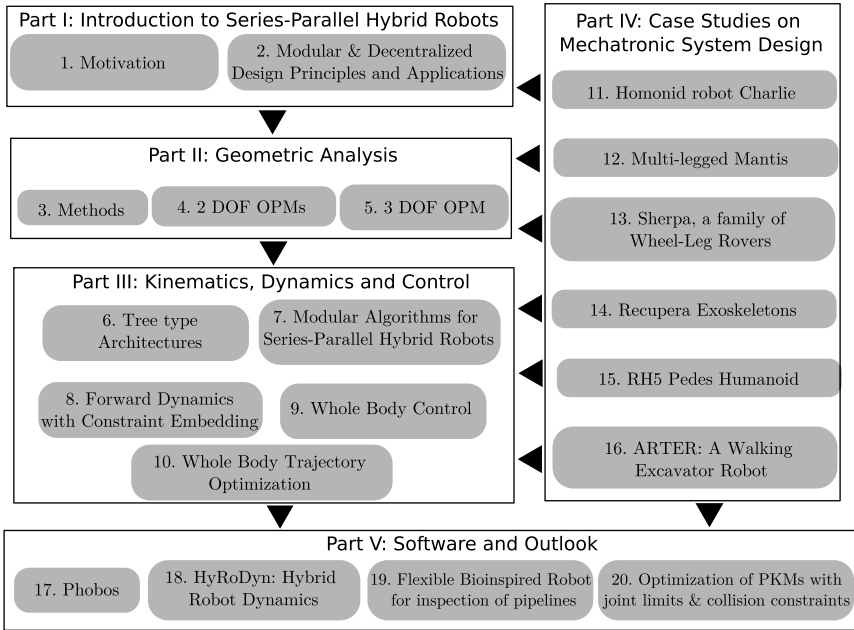


Figure 1.2 Book Outline.

domains. It also provides a survey on various approaches used in their modeling and control. The content presented in this chapter is based on the prior publication [4] (CC BY 4.0).

Part 2: Geometric analysis

This part collects the Chapters 3, 4, and 5 of this book as they essentially present the geometric analyses of parallel mechanisms.

Chapter 3 (Methods for geometric analysis of parallel mechanisms)

This chapter provides an overview of modern approaches such as screw theory and algebraic geometry in the geometric analysis of closed loop mechanisms. The content is adapted from author's PhD thesis [24].

Chapter 4 (2 DOF orientational parallel mechanisms)

This chapter presents a comprehensive geometric analysis of two DOF orientational parallel mechanism (OPM) which can be used as an abstraction to universal joint. The chapter introduces the mechanism's architecture, so-

lutions to forward and inverse geometric problems and workspace analysis using tools from computational algebraic geometry. Variants of this mechanism are used in RH5 humanoid in the ankle [25], wrist [26], and torso [24] designs.

Chapter 5 (3 DOF orientational parallel mechanism)

This chapter presents a thorough geometric analysis of a three DOF almost spherical parallel mechanism which can be used as an abstraction to a spherical joint. The chapter introduces the mechanism's architecture, solutions to forward and inverse geometric problems and workspace analysis using tools from computational algebraic geometry. A novel technique for solving its rotative inverse geometric problem using the concept of virtual joints is also discussed which enables the real time control of this mechanism. This mechanism has been used in Recupera Exoskeleton in hip and ankle designs. The content presented here is based on prior publications [27,28].

Part 3: Kinematics, dynamics, and control

This part collects the Chapters 6–10 which together describe the modular recursive methods for solving the kinematics and dynamics of series-parallel hybrid robots, and methods for whole body trajectory optimization and whole body control.

Chapter 6 (Kinematics and dynamics of tree-type systems)

This chapter presents the screw theory and Lie group methods for multi-body dynamics of tree type systems. The provided treatment is recursive in nature and rooted in graph theoretic description of such systems. Its application in solving the forward and inverse dynamics problems is presented. The content is adapted from author's PhD thesis [24].

Chapter 7 (Modular algorithms for kinematics and dynamics of series-parallel hybrid robots)

This chapter presents the modular methods for solving the kinematics and dynamics of series-parallel hybrid systems. The notion of the modularity is motivated from the robot design and is reflected in the modular enumeration of the graph. This allows a modular composition of loop closure constraints which can be used for modular and recursive computation of kinematics and dynamics. The explanation of the approach is aided with an example of a series-parallel humanoid leg. The content presented in this chapter is based on the prior publications [29,30].

Chapter 8 (Forward dynamics with constraint embedding for dynamic simulation)

This chapter presents the constraint embedding [31] approach for recursively computing the forward dynamics of a series-parallel hybrid robot in minimal coordinates. The content is based on the prior publication [30].

Chapter 9 (Whole body control)

This chapter presents a Quadratic Programming (QP) based whole body control approach for series parallel hybrid robots directly in the actuation space taking into account their full dynamics based on the work presented in [23].

Chapter 10 (Whole body trajectory optimization)

This chapter presents a investigative case study on whole body trajectory optimization for weight lifting task [22] using differential dynamic programming (DDP) based optimal control algorithm.

Part 4: Case studies on mechatronic system design

This part is a collection of Chapters 11–16 which describes mechatronic system design of various series-parallel hybrid robots developed at DFKI Robotics Innovation Center in the last decade.

Chapter 11 (Homonid robot Charlie)

This chapter presents the mechatronic system design of the homonid robot Charlie [32] which features an active spine (based on Stewart platform) and can perform both quadrupedal and bipedal locomotion modalities.

Chapter 12 (Multi-legged robot Mantis)

This chapter describes the design and development of a six-legged robot Mantis [12] inspired from insect mantis. The robot can use the front two arms for manipulation and its rear four legs can be used for locomotion. Its software and application areas are further described.

Chapter 13 (Sherpa, a family of wheel-leg rovers)

This chapter presents the mechatronic system design of a family of three wheel-legged hybrid rovers called Sherpa (SHerpa: Expandable Rover for Planetary Applications) rovers [17]. These robots have been tested in various field trials including the deserts of Utah and Morocco.

Chapter 14 (Recupera exoskeletons)

This chapter presents the mechatronic system design of improved Recupera-Reha Exoskeleton originally proposed in [33,34]. The improved design consist of a fully active wrist mechanism and various other design improvements in the spine joints and placement of hip and ankle actuators.

Chapter 15 (RH5 Pedes humanoid)

This chapter presents the new humanoid named RH5 Pedes developed out of the combination of lower body design from the RH5 humanoid [35,36] and the upper body design from RH5 Manus humanoid [37].

Chapter 16 (ARTER: a walking excavator robot)

This chapter describes a walking excavator robot called ARTER [38]. The system is a wheel-legged robot with manipulator arm with several closed loop kinematic joints actuated with hydraulic cylinders.

Part 5: Software and outlook

This part contains the Chapters 17–20 which describe the software implementation for modeling series-parallel hybrid robots and some advanced topics such as flexible series-parallel hybrid robots and design optimization.

Chapter 17 (Phobos)

This chapter describes a Blender based visual modeling tool named as Phobos which can help create and maintain the models of complex series-parallel hybrid robots.

Chapter 18 (HyRoDyn)

This chapter presents the modular software framework called HyRoDyn (HyRoDyn) which implements the methods presented in Chapters 6–8 for efficiently solving the kinematics and dynamics of series-parallel hybrid robots. It can exploit the analytical solutions for known parallel mechanisms (like the ones presented in Chapters 3–5) and use numerical loop closure techniques for cases where the analytical solution is not known leading to a hybrid numerical-analytical modeling approach which keeps a good balance between domain specific knowledge and generality. The content presented in this chapter is based on the prior publications [29,30].

Chapter 19 (Design of a flexible bio-inspired robot for inspection of pipelines)

This chapter presents the design of a biologically inspired series-parallel hybrid robot with flexible elements for pipeline inspection.

Chapter 20 (Optimization of parallel mechanisms with joint limits and collision constraints)

This chapter deals with the design optimization of parallel mechanisms using Nelder-Mead method. The proposed method can take into account joint limits and collision constraints in the design optimization process. This chapter is based on a prior publication [39].

References

- [1] J. Brinker, B. Corves, A survey on parallel robots with Delta-like architecture, in: The 14th IFToMM World Congress, 25.10.2015–30.10.2015, Taipei, Taiwan, 2015, <https://doi.org/10.6567/IFToMM.14TH.WC.PS13.003>.
- [2] D. Stewart, A platform with six degrees of freedom, *Proceedings of the Institution of Mechanical Engineers* 180 (1) (1965) 371–386, https://doi.org/10.1243/PIME_PROC_1965_180_029_02.
- [3] C.M. Gosselin, J. Hamel, The agile eye: a high-performance three-degree-of-freedom camera-orienting device, in: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, vol. 1, 1994, pp. 781–786, <https://doi.org/10.1109/ROBOT.1994.351393>.
- [4] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Mechatronics* 68 (2020) 102367, <https://doi.org/10.1016/j.mechatronics.2020.102367>, <https://www.sciencedirect.com/science/article/pii/S0957415820300477>.
- [5] M. To, P. Webb, An improved kinematic model for calibration of serial robots having closed-chain mechanisms, *Robotica* 30 (6) (2012) 963–971, <https://doi.org/10.1017/S0263574711001184>.
- [6] M. Gautier, W. Khalil, P.P. Restrepo, Identification of the dynamic parameters of a closed loop robot, in: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, 1995, pp. 3045–3050, <https://doi.org/10.1109/ROBOT.1995.525717>.
- [7] J.P. Merlet, Structural synthesis and architectures, in: *Parallel Robots*, in: *Solid Mechanics and Its Applications*, vol. 128, Springer, Netherlands, Dordrecht, 2006, pp. 19–94, https://doi.org/10.1007/1-4020-4133-0_2.
- [8] X. Kong, C.M. Gosselin, *Type Synthesis of Parallel Mechanisms*, 1st edition, Springer Publishing Company, Incorporated, 2007.
- [9] J. Serracín, L. Puglisi, R. Saltaren, G. Ejarque, J. Sabater-Navarro, R. Aracil, Kinematic analysis of a novel 2-d.o.f. orientation device, *Robotics and Autonomous Systems* 60 (6) (2012) 852–861.
- [10] S. Kumar, M. Simnofske, B. Bongardt, A. Mueller, F. Kirchner, Integrating mimic joints into dynamics algorithms – exemplified by the hybrid recupera exoskeleton, in: *Advances In Robotics (AIR-2017)*, June 28–July 2, New Delhi, India, ACM-ICPS, 2017.

- [11] D. Kuehn, F. Bernhard, A. Burchardt, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, Distributed computation in a quadrupedal robotic system, *International Journal of Advanced Robotic Systems* 11 (7) (2014) 110.
- [12] S. Bartsch, M. Manz, P. Kampmann, A. Dettmann, H. Hanff, M. Langosz, K. von Szadkowski, J. Hilljegerdes, M. Simnofske, P. Kloss, M. Meder, F. Kirchner, Development and control of the multi-legged robot MANTIS, in: *Proceedings of ISR 2016: 47st International Symposium on Robotics*, 2016, pp. 1–8.
- [13] S. Bai, S. Christensen, M.R.U. Islam, An upper-body exoskeleton with a novel shoulder mechanism for assistive applications, in: *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 1041–1046, <https://doi.org/10.1109/AIM.2017.8014156>.
- [14] S. Lohmeier, T. Buschmann, H. Ulbrich, F. Pfeiffer, Modular joint design for performance enhanced humanoid robot LOLA, in: *Proceedings 2006 IEEE International Conference on Robotics and Automation*, 2006. ICRA 2006, 2006, pp. 88–93.
- [15] B. Lee, C. Knabe, V. Orekhov, D. Hong, Design of a human-like range of motion hip joint for humanoid robots, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 46377, American Society of Mechanical Engineers, 2014, p. V05BT08A018, <https://doi.org/10.1115/DETC2014-35214>.
- [16] J. Nielsen, B. Roth, On the kinematic analysis of robotic mechanisms, *The International Journal of Robotics Research* 18 (12) (1999) 1147–1160, <https://doi.org/10.1177/02783649922067771>.
- [17] F. Cordes, F. Kirchner, A. Babu, Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain, *Journal of Field Robotics (JFR)* 35 (7) (2018) 1149–1181.
- [18] M.A. Hopkins, S.A. Ressler, D.F. Lahr, A. Leonessa, D.W. Hong, Embedded joint-space control of a series elastic humanoid, in: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3358–3365, <https://doi.org/10.1109/IROS.2015.7353845>.
- [19] P. Vonwirth, Modular control architecture for bipedal walking on a single compliant leg, Master's thesis, Robotics Research Lab, University of Kaiserslautern, unpublished; supervised by Steffen Schütz (September 2017).
- [20] N. Paine, J.S. Mehling, J. Holley, N.A. Radford, G. Johnson, C.-L. Fok, L. Sentis, Actuator control for the NASA-JSC Valkyrie humanoid robot: a decoupled dynamics approach for torque control of series elastic robots, *Journal of Field Robotics* 32 (3) (2015) 378–396, <https://doi.org/10.1002/rob.21556>, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21556>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21556>.
- [21] S. Kumar, J. Martensen, A. Mueller, F. Kirchner, Model simplification for dynamic control of series-parallel hybrid robots – a representative study on the effects of neglected dynamics shivesh, in: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5701–5708, <https://doi.org/10.1109/IROS40897.2019.8967786>.
- [22] M. Boukheddimi, R. Kumar, S. Kumar, J. Carpentier, F. Kirchner, Investigations into exploiting the full capabilities of a series-parallel hybrid humanoid using whole body trajectory optimization, in: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 10433–10439, <https://doi.org/10.1109/IROS55552.2023.10341784>.
- [23] D. Mronga, S. Kumar, F. Kirchner, Whole-body control of series-parallel hybrid robots, in: *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 228–234, <https://doi.org/10.1109/ICRA46639.2022.9811616>.
- [24] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.

- [25] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: J. Lenarcic, V. Parenti-Castelli (Eds.), *Advances in Robot Kinematics 2018*, Springer International Publishing, Cham, 2019, pp. 431–439.
- [26] C. Stoeffler, A. del Rio Fernandez, H. Peters, M. Schilling, S. Kumar, Kinematic analysis of a novel humanoid wrist parallel mechanism, in: O. Altuzarra, A. Kecskeméthy (Eds.), *Advances in Robot Kinematics 2022*, Springer International Publishing, Cham, 2022, pp. 348–355.
- [27] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, *Journal of Intelligent & Robotic Systems* 94 (2019) 303–325, <https://doi.org/10.1007/s10846-018-0792-x>.
- [28] S. Kumar, A. Nayak, B. Bongardt, A. Mueller, F. Kirchner, Kinematic Analysis of Active Ankle Using Computational Algebraic Geometry, Springer International Publishing, Cham, 2018, pp. 117–125.
- [29] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, *Journal of Mechanisms and Robotics* 12 (2) (2020) 021114, <https://doi.org/10.1115/1.4045941>.
- [30] R. Kumar, S. Kumar, A. Müller, F. Kirchner, Modular and hybrid numerical-analytical approach – a case study on improving computational efficiency for series-parallel hybrid robots, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 3476–3483, <https://doi.org/10.1109/IROS47612.2022.9981474>.
- [31] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*, Springer Verlag, 2011.
- [32] D. Kuehn, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, System design and testing of the hominid robot Charlie, *Journal of Field Robotics* 34 (4) (2017) 666–703, <https://doi.org/10.1002/rob.21662>, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21662>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21662>.
- [33] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the recupera exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019), <https://doi.org/10.3390/app9040626>, <http://www.mdpi.com/2076-3417/9/4/626>.
- [34] S. Kumar, M. Simnofske, B. Bongardt, A. Müller, F. Kirchner, Integrating mimic joints into dynamics algorithms: exemplified by the hybrid recupera exoskeleton, in: *Proceedings of the Advances in Robotics, AIR '17*, ACM, New York, NY, USA, 2017, pp. 27:1–27:6, <https://doi.org/10.1145/3132446.3134891>, <http://doi.acm.org/10.1145/3132446.3134891>.
- [35] J. Esser, S. Kumar, H. Peters, V. Bargsten, J. de Gea Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid robot, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), 2021, pp. 400–407, <https://doi.org/10.1109/HUMANOIDS47582.2021.9555770>.
- [36] I. Bergonzani, M. Popescu, S. Kumar, F. Kirchner, Fast dynamic walking with RH5 humanoid robot, in: 2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), 2023, pp. 1–8, <https://doi.org/10.1109/Humanoids57100.2023.10375193>.
- [37] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 01–07, <https://doi.org/10.1109/ICRA46639.2022.9811843>.

- [38] A. Babu, L.C. Danter, P. Willenbrock, S. Natarajan, D. Kuehn, F. Kirchner, ARTER: a walking excavator robot for autonomous and remote operations, *Automatisierungstechnik* 70 (10) (2022) 876–887, <https://doi.org/10.1515/auto-2022-0056> [cited 2024-09-29].
- [39] D.H. Salunkhe, G. Michel, S. Kumar, M. Sanguineti, D. Chablat, An efficient combined local and global search strategy for optimization of parallel kinematic mechanisms with joint limits and collision constraints, *Mechanism and Machine Theory* 173 (2022) 104796, <https://doi.org/10.1016/j.mechmachtheory.2022.104796>, <https://www.sciencedirect.com/science/article/pii/S0094114X22000684>.

This page intentionally left blank

CHAPTER 2

Modular and decentralized design principles and applications

Shivesh Kumar^a, Hendrik Wöhrle^b, José de Gea Fernández^a,
Andreas Müller^c, and Frank Kirchner^{a,d}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bIntelligente autonome Sensor- und Aktor-Systeme, Fachhochschule Dortmund, Dortmund, Germany

^cInstitute of Robotics, Johannes Kepler University, Linz, Austria

^dWorking Group Robotics, University of Bremen, Bremen, Germany

This chapter presents the state of the art in design and control of series-parallel hybrid robots. It is an updated version of [1] (licensed as CC by 4.0 <https://creativecommons.org/licenses/by/4.0/>). The chapter is organized as follows. Section 2.1 provides a survey on various series-parallel hybrid robots in various application domains. Section 2.2 presents the notion of modularity and distributivity in these hybrid robots from a hardware and software perspective. Section 2.3 discusses the modeling and control methods for these robots, highlighting the theoretical challenges and adopted solution approaches.

2.1 Series-parallel hybrid designs

Series-parallel hybrid robots combine the advantages of serial and parallel mechanisms. Similar designs are frequently used in the field of heavy machinery, e.g., cranes, excavator arms, bulldozers, etc., for a long time. Only in the last two decades has the robotics world begun to adopt these concepts for real-world applications. The overview about these types of systems presented in this section reviews the development of series-parallel hybrid robots in the following robot areas: Humanoid/Biped, Multi-legged Robotic Systems, Exoskeletons, or physical human-machine interfaces, and industrial automation.

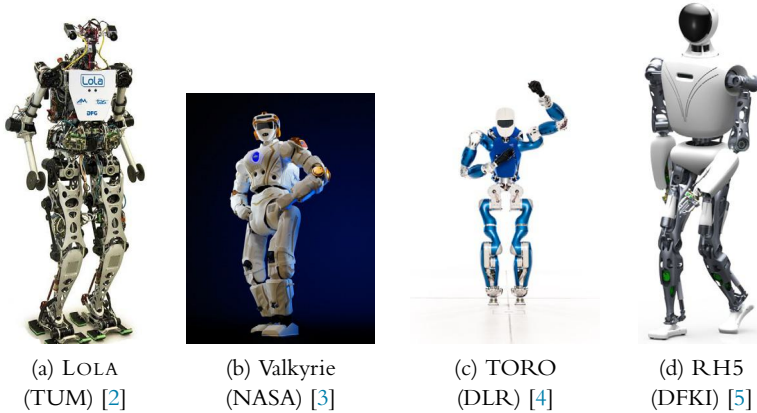


Figure 2.1 Humanoid robots with series-parallel hybrid design [1] (CC 4.0).

2.1.1 Humanoids

Humanoid robots are systems with two legs, torso, arms, and head that imitate human anatomy. They are complex mechatronic systems with highly interdependent technical aspects. Research in the field of humanoid robotics over the past few decades has shown that humanoids can be designed for high dynamic performance require a rigid structure and good mass distribution [6]. Both of these properties can be achieved by utilizing parallel mechanisms in the design. Fig. 2.1a shows the bipedal robot LOLA [2], which was developed in 2006 and is probably the first humanoid robot designed using a modular parallel joint concept. The non-linear transmission between the actuator space and output space in a parallel mechanism can be utilized advantageously by adjusting its design parameters to the torque-speed characteristics of typical movements of the robot [7]. Moreover, utilizing multi-DOF parallel mechanisms can also help in achieving a more uniform load distribution on the actuators. The design of NASA VALKYRIE humanoid [3], as shown in Fig. 2.1b, followed a similar design concept by utilizing PKM modules for its wrist, torso, and ankle joints. Similarly, the robot AILA [8] also used PKM modules for her wrist, neck, and torso joints. The torque-controlled humanoid TORO from DLR [4] and TALOS [9] (PAL Robotics) are largely tree type systems, but utilize a simple parallelogram linkage in their ankle for creating the pitch movement (see Fig. 2.1c). Humanoid robots THOR [10], [11], and SAFFIR [12] from Virginia Tech. use 2 DOF PKM for hip roll-yaw and ankle joints and utilize Hoeken's mechanism for hip pitch and knee joints. LARMBOT [13]

is a recently reported humanoid robot prototype which utilizes two linearly actuated tripod parallel mechanisms as legs and a cable driven parallel mechanism (CPDR) for its torso design. CARL is a compliant walking leg designed using parallel, redundant actuation that mimics the behavior of the human muscles [14]. RH5 is a lightweight and biologically inspired humanoid robot recently developed by DFKI-RIC and uses linkages and PKM modules for most of its joints, e.g., hip flexion–extension, knee, ankle, torso, and wrist [5] (see Fig. 2.1d). It weighs 64 kg and is designed for a payload capacity of 5 kg in each arm. RH5 Manus [15] is a powerful humanoid upperbody design built using a series–parallel hybrid architecture. It uses PKM modules in torso, elbow, and wrist joints. Kangaroo [16], the latest robot from PAL robotics, is a bipedal robot which, weighs 40 kg and 1.45 m tall, is capable of performing jumps. Its system design employs a series–parallel hybrid design architecture which places 5 out of 6 actuators at the hip and only 1 is placed at the femur. Also, various athletic bipedal robots from commercial companies such as Digit/Cassie [17] from Agility Robotics and Atlas [18] from Boston Dynamics employ closed-loop linkages across their robot designs, which utilize this design philosophy very effectively.

2.1.2 Multi-legged robots

Closed loop linkages and parallel mechanisms are increasingly being used in multi-legged robots which are specifically designed for high-payload applications. With their use, certain joints can be strengthened without compromising the overall leg inertia. The multi-legged robot MANTIS [19], depicted in Fig. 2.2a, contains PKMs of type 2-SPU+1U [20] in its ankle joints and slider–crank mechanisms that drive certain revolute joints in its legs and torso. Fig. 2.2b shows the hominid robot CHARLIE [21], featuring a Stewart platform of type 6-RUS as a six DOF active joint module in spine and neck. It also utilizes another parallel mechanism in the ankle joint. This robot supports both bipedal and quadrupedal walking gaits. SHERPATT rover, which is a wheeled-leg hybrid robot, uses a closed loop linkage in inner and outer leg joints [22], [23]. HERITAGEBOT [24] from University of Cassino is a hybrid robot which is capable of flying and legged locomotion on ground. Its modular design makes use of four tripod parallel mechanisms for the leg design.

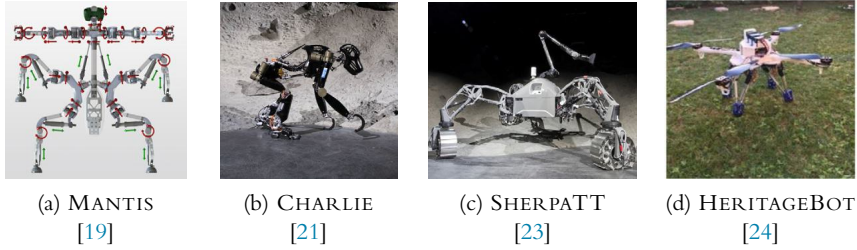


Figure 2.2 Multi-legged robots with series-parallel hybrid design [1] (CC 4.0).

2.1.3 Exoskeletons

In exoskeletons or physical man-machine interfaces, most joints require a limited range of motion, because most of the human joints like the wrist or ankle are not able to perform a 360° rotation movement. Hence, in exoskeletons based on serial kinematic chain, a physical limitation of joint movements is necessary for safety reasons. Software-based joint limits may fail, hence, additional mechanical end stops are required at each joint. The use of parallel mechanisms in exoskeletons can not only lead to a lightweight design, but also their limited workspace may be exploited as an additional safety feature. An early exoskeleton design utilizing closed loop linkage is BLEEX [25], as shown in Fig. 2.3a. Fig. 2.3b demonstrates a highly modular light weight RECUPERA full-body exoskeleton [26,27] with 32 active DOFs which is built by combining several higher DOF joint modules: a Stewart platform of type 6-UPS in torso, a double parallelogram [28] in shoulder for flexion-extension movement and ACTIVE ANKLE mechanism [29] as a 3-DOF joint in hip and ankle. Due to high modularity of its mechanics and electronics design, the upper body including the two arms can be mounted on a wheelchair for rehabilitation applications (see Fig. 2.3c). A passive double parallelogram mechanism has also been employed in the design of AXO-SUIT [30] and a parallelogram mechanism has been used in Harmony dual arm exoskeleton [31]. Another recent development is the SPHERICAL EXO SUIT as shown in Fig. 2.3d which employs AGILE WRIST mechanism as a 3-DOF spherical joint module at hip and ankle joints [32].

2.1.4 Industrial automation

Hybrid serial-parallel robots are also gaining recognition in the industrial applications. For high stiffness applications, serial robot designs are often

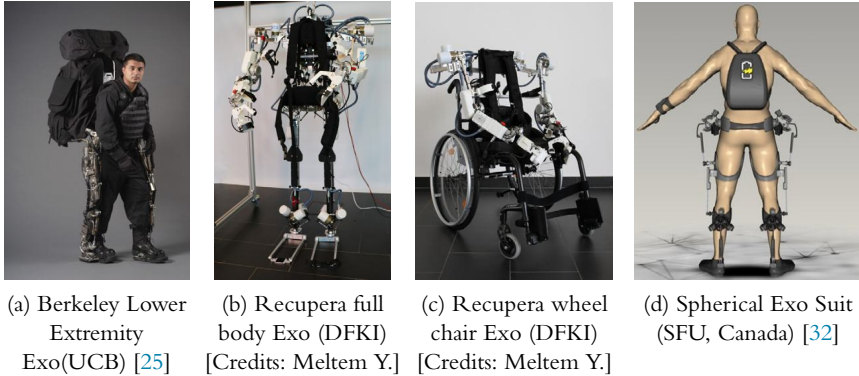


Figure 2.3 Exoskeletons with series-parallel hybrid design [1] (CC 4.0).

complemented by a parallelogram mechanism in the design (see, Fig. 2.4b). Industrial robots such as ABB's IRB4400, IRB6660, KUKA's KR 40-PA, KR 50-PA, KR 700-PA robots, and Comau's Smart NJ series, SR400 utilize parallelograms for increasing stiffness of one or several joints [33], [34]. Fig. 2.4a shows Logabex's LX4 robot, which piles four Stewart platform modules in series to achieve a redundant series-parallel hybrid manipulator with large workspace and good payload (75 kg) to weight (120 kg) ratio. The control of such a robot is difficult [35]. The 3-DOF DELTA robot has gained high popularity for fast pick-and-place operations in the industry. Soon, a demand for mounting an active wrist was realized and several such robots with 1, 2, and 3 DOF wrists were developed. A comparative study of various series-parallel delta robot designs can be found in [36]. FANUC M-3iA/6A Delta Robot is an interesting six-axis series-parallel hybrid robot designed for pick-and-place operations in the food industry (see Fig. 2.4c). It is also available in four-axis version with a single-axis wrist design.

2.2 Modular and distributive aspects

In this section, we present the notion of modularity and distributivity in the series-parallel hybrid robots from a hardware and software perspective. The subsection Hardware is further subdivided into Mechanics and Electronics design. The subsection Software discusses the software architectures used in the control of such robotic systems.

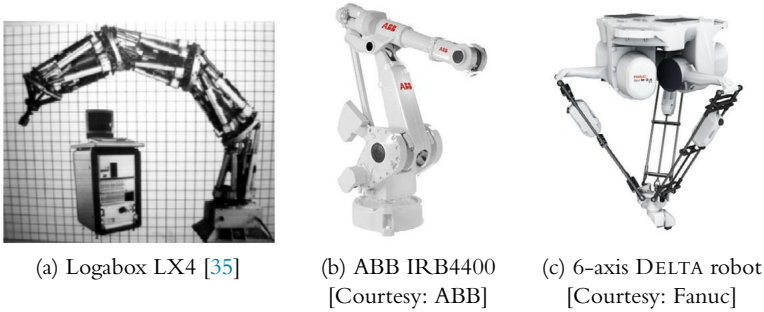


Figure 2.4 Some series-parallel hybrid robots used in industrial automation [1] (CC 4.0).

2.2.1 Hardware

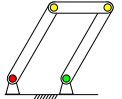
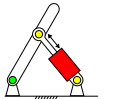
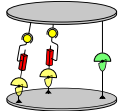
Hardware of any robotic system can be broadly categorized into mechanics and electronics domain. In the following, we discuss these aspects of series-parallel hybrid robots.

2.2.1.1 Mechanics

Hybrid robots are robots that are composed of a series of serial or parallel submechanisms. Table 2.1 and Table 2.2 present a list of closed-loop linkages and parallel submechanism modules, which have been utilized in series-parallel hybrid robot designs discussed in Section 2.1. In all cases, these submechanisms are used as a mechanical generator of m -dimensional motion subspaces of $SE(3)$, i.e., they are used as an abstraction to either an active lower pair joint (e.g., revolute joint, spherical joint, etc.) or universal joint, which are building blocks of most robotic systems. It can be immediately noticed from these tables that parallel submechanism modules are mostly used as abstractions to orientational joints. Exceptions are their application as 6-DOF joint in CHARLIE, RECUPERA-REHA exoskeleton, and 2R1T wrist in R1 humanoid [42]. The most popular abstractions are discussed in the following.

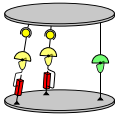
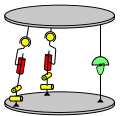
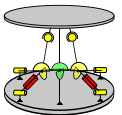
- 1-DOF Revolute joint: Abstraction of a revolute joint is basically done using variants of a four-bar linkage. For a simple 1 : 1 transmission, a parallelogram [4], [9] or double-parallelogram linkage [28] is often employed. In comparison to direct drive joints, the main motivation here is to reduce the resulting inertia of the robot or to create a virtual center of rotation. The slider-crank mechanism (identified as 1-RR \underline{P} R in Table 2.1) is used to exploit the non-linear transmission properties

Table 2.1 Linkages and PKM modules in series-parallel hybrid robots where platform coordinates can be measured directly. Red, green, and yellow colors denote active, EE, and other passive joints [1] (CC 4.0).

Mechanism		Practical Applications		
Schematic	Type	Mobility (m)	Application	Hybrid Robot
	1- <u>R</u> RRR	1 DOF rotational	Ankle pitch Series-elastic leg Ankle pitch Shoulder flexion–extension Joint 2 and/or Joint 3	TORO humanoid [4] ATRIAS [37] TALOS humanoid [9] RECUPERA-REHA [28], HARMONY [31] ABB IRB4400, KUKA KR 40-PA, etc.
	1-RR <u>P</u> R	1 DOF rotational	Hip, Torso Hip flexion–extension, Knee Inner and Outer leg joints Hip-Knee, Knee-Ankle joints	MANTIS [19] RH5 [5], HADE leg [38] SHERPATT rover [23] CARL [39]
	2- <u>S</u> PU+1U	2 DOF universal	Wrist, Torso Ankle Ankle Hip Roll-Yaw, Ankle Thruster steering	RH5 humanoid [5] MANTIS [19] LOLA humanoid [2] THOR [10], SAFFiR [12] AUV Leng, EurEx [40]

continued on next page

Table 2.1 (continued)

Mechanism		Practical Applications		
Schematic	Type	Mobility (m)	Application	Hybrid Robot
	2- <u>P</u> US+1U	2 DOF universal	Wrist, Ankle, and Torso	VALKYRIE humanoid [3]
	2-S <u>P</u> RR+1U	2 DOF universal	Ankle	RH5 [41]
	2-SU[1-RR <u>P</u> R]+1U	2 DOF universal	Ankle	CHARLIE [21]

of the mechanism. Also, the prismatic actuation gives the possibility to mount the actuator in longitudinal direction of the link, which is advantageous from a construction perspective.

- **2-DOF Universal joint:** Abstraction of a universal joint is very useful when the joint requires only a limited range of rotational motion and has to be placed away from the base of the robot. Hence, the most common application is the design of ankle joint in humanoids or legged robots. They also have been used in the design of wrist and torso mechanisms. All the orientational parallel manipulators that have been used in this context are equipped with a passive leg containing the universal joint [43]. Such designs are very advantageous because they do not have output singularities and provide good stiffness to the moving platform. Also, it is easy to install rotary encoders to measure the orientation of the platform directly, so that forward kinematics of the mechanism is not required to be solved in real time.
- **3-DOF Spherical joint:** Since the workspace of 3-DOF orientational parallel manipulators is limited, they have been mostly used in the design of exoskeletons for the abstraction of hip and ankle joints. *AGILE EYE* [44], which was originally developed as a fast camera orienting device, is used as hip and ankle module in *SPHERICAL EXO SUIT* concept. The disadvantage of this design is that it requires all the revolute joint axes to intersect at one point, which is not good for high payload or impact applications. *ACTIVE ANKLE* [45] is an interesting parallel mechanism which overcomes this problem by utilizing spatial quadrilateral mechanism in each leg, where only push-pull forces can exist in each leg. However, it is an almost spherical mechanism and is only suited for applications where small translation of the EE point can be tolerated.
- **Six-DOF free joint:** Many joints in humans like spine are actually very complicated and not easy to abstract using lower kinematic pairs. To provide a 6 DOF active spine to human-like robots, variants of *STEWART GOUGH* platform have been utilized. In this survey, one could find designs with both rotary (6-RUS) and prismatic actuation (6-UPS). The active spine can also be used as a 6-DOF force-torque sensor and improves the workspace of limbs and lowers the velocity requirements of other limb joints [46]. The disadvantage of using such mechanisms is the complicated nature of their forward kinematic problem.

Table 2.1 shows the list of linkages and parallel submechanisms where the platform coordinates (highlighted with green) are a subset of coordinates used to describe the mechanism. Since it is well known that it's difficult to solve the forward kinematics of the parallel robots in real time, there is a tendency to choose parallel mechanisms where additional sensors can be integrated to measure the platform coordinates. Table 2.2 shows the list of parallel submechanisms where the platform coordinates are not a subset of mechanism coordinates. Here, it's also not possible to put extra sensors to measure the platform coordinates directly, but in some cases they may be integrated in other passive joints to simplify the calculation of platform coordinates from actuator states. Hence, the use of such parallel submechanisms is less common in the design of series-parallel hybrid robots in comparison to the ones presented in Table 2.1. Overall, two observations can be made from this survey: submechanism modules are used as a mechanical generator of a motion subspace (revolute, universal, spherical, free joint, etc.) and the same type of submechanism with different physical parameters is utilized as a module to serve different purposes (ankle, wrist, torso joints, etc.) in the same robot.

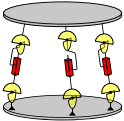
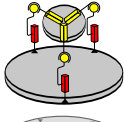
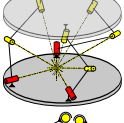
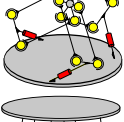
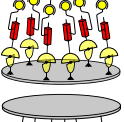
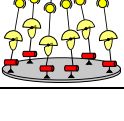
2.2.1.2 Electronics

An important development in electronic systems for robots, particularly useful for parallel and hybrid robots, is the use of *intra-system* decentralized or distributed processing and control architectures. To obtain such decentralized processing and control architectures, two aspects are especially important: (1) computations must be distributed inside the system, and (2) a communication medium is required.

In classical, i.e., centralized control systems, all actuators are connected to a central host computer system (Fig. 2.5(a)), which is used for processing of data and (nearly all) computations. In contrast, distributed processing systems (Fig. 2.5(b, c)) are composed of multiple *computing nodes* or *processing elements* that are connected through a network to exchange data with each other. Decentralized processing systems can be used to implement local control systems that control multiple joints in a synchronized manner (see Sections 2.2.2 and 2.3.4), which is especially important for controlling parallel and thus hybrid systems.

In both centralized and decentralized systems, a communication network is required to provide a networked control system [49], i.e., exchange data between sensors, actuators, and controllers [49]. Although there have been some developments in recent years with wireless networks for these

Table 2.2 PKM modules in series-parallel hybrid robots where platform coordinates can not be sensed. Red and yellow colors denote active and passive joints, respectively [1] (CC 4.0).

Mechanism		Practical Applications		
Schematic	Type	Mobility (m)	Application	Hybrid Robot
	3- <u>UPU</u>	3 DOF translational	Leg	LARMBOT [13], HERITAGEBOT [24]
	3- <u>PSP</u>	3 DOF 2 Rotational 1 Translation	Torso [47], Wrist [48]	R1 humanoid [42]
	3- <u>RRR</u>	3 DOF spherical	Hip, Ankle	SPHERICAL EXO SUIT [32]
	3- <u>R[2-SS]</u>	3 DOF almost spherical	Hip, Ankle	RECUPERA-REHA full-body exo [45]
	6- <u>UPS</u>	6 DOF free	Torso	RECUPERA-REHA full-body exo [26]
	6- <u>RUS</u>	6 DOF free	Torso, Neck	CHARLIE Hominidae [21]

needs [50], most intra-robot communication networks are still wired due to the uncertainties that still exist in wireless communication. The commonly used wired communication technologies can be divided roughly into the following categories: (1) specialized communication protocols for local intra-system communication and (2) technologies that use or are based on Ethernet.

Specialized communication protocols used in robots are often Field-bus systems that are also commonly used in other industrial or automotive applications, such as CAN/CAN-FD, DeviceNet, FlexRay, BACnet, or higher-level protocols like PROFIBUS DP [51], Modbus, or CC-Link, if they are used with another physical/data-link medium than Ethernet.

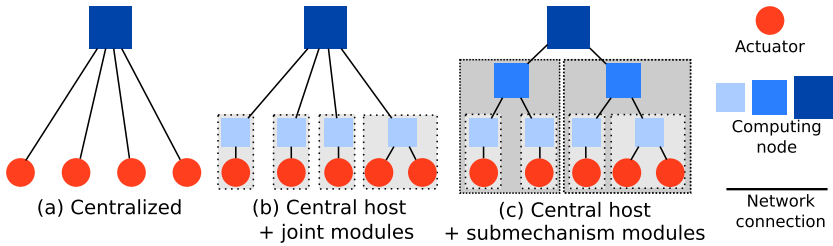


Figure 2.5 Different types of centralized and modular processing/control architectures [1] (CC 4.0).

Ethernet-based communication is used widely in different types of applications beyond robotics. However, from a robotic control perspective, a major problem of standard Ethernet-based communication is that it is not real-time capable. Hence, nowadays various modified Ethernet-based protocols with real-time support are gaining popularity [52], for example, PROFINET [53], EtherCAT [54], or SERCOS [55]. Although the bandwidth required to coordinate multiple joints is quite small (depending on the control strategy, the data exchanged consists parameters, e.g., desired position, speed, torque), the response time should be as short as possible. The nodes contribute to the overall processing of the system by performing specific processing tasks and providing additional capabilities, for example, peripherals. Distributed architectures are important for modularity, since they provide the possibility to build independent substructures of robot systems. Local controllers implemented in the distributed hardware can be used with high sensor sampling rates and a low delay in comparison to a centralized approach.

Similar to the different types of configurations used in distributed networked control systems [56], different types of network topologies are used in intra-robot communication. In robotics, a hierarchy with three types of modularity can be defined as follows: (1) joint-level modules, i.e., highly integrated, fully operational modules that are located in close proximity to an actuator and sensors which include all electronics that are required for signal processing, communication, and control. Commercial of the shelf examples are the Robotis Dynamixel and Dynamixel Pro series [57,58], the Kinova KA-58 and KA-75+ actuators [59], the Schunk PowerCube, PR 2, PRH, PDU 2, and PSM 2 series, TREE actuators [60], ANYdrive [61], and SMARCOS [62].

However, reusable joint modules are not only used commercially, but also developed by research facilities to build novel and experimental robotic

systems, such as the DLR LWR III robot [63] or DFKI Charlie [64] joints. Further examples support often additional features, such as joint-modules with series elastic actuation, as used in the WALK-MAN [65] and RoboSimian [66] robots, an integrated inertial measurement unit [67], or even space compliance such as the DFKI-X joint [68]. However, it is important to note that many research joint modules usually do not have integrated, but closely attached electronic controllers.

Series-parallel hybrid robots that contain reusable joint modules are the CHARLIE [21], MANTIS [19], RECUPERA [26], and RH5/RH5 MANUS [26] systems. In these examples, one or a few actuators are combined with a processing system that is responsible for the first level control of an actuator. Joint-specific limits (position, velocity, torque) can also be implemented in the joint-level controller for a hierarchical safety implementation.

Joint-level modules can be used to create systems with (2) submechanism-level modularity, i.e., more complex subsystems that need to be controlled as a whole. Examples are classical serial mechanisms like limbs or parallel mechanisms as shown in Table 2.1 and 2.2. Examples for such independently working modules are limbs in the ECCE humanoid robot [69] or the ROBONAUT hand [70] or the spine in CHARLIE [21]. One important point to note is that the use of these mechanisms requires to solve the (forward/inverse) kinematics and dynamics of the corresponding subsystem, which can be performed locally to maintain modularity. This constitutes the second-level control of the robotic system.

The next modularity level is (3) the complete robot system itself (which can in turn build teams or swarms with other robots). Regarding the processing architectures, it can be observed that even in robots that consist of independent modules, the distributed computing located in the modules are usually connected to a central host that is responsible for overall control. Different devices can be used to implement a distributed robotic control system, e.g., microcontrollers as in ECCE [69] or FPGA/processor combinations as in the ROBONAUT [71]/ROBONAUT 2 [72] and VALKYRIE [3] robots or CHARLIE [21], MANTIS [19], and RECUPERA [26].

2.2.1.3 Hardware acceleration

One development that has steadily gained importance in recent years is the use of application- or domain-specific hardware accelerators for robotics. In order to take this development into account, not too long ago a ROS2 Working Group on this topic was founded [73]. While the focus in the

robotics community is mainly on the use of hardware accelerators for perception [74], the use of hardware accelerators for control is also an important topic, especially for embedding highly complex control algorithms into the robot.

For this purpose, the term *Robomorphic Computing* was introduced [75], which encompasses the use of hardware accelerators for control adapted to the topology of the robot that can be used in FPGAs or to build custom ASICs. Here, the focus is on accelerating the computations required to compute the dynamics of the robot. However, one challenge in using hardware accelerators for robot control is the wide variety of different mechanism topologies in robotics, so that a single custom hardware accelerator is not sufficient to cover this variety of different systems in their entirety. Therefore, it is important that hardware accelerators for robot control are flexible and adaptable. One way to achieve this is to automatize generation of hardware accelerators for a given robotic robot system, i.e., its topology and characteristics. These should then take into account the system-dependent possibilities of parallelism and sparsity and achieve maximum efficiency.

These developments are particularly important for distributed control architectures, since these approaches make it possible to reduce the latency of communication in the distributed control architectures shown in Fig. 2.5: when using FPGAs or ASICs instead of CPUs at the higher control levels, i.e., beyond the joint modules, the readings from the sensors, and the commands to the actuators can be processed directly in the hardware accelerators without having to first transfer this data to the host processor. This can significantly reduce the latency of communication between the individual modules and the higher control levels.

2.2.2 Software

Software is important for modular robots in especially two aspects: (1) during the design phase for simulation and design optimization, and (2) during the runtime/application phase.

It is widely acknowledged that the construction of a multi-DOF robot is a highly challenging task. Hence, to simplify this design process, reusable modules can be used. This leads to the development of tools like H-ROS [76] and HRIM [77]. Especially for modular series-parallel hybrid robots like the systems introduced in Sec. 2.1, the use of appropriate design and modeling software is important (see Sec. 2.3.1). For modular robots, frameworks like D-Rock [78] can be used, which provide tools

for a model-based robot development process. For the remainder of this section, however, we will focus on the software frameworks that are used during the runtime/application phase.

In modular robots with distributed control, it is important to enable different components to exchange data in order to synchronize the control of several joints. This is especially important for modular robots that contain PKMs (see Sec. 2.3.4). In this case, the overall operation of the robot is distributed across multiple processors to maintain modularity, and, hence, several actuators have to be coordinated. Accordingly, some sort of communication middleware is required [79,80]. The middleware should provide a coherent interface to the different hardware components and provide mechanisms of data transfer. Many different robot middleware frameworks are available for the research community, the most popular examples probably the Robot Operating System (ROS) [81] or Yet Another Robot Platform (YARP) [82]. Each framework has specific advantages and core application areas (e.g., perception, manipulation, locomotion for YARP, and navigation, planning for ROS [83]).

All of these frameworks support some kind of modularity. In ROS, for example, the functionality is implemented inside so-called *nodes*. Each node runs inside a different process, the communication between nodes is mediated by an anonymous publish-subscribe middleware. However, for modular hybrid robotic systems, ROS has a number of shortcomings. ROS, for instance, has neither real-time capabilities nor is it possible to use ROS on small *bare metal* systems like microcontrollers in actuator modules (see Sec. 2.2.1.2). Finally, ROS does not support fieldbus or embedded communication systems like EtherCAT or even CAN directly. Hence, it is not possible to, e.g., instantiate ROS nodes that run on actuator modules to control a specific robot joint directly or even build a complete system out of them. Similar limitations are prevalent in most other robot frameworks: it is possible to build modular distributed systems on a software level, but not on a bare-metal hardware level.

Synchronization of different actuators and, hence, realtime capabilities, are of high importance for parallel and hybrid systems. A limited support of these capabilities are offered by the real-time variant of Orocos, Orocos RT [84], [85], and, hence, the derived Robot Construction Kit (RoCK) [86]. However, similar to ROS, these frameworks do not support low-level systems.

As a consequence, one aim of recent developments is to include bare-metal controllers and support field-busses. Following this approach will also

enhance the mechanism-level modularity in series-parallel hybrid robots, since in this case multiple joints need to be controlled in combination (see Sec. 2.2.1.2). Examples where this design principle has been used are the computation of the inverse kinematics of a Stewart platform representing the spine on a softcore CPU in CHARLIE [21] to satisfy constraints regarding size and power consumption which was built using NDLCOM to communicate between the controllers [87]. Another recent example is Finroc [88], which has been used to, e.g., implement a distributed controller for a series-parallel hybrid leg with redundant actuation [89].

Nevertheless, with increasing requirements and the need to use small embedded systems, new robot software frameworks like ROS 2 [90], are being developed. They support technologies like Protocol Buffers [91], ZeroMQ [92], and the Data Distribution Service (DDS) [93,94] and might satisfy the constraints of mechanism-level modularity in future applications. Although ROS 2 uses the DDS, which does not result in a performance benefit of ROS 2 over the plain TCP or UDP used in ROS [95], it supports real-time requirements and guarantees different Quality of Service (QoS) levels.

Another important development that has to be mentioned for completeness is the OPC Unified Architecture (UA), which is an industrial machine-to-machine communication protocol [96]. It not only provides mechanisms to transfer, but also semantically annotates the data. However, until now, OPC UA is mainly used in industrial robotics and only rarely in research.

One important point to notice is the communication overhead that results from a decentralized approach. Clearly, if software modules are distributed across different hardware modules, the time to transfer data between central controller and leaf modules must be considered. As discussed in Sec. 2.2.1.2, many different fieldbus or custom systems [76,87] are used in robotics. They can help to optimize the system regarding throughput, realtime requirements, and QoS level on the physical and data link layers and are required to fulfill these requirements at the higher layers. Hence, the choice of appropriate communication hardware is essential to build distributed control systems.

2.3 Modeling and control

Multi-body kinematics and dynamics has been an area of extensive research during the past decades. While the term kinematics encompasses problems

of position, velocity, and acceleration analysis, the term dynamics refers to problems associated with the study of forces and torques and their effect on motion of multi-body systems. Kinematics and dynamics essentially forms the basis of behavior modeling and control of any robotic system. The usage of parallel submechanisms in a robot's design introduces a new level of complexity in description, kinematics, dynamics, and control. In this subsection, we discuss these domain-specific difficulties and present some practical approaches used in controlling series-parallel hybrid robots.

2.3.1 Modeling

For describing serial robots, Denavit–Hartenberg (DH) parameters [97], and their modifications [98], have become the de facto standard: they specify each coordinate transformation by only four parameters instead of six parameters, due to the particular placement of local coordinate systems at specific locations. Since the placement of these coordinate systems requires manual effort, work has been invested to extract the DH parameters automatically from computer aided design (CAD) models of the serial manipulators [99]. In case of tree-type robots and robots with closed loops, the traditional notion of DH parameters cannot be used and hence various extensions have been proposed in the literature [100]. A comparison of various robot parameterization techniques with the Sheth–Uicker parameters can, for example, be found in [101]. Due to the dependency of the frame placement on the link geometries, the modeling becomes unintuitive, in particular for complex link shapes (for example, in exoskeletons or human-machine interfaces). For these reasons, standard open source robot description formats, such as URDF¹ (ROS), COLLADA² (OpenRAVE), or SDF³ (Gazebo), do not rely on DH parameters (or extensions) for representing the coordinate transforms and, instead, store the required transformations by six parameters. These coordinate transforms requested by these description formats can be automatically extracted from CAD environments by programs such as CAD-2-SIM [102] and SolidWorks to URDF Exporter.⁴ Even with the presence of such tools, it can be very time consuming to create complete robot description models for series-parallel hybrid robots, because most formats do not allow the possibility of a modular description. Further,

¹ <http://wiki.ros.org/urdf/XML>.

² <https://www.khronos.org/collada/>.

³ <http://sdformat.org/>.

⁴ http://wiki.ros.org/sw_urdf_exporter.

URDF does not allow for proper definition of closed loops, which often leads to complicated work-arounds when used for description of contemporary, complex robots. Other formats, such as SDF, allow the definition of parallel linkages, but do not further provide the functionality to explicitly define a spanning tree of a looped graph, necessary for a standardized tree representation of a model.

2.3.2 Kinematics

Kinematics is often regarded as the study of *geometry of motion*. Direct or inverse geometric problems generally result in a set of non-linear algebraic equations, regardless of the method of problem formulation. For serial robots, inverse geometric problem and for parallel robots, forward geometric problem are easy to formulate but difficult to solve. The three most useful solution techniques to deal with such problems are polynomial continuation, Gröbner bases, and elimination method [103]. J. P. Merlet in [104] presented a list of open problems for parallel robots, including forward kinematics, workspace and singularity analysis, singularity free trajectory planning, optimal design, etc., A rigorous kinematic analysis of generic series-parallel hybrid robots is also an open problem, because they carry kinematic complexity of both serial and parallel topologies. It is still quite appropriate to quote Nielsen and Roth [103] in this context: “Yet, a lot remains to be done before the subject of kinematic position analysis in robotics can be considered closed. Large structural classes, such as hybrid series and in-parallel systems, are just beginning to be treated in a systematic manner. Mainly, such studies are still done on a case-by-case basis, without a general theory and framework.” An example of such an analysis of 3-RPS-3-SPR type series-parallel hybrid manipulator can be found in [105].

Nevertheless, a Jacobian based numerical solution to the geometric problems in singularity-free regions is usually possible, which forms the basis of several multibody simulation software. Also, developments in the field of modern kinematics [106] such as screw theory and computational algebraic geometry has helped researchers a great deal to study specific families of parallel or series-parallel hybrid mechanisms. Also, it is important to note that comprehensive kinematic analysis of all the mechanisms listed in Table 2.1 and Table 2.2 is readily available in the literature.

2.3.3 Dynamics

Notable works in the field of robot dynamics include Newton–Euler [107], [98], and Lagrangian [98] formulations, the Decoupled Natural Or-

thogonal Compliment (DeNOC) formulation [108], and Kane's method [109], [110]. Traditionally, the equations of motion were described using 3D vectors – which quickly yields a large amount of equations for systems of connected bodies [111]. To address this issue, alternative compact and user-friendly formulations have been developed based on screw theory [107], [112] and Lie group theory [113], [114], which can easily be transformed into program code for modern computers.

Robots containing closed loops are subjected to additional geometric loop closure constraints which are difficult to resolve at position level and hence most multi-body dynamics software frameworks try to resolve them at velocity and acceleration levels and arrive at position constraints numerically. The Rigid Body Dynamics Library (RBDL) [115] and `OpenSim` [116] are some examples of open source libraries that implement such algorithms and import robot descriptions using various robot description formats. Another issue with series-parallel robots is the considerably large number of their spanning tree DOFs. Let n be the number of DOFs of the spanning tree representing a series-parallel hybrid robot, m be the total mobility of the robot, p be the number of actuators in the system and c be the number of independent closed loops. RH5 humanoid, which only contains relatively simple parallel mechanism modules (with less than 3 DOF) has 32 DOF ($m = p = 32$), $c = 15$ independent closed loops and $n = 76$ DOF in its spanning tree. For a complicated hybrid robot such as Recupera-Reha exoskeleton, the robot has $m = p = 30$, $c = 29$, and $n = 128$. Hence, it can be very challenging to solve the full dynamic model of such systems in real time. In most practical applications reported in literature, full dynamic model of the series-parallel hybrid robots is not exploited [117], [14], [39].

2.3.4 Control

As previously stated, parallel robots can provide higher stiffness, speed, and accuracy than their serial counterparts. However, oftentimes those theoretical capabilities are not easy to achieve in practical applications due to different error sources. While higher speeds have been vastly shown in parallel systems and high accuracies in many of them, a broad sense of higher accuracy and stiffness control capabilities are more difficult to be generically achieved in practical applications [118,119]. One reason is that accuracy is affected by different sources: on the one side, parallel systems might require more joints than a serial one for the same task, and thus the kinematics er-

rors due to assembly inaccuracies or simplified kinematics might negatively affect the final system's accuracy. On the other side, the stiffness control is more complex since it requires of a workspace analysis to locate optimal areas for higher stiffness (some areas are of very low stiffness) [120]. Both accuracy and stiffness control are additionally affected by the larger amount of singularities present in parallel robots, which need to be removed, basically by highly-redundant actuation [121][122][123]. Notwithstanding, more complex control strategies can also significantly contribute to having better dynamic performance, i.e., closer performance to the expected theoretical one. However, historically the same classical control strategies used in serial robots have been reused for parallel robots, and there is relatively little specific literature on control of parallel devices [124] in comparison with their serial counterparts. The approaches followed basically fall into two categories: (a) the model-free control schemes such as PID control [125,126], fuzzy control [127], use neural networks to learn dynamics without a priori knowledge of the system [128] or based on force feedback such as in the seminal work in [129], and (b) model-based control schemes such as optimal control [122] or the use of machine learning methods [119]. The use of same control methods as in serial robots leads to some limitations inherent to the nature of those methods. For instance, in serial robots it is a de facto standard to use joint space control, which is not the most suitable strategy for parallel robots and, consequently, not for hybrid systems. A parallel robot is described by its end-effector pose rather than by joint configuration as in serial systems. For that reason, Cartesian (or task) space control is more suitable for complex hybrid mechanisms, especially multipurpose systems that use redundancy to perform a number of tasks simultaneously. However, some task controllers will likely need information from the position in abstract joint space (ref. to Fig. 2.6), which for parallel systems is much more complex problem, usually solved numerically. The fact that forward kinematics is often not in closed form means that a certain joint configuration can lead to different end-effector poses.

Additionally, in rigid serial robots the assumption is that each actuator (abstract joint as in Fig. 2.6) can be considered almost dynamically decoupled from the others ones (especially a good assumption in actuators with high gear ratios) and thus, typically a linear PID controller is used at each axis. The assumption does not hold true for lightweight serial robots moving at high speeds, because the nonlinear dynamics cannot be properly compensated. Parallel robots would suffer the same issues, as they are highly nonlinear and highly-coupled systems by nature. Fortunately, model-based

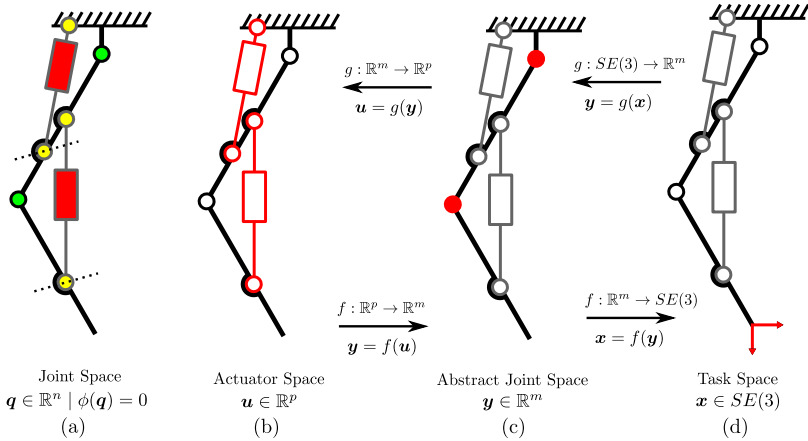


Figure 2.6 Different abstractions used for modeling and control of hybrid robots [1] (CC 4.0).

control approaches, such as computed-torque control [130], can alleviate some of those problems and compensate nonlinear dynamics, provided, once more, that there is an accurate robot model. Some other techniques such as robust control [131] can also be used with simplified dynamics or any kind of nonlinear controllers [126][132]. In any case, for parallel systems, the optimal strategy would be to work in Cartesian space, although that would mean facing the problem of directly measuring the end-effector pose, since using numerical estimation and forward kinematics to compute the end-effector pose could degrade again the dynamic performance of the robot.

2.3.5 Adopted practices

Series-parallel hybrid robots are highly complex mechatronic systems and generic treatment of such robots remains an open problem. Hence, there is always a trade-off between modeling depth, accuracy, and computational efficiency. However, modularity in robot design allows for certain abstractions that simplify their modeling and control. Such abstractions are shown in Fig. 2.6. While Fig. 2.6(a) captures the true complexity of the robot, due to absence of generic methods to model and control such systems, three different abstractions are adopted to simplify the modeling and control. In the following, we discuss the practices used in design, modeling, and control of series-parallel hybrid robots.

- **Design:** It is common practice to avoid any switch of assembly mode⁵ in the design of parallel submechanism modules for hybrid robots. It is achieved by choosing appropriate design parameters and physically restricting the movement of the joints in the parallel submechanism module. This ensures a unique forward kinematics solution for any given actuator input, which makes the behavior of submechanism modules similar to serially connected joints and greatly simplifies the modeling and control of such systems. However, it comes at a cost of workspace restriction, as certain kind of singularities can be crossed using appropriate trajectory planning in case of parallel robots [133].
- **Kinematics:** Forward and inverse kinematics of the submechanism module is usually solved to provide a bi-directional map between actuation space and abstract joint space (see Fig. 2.6(b) & (c)). Forward and inverse kinematics of parallel submechanism modules can be solved on local controllers either analytically or in resource-constrained systems with the help of Look Up Tables (LUTs). Analytical solutions are preferred when embedded hardware includes a microcontroller with a Floating Point Unit (FPU) (e.g., parallel joints in THOR [117]) or in cases when parallel submechanism modules bear more than two DOF (e.g., 6-DOF spine joint in Charlie [21]). As an alternative, LUTs can be used for systems without FPUs or FPGA-based local controllers (e.g., 1- or 2-DOF parallel joints in MANTIS [19] and 2-DOF ankle in Charlie [21]). Once a mapping is available, the robot can be treated purely as a serial or tree type structure, for which forward and inverse kinematics problems are easy to solve on the main controller (see Fig. 2.6(d)). Many series-parallel hybrid robots such as SherpaTT, MANTIS, and Charlie are kinematically controlled and compliance is realized only with the help of force/torque measurements. Further, it is not common practice to compute the full kinematic state of the spanning tree (see Fig. 2.6(a)), since such calculations can be computationally expensive.
- **Dynamics:** As pointed out before, the computation of full inverse dynamic model for hybrid robots can be computationally very expensive due to the large size of their spanning trees and the large number of loop closure constraints to be resolved. The moving parts inside a parallel submechanism module may have relatively small contribution to the overall dynamics of the system, which is essentially due to dynamics of

⁵ Assembly modes are different solutions to forward kinematics problem.

link segments, and joint friction, etc. [134]. Hence, an inverse dynamic model in abstract joint space is often combined with an inverse static model in actuation space to compute the actuator forces [117], [135]. This approach is used in torque-controlled series-parallel hybrid humanoids such as THOR, Valkyrie, Lola, etc. To the best knowledge of the author, the trade-off between the complete dynamic model and simplified dynamic model has not been reported in the literature.

2.4 Conclusion

This chapter presents the state of the art in design and control of series-parallel hybrid robots. Despite their kinematic complexity, such designs are becoming increasingly popular due to the mechanical advantage. Overall, one could conclude that by adopting certain practices in design, modeling, and control, it is possible to use such designs in various robotics applications. Modularity in kinematics and dynamics algorithms and their distributed implementation can make it easy to deal with high complexity of series-parallel hybrid robots. However, there is a lack of general framework for analyzing and modeling such systems. This will be addressed in the later chapters of this thesis.

References

- [1] S. Kumar, H. Wöhrle, J. de Gea Fernandez, A. Mueller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Mechatronics* 68 (2020) 102367.
- [2] S. Lohmeier, T. Buschmann, H. Ulbrich, F. Pfeiffer, Modular joint design for performance enhanced humanoid robot LOLA, in: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006*, pp. 88–93.
- [3] N.A. Radford, P. Strawser, K. Hambuchen, J.S. Mehling, W.K. Verdeyen, A.S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, R. Berka, R. Ambrose, M. Myles Markee, N.J. Fraser-Chanpong, C. McQuin, J.D. Yamokoski, S. Hart, R. Guo, A. Parsons, B. Wightman, P. Dinh, B. Ames, C. Blakely, C. Edmondson, B. Sommers, R. Rea, C. Tobler, H. Bibby, B. Howard, L. Niu, A. Lee, M. Conover, L. Truong, R. Reed, D. Chesney, R. Platt Jr, G. Johnson, C.-L. Fok, N. Paine, L. Sentis, E. Cousineau, R. Sinnet, J. Lack, M. Powell, B. Morris, A. Ames, J. Akinyode, Valkyrie: NASA's first bipedal humanoid robot, *Journal of Field Robotics* 32 (3) (2015) 397–419, <https://doi.org/10.1002/rob.21560>.
- [4] J. Engelsberger, A. Werner, C. Ott, B. Henze, M.A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, A. Albu-Schäffer, Overview of the torque-controlled humanoid robot TORO, in: *2014 IEEE-RAS International Conference on Humanoid Robots, 2014*, pp. 916–923.
- [5] H. Peters, P. Kampmann, M. Simnofske, Konstruktion eines zweibeinigen humanoiden Roboters, in: *2. VDI Fachkonferenz Humanoide Roboter, 2017*.

- [6] O. Stasse, T. Flayols, *An Overview of Humanoid Robots Technologies*, Springer International Publishing, Cham, 2019, pp. 281–310, https://doi.org/10.1007/978-3-319-93870-7_13.
- [7] S. Lohmeier, *Design and realization of a humanoid robot for fast and autonomous bipedal locomotion*, Ph.D. thesis, Technischen Universität München, 2010.
- [8] J. Lemberg, J. de Gea Fernández, M. Eich, D. Mronga, P. Kampmann, A. Vogt, A. Aggarwal, Y. Shi, F. Kirchner, AILA – design of an autonomous mobile dual-arm robot, in: 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 5147–5153, <https://doi.org/10.1109/ICRA.2011.5979775>.
- [9] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A.D. Prete, P. Souères, N. Mansard, F. Lamiroux, J.P. Laumond, L. Marchionni, H. Tome, F. Ferro, TALOS: a new humanoid research platform targeted for industrial applications, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017, pp. 689–695, <https://doi.org/10.1109/HUMANOIDS.2017.8246947>.
- [10] B. Lee, C. Knabe, V. Orekhov, D. Hong, Design of a human-like range of motion hip joint for humanoid robots, <https://doi.org/10.1115/DETC2014-35214>, 2014.
- [11] B.K.T.-S. Lee, *Design of a humanoid robot for disaster response*, Ph.D. thesis, Virginia Tech, 2014, <http://hdl.handle.net/10919/47492>.
- [12] D.F. Lahr, H. Yi, D.W. Hong, Biologically inspired design of a parallel actuated humanoid robot, *Advanced Robotics* 30 (2) (2016) 109–118, <https://doi.org/10.1080/01691864.2015.1094408>.
- [13] D. Cafolla, M. Wang, G. Carbone, M. Ceccarelli, LARMBot: a new humanoid robot with parallel mechanisms, in: V. Parenti-Castelli, W. Schiehlen (Eds.), *ROMANSY 21 – Robot Design, Dynamics and Control*, Springer International Publishing, Cham, 2016, pp. 275–283.
- [14] S. Schütz, A. Nejadfard, K. Mianowski, P. Vonwirth, K. Berns, CARL – a compliant robotic leg featuring mono- and biarticular actuation, in: *IEEE-RAS International Conference on Humanoid Robots*, 2017.
- [15] M. Boukhebbi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 01–07, <https://doi.org/10.1109/ICRA46639.2022.9811843>.
- [16] E.M. Hoffman, S. Kothakota, A. Moreno, A. Curti, N. Miguel, L. Marchionni, Whole-body kinematics modeling in presence of closed-linkages: application to the kangaroo biped robot, in: *International Conference on Robotics and Automation (ICRA) 2022, Workshop on New Frontiers of Parallel Robotics*, second edition, 2022.
- [17] J. Hurst, Walk this way: to be useful around people, robots need to learn how to move like we do, *IEEE Spectrum* 56 (3) (2019) 30–51, <https://doi.org/10.1109/MSPEC.2019.8651932>.
- [18] E. Guizzo, By leaps and bounds: an exclusive look at how Boston dynamics is redefining robot agility, *IEEE Spectrum* 56 (12) (2019) 34–39, <https://doi.org/10.1109/MSPEC.2019.8913831>.
- [19] S. Bartsch, M. Manz, P. Kampmann, A. Dettmann, H. Hanff, M. Langosz, K.v. Szadkowski, J. Hilljegerdes, M. Simnofske, P. Kloss, M. Meder, F. Kirchner, Development and control of the multi-legged robot mantis, in: *Proceedings of ISR 2016: 47st International Symposium on Robotics*, 2016, pp. 1–8.
- [20] J. Serracín, L. Puglisi, R. Saltaren, G. Ejarque, J. Sabater-Navarro, R. Aracil, Kinematic analysis of a novel 2-d.o.f. orientation device, *Robotics and Autonomous Systems* 60 (6) (2012) 852–861.

- [21] D. Kuehn, F. Bernhard, A. Burchardt, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, Distributed computation in a quadrupedal robotic system, *International Journal of Advanced Robotic Systems* 11 (7) (2014) 110.
- [22] F. Cordes, F. Kirchner, A. Babu, Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain, *Journal of Field Robotics (JFR)* 35 (7) (2018) 1149–1181.
- [23] F. Cordes, A. Babu, F. Kirchner, Static force distribution and orientation control for a rover with an actively articulated suspension system, in: *Proceedings of the 2017 IEEE/R SJ International Conference on Intelligent Robots and Systems. IEEE/R SJ International Conference on Intelligent Robots and Systems (IROS-17)*, Friendly People, Friendly Robots, Vancouver, BC, Canada, September 24–28, IEEE/R SJ, 2017.
- [24] M. Ceccarelli, D. Cafolla, M. Russo, G. Carbone, HeritageBot platform for service in cultural heritage frames, *International Journal of Advanced Robotic Systems* 15 (4) (2018) 1729881418790692, <https://doi.org/10.1177/1729881418790692>.
- [25] A.B. Zoss, H. Kazerooni, A. Chu, Biomechanical design of the Berkeley lower extremity exoskeleton (BLEEX), *IEEE/ASME Transactions on Mechatronics* 11 (2) (2006) 128–138.
- [26] E.A. Kirchner, N. Will, M. Simnofske, L.M.V. Benitez, B. Bongardt, M.M. Krell, S. Kumar, M. Mallwitz, A. Seeland, M. Tabie, H. Woehrl, M. Yuelsel, A. Hess, R. Buschfort, F. Kirchner, Recupera-Reha: exoskeleton technology with integrated biosignal analysis for sensorimotor rehabilitation, in: *Transdisziplinaere Konferenz SmartASSIST*, 2016, pp. 504–517.
- [27] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the RECUPERA exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019), <https://doi.org/10.3390/app9040626>.
- [28] S. Kumar, M. Simnofske, B. Bongardt, A. Mueller, F. Kirchner, Integrating mimic joints into dynamics algorithms – exemplified by the hybrid Recupera exoskeleton, in: *Advances in Robotics (AIR-2017)*, New Delhi, India, June 28–July 2, ACM-ICPS, 2017.
- [29] M. Simnofske, S. Kumar, B. Bongardt, F. Kirchner, Active ankle – an almost-spherical parallel mechanism, in: *47th International Symposium on Robotics (ISR)*, 2016.
- [30] S. Bai, S. Christensen, M.R.U. Islam, An upper-body exoskeleton with a novel shoulder mechanism for assistive applications, in: *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 1041–1046, <https://doi.org/10.1109/AIM.2017.8014156>.
- [31] B. Kim, A.D. Deshpande, An upper-body rehabilitation exoskeleton harmony with an anatomical shoulder mechanism: design, modeling, control, and performance evaluation, *The International Journal of Robotics Research* 36 (4) (2017) 414–435, <https://doi.org/10.1177/0278364917706743>.
- [32] S. Sadeqi, S.P. Bourgeois, E.J. Park, S. Arzanpour, Design and performance analysis of a 3-RRR spherical parallel manipulator for hip exoskeleton applications, *Journal of Rehabilitation and Assistive Technologies Engineering* 4 (2017) 2055668317697596, <https://doi.org/10.1177/2055668317697596>.
- [33] M. To, P. Webb, An improved kinematic model for calibration of serial robots having closed-chain mechanisms, *Robotica* 30 (6) (2012) 963–971, <https://doi.org/10.1017/S0263574711001184>.
- [34] M. Gautier, W. Khalil, P.P. Restrepo, Identification of the dynamic parameters of a closed loop robot, in: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, 1995, pp. 3045–3050, <https://doi.org/10.1109/ROBOT.1995.525717>.

- [35] Structural synthesis and architectures, in: *Parallel Robots*, Springer Netherlands, Dordrecht, 2006, pp. 19–94, https://doi.org/10.1007/1-4020-4133-0_2.
- [36] J. Brinker, N. Funk, P. Ingenlath, Y. Takeda, B. Corves, Comparative study of serial-parallel delta robots with full orientation capabilities, *IEEE Robotics and Automation Letters* 2 (2) (2017) 920–926, <https://doi.org/10.1109/LRA.2017.2654551>.
- [37] C. Hubicki, J. Grimes, M. Jones, D. Renjewski, A. Spröwitz, A. Abate, J. Hurst, ATRIAS: design and validation of a tether-free 3D-capable spring-mass bipedal robot, *The International Journal of Robotics Research* 35 (12) (2016) 1497–1521, <https://doi.org/10.1177/0278364916648388>.
- [38] E. Garcia, J.C. Arevalo, G. Muñoz, P. Gonzalez-de Santos, On the biomimetic design of agile-robot legs, *Sensors* 11 (12) (2011) 11305–11334, <https://doi.org/10.3390/s111211305>.
- [39] A. Nejadfard, S. Schütz, P. Vonwirth, K. Mianowski, B. Karsten, Moment arm analysis of the biarticular actuators in compliant robotic leg CARL, in: *Conference on Biomimetic and Biohybrid Systems*, Springer, Springer International Publishing, 2018, pp. 348–360.
- [40] M. Hildebrandt, J. Albiez, M. Fritsche, J. Hilljegerdes, P. Kloss, M. Wirtz, F. Kirchner, Design of an autonomous under-ice exploration system, in: 2013 OCEANS, San Diego, 2013, pp. 1–6, <https://doi.org/10.23919/OCEANS.2013.6741164>.
- [41] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Mueller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: M. Carricato (Ed.), *Advances in Robot Kinematics*, Springer Verlag GmbH, Cham, 2018.
- [42] A. Parmiggiani, L. Fiorio, A. Scalzo, A.V. Sureshbabu, M. Randazzo, M. Maggiali, U. Pattacini, H. Lehmann, V. Tikhonoff, D. Domenichelli, A. Cardellino, P. Congiu, A. Pagnin, R. Cingolani, L. Natale, G. Metta, The design and validation of the R1 personal humanoid, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 674–680, <https://doi.org/10.1109/IROS.2017.8202224>.
- [43] C.-H. Kuo, J.S. Dai, Task-oriented structure synthesis of a class of parallel manipulators using motion constraint generator, *Mechanism and Machine Theory* 70 (2013) 394–406.
- [44] C.M. Gosselin, J. Hamel, The agile eye: a high-performance three-degree-of-freedom camera-orienting device, in: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, vol. 1, 1994, pp. 781–786, <https://doi.org/10.1109/ROBOT.1994.351393>.
- [45] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, *Journal of Intelligent & Robotic Systems* 94 (2019) 303–325, <https://doi.org/10.1007/s10846-018-0792-x>.
- [46] D. Kuehn, A. Dettmann, F. Kirchner, Analysis of using an active artificial spine in a quadruped robot, in: 2018 4th International Conference on Control, Automation and Robotics (ICCAR) International Conference on Control, Automation and Robotics (ICCAR-2018), April 20–23, Auckland, New Zealand, 2018, IEEE Xplore online, <https://ieeexplore.ieee.org/document/8384641>.
- [47] L. Fiorio, A. Scalzo, L. Natale, G. Metta, A. Parmiggiani, A parallel kinematic mechanism for the torso of a humanoid robot: design, construction and validation, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 681–688, <https://doi.org/10.1109/IROS.2017.8202225>.
- [48] A.V. Sureshbabu, J.H. Chang, L. Fiorio, A. Scalzo, G. Metta, A. Parmiggiani, A parallel kinematic wrist for the R1 humanoid robot, in: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2017, pp. 1215–1220, <https://doi.org/10.1109/AIM.2017.8014184>.
- [49] L. Zhang, H. Gao, O. Kaynak, Network-induced constraints in networked control systems—a survey, *IEEE Transactions on Industrial Informatics* 9 (1) (2012) 403–416.

- [50] D.-I. Curiac, Towards wireless sensor, actuator and robot networks: conceptual framework, challenges and perspectives, *Journal of Network and Computer Applications* 63 (2016) 14–23, <https://doi.org/10.1016/j.jnca.2016.01.013>, <https://www.sciencedirect.com/science/article/pii/S1084804516000345>.
- [51] PROFIBUS & PROFINET International (PI), PROFIBUS, <https://www.profibus.com/technology/profibus/>. (Accessed 9 June 2018).
- [52] C.S.V. Gutiérrez, L.U.S. Juan, I.Z. Ugarte, V.M. Vilches, Time-sensitive networking for robotics, *CoRR*, arXiv:1804.07643 [abs], 2018.
- [53] PROFIBUS & PROFINET International (PI), PROFINET the leading industrial ethernet standard, <https://www.profibus.com/technology/profinet/>. (Accessed 9 June 2018).
- [54] EtherCAT Technology Group, EtherCAT – the ethernet fieldbus, <https://www.ethercat.org/en/technology.html>, 2003. (Accessed 6 September 2018).
- [55] Sercos International e. V., SERCOS – the automation bus, <https://www.sercos.org>, 2005. (Accessed 6 September 2018).
- [56] X. Ge, F. Yang, Q.-L. Han, Distributed networked control systems: a brief overview, *Information Sciences* 380 (2017) 117–131, <https://doi.org/10.1016/j.ins.2015.07.047>, <https://www.sciencedirect.com/science/article/pii/S0020025515005551>.
- [57] Robotis, Dynamixel series, <http://www.robotis.us/dynamixel>. (Accessed 22 August 2018).
- [58] Robotis, Dynamixel pro series, <http://www.robotis.us/dynamixel-pro>. (Accessed 22 August 2018).
- [59] Kinova, Kinovamactuator, <https://www.kinovarobotics.com/en/products/actuator-series>, 2018.
- [60] TREE, Technology for Robotic systems Entering real Environments (TREE) Actuators, <https://www.treerobotics.eu/actuators/tree-robotics-actuators-info-sheet/file>, 2018. (Accessed 9 October 2018).
- [61] ANYbotics, ANYdrive, <https://www.anybotics.com/anydrive/>, 2016. (Accessed 15 May 2018).
- [62] V. Ducastel, K. Langlois, M. Rossini, V. Grosu, B. Vanderborght, D. Lefeber, T. Verstraten, J. Geeroms, SMARCOS: off-the-shelf smart compliant actuators for human-robot applications, *Actuators* 10 (11) (2021), <https://doi.org/10.3390/act10110289>, <https://www.mdpi.com/2076-0825/10/11/289>.
- [63] A.A. Schaeffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, G. Hirzinger, The DLR lightweight robot: design and control concepts for robots in human environments, *Industrial Robot: An International Journal* 34 (5) (2007) 376–385, <https://doi.org/10.1108/01439910710774386>.
- [64] D. Kuehn, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, System design and field testing of the hominid robot Charlie, *Journal of Field Robotics* 34 (4) (2016) 666–703.
- [65] F. Negrello, M. Garabini, M.G. Catalano, J. Malzahn, D.G. Caldwell, A. Bicchi, N.G. Tsagarakis, A modular compliant actuator for emerging high performance and fall-resilient humanoids, in: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015, pp. 414–420, <https://doi.org/10.1109/HUMANOIDS.2015.7363567>.
- [66] P. Hebert, M. Bajracharya, J. Ma, N. Hudson, A. Aydemir, J. Reid, C. Bergh, J. Borders, M. Frost, M. Hagman, J. Leichty, P. Backes, B. Kennedy, P. Karplus, B. Satzinger, K. Byl, K. Shankar, J. Burdick, Mobile manipulation and mobility as manipulation—design and algorithms of RoboSimian, *Journal of Field Robotics* 32 (2) (2015) 255–274, <https://doi.org/10.1002/rob.21566>.
- [67] S. Rader, L. Kaul, P. Weiner, T. Asfour, Highly integrated sensor-actuator-controller units for modular robot design, in: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2017, pp. 1160–1166, <https://doi.org/10.1109/AIM.2017.8014175>.

- [68] R. Sonsalla, H. Hanff, P. Schöberl, T. Stark, N.A. Mulsow, DFKI-X: a novel, compact and highly integrated robotics joint for space applications, in: Proceedings of the 17th European Space Mechanisms and Tribology Symposium. European Space Mechanisms and Tribology Symposium (ESMATS-2017), September 20–22, Hatfield, United Kingdom, ESMATS, 2017.
- [69] M. Jäntschi, S. Wittmeier, A. Knoll, Distributed control for an anthropomorphic robot, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 5466–5471, <https://doi.org/10.1109/IROS.2010.5651169>.
- [70] L.B. Bridgwater, C. Ihrke, M.A. Diffler, M.E. Abdallah, N.A. Radford, J. Rogers, S. Yayathi, R.S. Askew, D.M. Linn, The Robonaut 2 hand – designed to do work with tools, in: International Conference on Robotics and Automation (ICRA), IEEE, 2012, pp. 3425–3430.
- [71] R.O. Ambrose, H. Aldridge, R.S. Askew, R.R. Burrige, W. Bluethmann, M. Diffler, C. Lovchik, D. Magruder, F. Rehnmark, Robonaut: NASA's space humanoid, IEEE Intelligent Systems & Their Applications 15 (4) (2000) 57–63.
- [72] M.A. Diffler, J. Mehling, M.E. Abdallah, N.A. Radford, L.B. Bridgwater, A.M. Sanders, R.S. Askew, D.M. Linn, J.D. Yamokoski, F. Permenter, et al., Robonaut 2 – the first humanoid robot in space, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2178–2183.
- [73] V.M. Vilches, ROS 2 hardware acceleration working group, Online, <https://github.com/ros-acceleration>, 2021.
- [74] Acceleration Robotics, Hardware acceleration report in robotics, Tech. Rep., 2022.
- [75] S.M. Neuman, B. Plancher, T. Bourgeat, T. Tambe, S. Devadas, V.J. Reddi, Robomorphic computing: a design methodology for domain-specific accelerators parameterized by robot morphology, in: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2021, pp. 674–686.
- [76] L. Zhang, P. Slaets, H. Bruyninckx, An open embedded hardware and software architecture applied to industrial robot control, in: 2012 IEEE International Conference on Mechatronics and Automation, 2012, pp. 1822–1828, <https://doi.org/10.1109/ICMA.2012.6285098>.
- [77] I. Zamalloa, I. Muguruza, A. Hernández, R. Kojcev, V. Mayoral, An information model for modular robots: the hardware robot information model (HRIM), arXiv preprint, arXiv:1802.01459, 2018.
- [78] D-RoCK, Models, methods and tools for the model based software development of robots, <https://robotik.dfki-bremen.de/en/research/projects/d-rock.html>, 2018. (Accessed 5 October 2018).
- [79] N. Mohamed, J. Al-Jaroodi, I. Jawhar, A review of middleware for networked robots, International Journal of Computer Science and Network Security 9 (5) (2009) 139–148.
- [80] A. Elkady, T. Sobh, Robotics middleware: a comprehensive literature survey and attribute-based bibliography, Journal of Robotics 2012 (2012).
- [81] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.
- [82] G. Metta, P. Fitzpatrick, L. Natale, YARP: yet another robot platform, International Journal of Advanced Robotic Systems 3 (1) (2006) 8, <https://doi.org/10.5772/5761>.
- [83] M. Aragão, P. Moreno, A. Bernardino, Middleware interoperability for robotics: a ROS–YARP framework, Frontiers in Robotics and AI 3 (2016) 64, <https://doi.org/10.3389/frobt.2016.00064>, <https://www.frontiersin.org/article/10.3389/frobt.2016.00064>.
- [84] H. Bruyninckx, Open robot control software: the OROCOS project, in: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.

- 01CH37164), vol. 3, 2001, pp. 2523–2528, <https://doi.org/10.1109/ROBOT.2001.933002>.
- [85] H. Bruyninckx, P. Soetens, B. Koninckx, The real-time motion control core of the Orocos project, in: 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), vol. 2, 2003, pp. 2766–2771, <https://doi.org/10.1109/ROBOT.2003.1242011>.
- [86] S. Joyeux, Rock: the robot construction kit, <https://www.rock-robotics.org/stable/>, 2010. (Accessed 5 October 2018).
- [87] M. Zenzes, P. Kampmann, T. Stark, M. Schilling, NDLCOM: simple protocol for heterogeneous embedded communication networks, in: Proceedings of the Embedded World Exhibition & Conference, Nuremberg, Germany, 2016, pp. 23–25.
- [88] M. Reichardt, T. Föhst, K. Berns, Introducing FINROC: a convenient real-time framework for robotics based on a systematic design approach, Tech. Rep., Robotics Research Lab, Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany, 2012.
- [89] M. Reichardt, S. Schütz, K. Berns, One fits more—on highly modular quality-driven design of robotic frameworks and middleware, *Journal of Software Engineering for Robotics* 8 (1) (2017) 141–153.
- [90] Open Source Robotics Foundation (OSRF), ROS2, <https://github.com/ros2/>, 2016.
- [91] Google, Protocol Buffers, <https://developers.google.com/protocol-buffers/>, 2017. (Accessed 18 May 2018).
- [92] iMatix, ZeroMQ distributed messaging, <http://zeromq.org/>, 2017. (Accessed 18 May 2018).
- [93] G. Pardo-Castellote, OMG data-distribution service: architectural overview, in: 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings, 2003, pp. 200–206, <https://doi.org/10.1109/ICDCSW.2003.1203555>.
- [94] J.M. Schlesselman, G. Pardo-Castellote, B. Farabaugh, OMG data-distribution service (DDS): architectural update, in: Military Communications Conference, vol. 2, IEEE, 2004, pp. 961–967.
- [95] Y. Maruyama, S. Kato, T. Azumi, Exploring the performance of ROS2, in: 2016 International Conference on Embedded Software (EMSOFT), 2016, pp. 1–10, <https://doi.org/10.1145/2968478.2968502>.
- [96] W. Mahnke, S.-H. Leitner, M. Damm, OPC Unified Architecture, Springer Science & Business Media, 2009.
- [97] D. Hartenberg, R. Scheunemann, A kinematic notation for lower-pair mechanisms based on matrices, *Journal of Applied Mechanics* 22 (2) (1955) 215–221, <https://doi.org/10.1115/1.4011045>.
- [98] W. Khalil, E. Dombre, Modeling, Identification and Control of Robots, 3rd edition, Taylor & Francis, Inc., Bristol, PA, USA, 2002.
- [99] C. Rajeevchana, S. Saha, S. Kumar, Automatic extraction of DH parameters of serial manipulators using line geometry, in: The 2nd International Conference on Multibody System Dynamics, 2012.
- [100] W. Khalil, J. Kleinfinger, A new geometric notation for open and closed-loop robots, in: Proceedings. 1986 IEEE International Conference on Robotics and Automation, vol. 3, 1986, pp. 1174–1179, <https://doi.org/10.1109/ROBOT.1986.1087552>.
- [101] B. Bongardt, Sheth–Uicker convention revisited, *Mechanism and Machine Theory* 69 (2013) 200–229.
- [102] B. Bongardt, CAD-2-SIM – Kinematic Modeling of Mechanisms Based on the Sheth–Uicker Convention, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 465–477.

- [103] J. Nielsen, B. Roth, On the kinematic analysis of robotic mechanisms, *The International Journal of Robotics Research* 18 (12) (1999) 1147–1160, <https://doi.org/10.1177/02783649922067771>.
- [104] J.P. Merlet, Parallel robots: open problems, in: 9th International Symposium on Robotics Research, 1999.
- [105] A. Nayak, S. Caro, P. Wenger, Kinematic analysis of the 3-RPS-3-SPR series-parallel manipulator, *Robotica* 37 (7) (2019) 1240–1266, <https://doi.org/10.1017/S0263574718000826>.
- [106] J.M. McCarthy, Polynomials, computers, and kinematics for the 21st century, in: J.M. McCarthy (Ed.), *21st Century Kinematics*, Springer London, London, 2013, pp. 1–12.
- [107] R. Featherstone, *Rigid Body Dynamics Algorithm*, Springer, 2008, <https://doi.org/10.1007/978-1-4899-7560-7>.
- [108] A.K. Rao, S. Saha, P. Rao, Dynamics modelling of hexaslides using the decoupled natural orthogonal complement matrices, *Multibody System Dynamics* 15 (2) (2006) 159–180.
- [109] M. Lesser, A geometrical interpretation of Kane's equations, *Proceedings: Mathematical and Physical Sciences* 436 (1896) (1992) 69–87.
- [110] K.W. Buffinton, *Robotics and Automation Handbook*, CRC Press, 2005, Ch. Kane's Method in Robotics.
- [111] J.Y.S. Luh, M.W. Walker, R.P.C. Paul, On-line computational scheme for mechanical manipulators, *ASME Journal of Dynamic Systems, Measurement, and Control* (1980).
- [112] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*, Springer Verlag, 2011.
- [113] A. Müller, Screw and Lie group theory in multibody kinematics, *Multibody System Dynamics* 43 (2018) 37–70, <https://doi.org/10.1007/s11044-017-9582-7>.
- [114] A. Müller, Screw and Lie group theory in multibody dynamics, *Multibody System Dynamics* 42 (2) (2018) 219–248, <https://doi.org/10.1007/s11044-017-9583-6>.
- [115] M.L. Felis, RBDL: an efficient rigid-body dynamics library using recursive algorithms, *Autonomous Robots* 41 (2) (2017) 495–511.
- [116] S.L. Delp, F.C. Anderson, A.S. Arnold, P. Loan, A. Habib, C.T. John, E. Guendelman, D.G. Thelen, OpenSim: open-source software to create and analyze dynamic simulations of movement, *IEEE Transactions on Biomedical Engineering* 54 (11) (2007) 1940–1950, <https://doi.org/10.1109/TBME.2007.901024>.
- [117] M.A. Hopkins, S.A. Ressler, D.F. Lahr, A. Leonessa, D.W. Hong, Embedded joint-space control of a series elastic humanoid, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 3358–3365, <https://doi.org/10.1109/IROS.2015.7353845>.
- [118] F. Paccot, N. Andreff, P. Martinet, A review on the dynamic control of parallel kinematic machines: theory and experiments, *The International Journal of Robotics Research* 28 (3) (2009) 395–416, <https://doi.org/10.1177/0278364908096236>.
- [119] H. Abdellatif, B. Heimann, Advanced model-based control of a 6-DOF hexapod robot: a case study, *IEEE/ASME Transactions on Mechatronics* 15 (2) (2010) 269–279, <https://doi.org/10.1109/TMECH.2009.2024682>.
- [120] J.-P. Merlet, Parallel robots: open problems, in: J.M. Hollerbach, D.E. Koditschek (Eds.), *Robotics Research*, Springer London, London, 2000, pp. 27–32.
- [121] H. Cheng, Y.-K. Yiu, Z. Li, Dynamics and control of redundantly actuated parallel manipulators, *IEEE/ASME Transactions on Mechatronics* 8 (4) (2003) 483–491, <https://doi.org/10.1109/TMECH.2003.820006>.
- [122] G.F. Liu, H. Cheng, Z.H. Xiong, X.Z. Wu, Y.L. Wu, Z.X. Li, Distribution of singularity and optimal control of redundant parallel manipulators, in: *Proceedings 2001*

- IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the Next Millennium (Cat. No. 01CH37180), vol. 1, 2001, pp. 177–182, <https://doi.org/10.1109/IRROS.2001.973355>.
- [123] A. Muller, Internal preload control of redundantly actuated parallel manipulators—its application to backlash avoiding control, *IEEE Transactions on Robotics* 21 (4) (2005) 668–677, <https://doi.org/10.1109/TRO.2004.842341>.
- [124] F.H. Ghorbel, O. Chetelat, R. Gunawardana, R. Longchamp, Modeling and set point control of closed-chain mechanisms: theory and experiment, *IEEE Transactions on Control Systems Technology* 8 (5) (2000) 801–815, <https://doi.org/10.1109/87.865853>.
- [125] M.A. Khosravi, H.D. Taghirad, Robust PID control of fully-constrained cable driven parallel robots, *Mechatronics* 24 (2) (2014) 87–97, <https://doi.org/10.1016/j.mechatronics.2013.12.001>, <http://www.sciencedirect.com/science/article/pii/S0957415813002353>.
- [126] Y. Su, B. Duan, C. Zheng, Nonlinear PID control of a six-DOF parallel manipulator, *IEE Proceedings. Control Theory and Applications* 151 (2004) 95–102(7), http://digital-library.theiet.org/content/journals/10.1049/ip-cta_20030967.
- [127] B. Zi, B. Duan, J. Du, H. Bao, Dynamic modeling and active control of a cable-suspended parallel robot, *Mechatronics* 18 (1) (2008) 1–12, <https://doi.org/10.1016/j.mechatronics.2007.09.004>, <http://www.sciencedirect.com/science/article/pii/S0957415807001055>.
- [128] M.A. Mirza, S. Li, L. Jin, Simultaneous learning and control of parallel Stewart platforms with unknown parameters, *Neurocomputing* 266 (2017) 114–122, <https://doi.org/10.1016/j.neucom.2017.05.026>, <http://www.sciencedirect.com/science/article/pii/S0925231217308639>.
- [129] J. Merlet, Force-feedback control of parallel manipulators, in: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, vol. 3, 1988, pp. 1484–1489, <https://doi.org/10.1109/ROBOT.1988.12277>.
- [130] W.-W. Shang, S. Cong, Y. Ge, Adaptive computed torque control for a parallel manipulator with redundant actuation, *Robotica* 30 (3) (2012) 457–466, <https://doi.org/10.1017/S0263574711000762>.
- [131] W. Shang, S. Cong, Robust nonlinear control of a planar 2-DOF parallel manipulator with redundant actuation, *Robotics and Computer-Integrated Manufacturing* 30 (6) (2014) 597–604, <https://doi.org/10.1016/j.rcim.2014.04.004>, <http://www.sciencedirect.com/science/article/pii/S0736584514000301>.
- [132] W.-w. Shang, S. Cong, S.-l. Jiang, Dynamic model based nonlinear tracking control of a planar parallel manipulator, *Nonlinear Dynamics* 60 (4) (2010) 597–606, <https://doi.org/10.1007/s11071-009-9617-6>.
- [133] G. Pagis, N. Bouton, S. Briot, P. Martinet, Enlarging parallel robot workspace through type-2 singularity crossing, *Control Engineering Practice* 39 (2015) 1–11, <https://doi.org/10.1016/j.conengprac.2015.01.009>, <http://www.sciencedirect.com/science/article/pii/S0967066115000271>.
- [134] T. Buschmann, V. Favot, M. Schwienbacher, A. Ewald, H. Ulbrich, Dynamics and control of the biped robot LOLA, in: H. Gatringer, J. Gerstmayr (Eds.), *Multi-body System Dynamics, Robotics and Control*, Springer Vienna, Vienna, 2013, pp. 161–173.
- [135] P. Vonwirth, Modular control architecture for bipedal walking on a single compliant leg, Master's thesis, Robotics Research Lab, University of Kaiserslautern, unpublished; supervised by Steffen Schütz, September 2017.

This page intentionally left blank

PART 2

Geometric analysis

This page intentionally left blank

CHAPTER 3

Methods for geometric analysis of parallel mechanisms

Shivesh Kumar^a and Andreas Müller^b

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

This chapter builds the fabric of the Geometric Analysis part of the book. It lays down the fundamental problems that one comes across in the geometric analyses of the robotic mechanisms (Section 3.1). It then introduces two modern approaches, namely screw theory (Section 3.3) and algebraic geometry (Section 3.2), and describes how they lead to the local and global analyses of these systems. Further, the advantages and disadvantages of these methods are highlighted. Lastly, a simple example of a slider crank mechanism is provided in Section 3.4 and these approaches are applied in its study. The content presented here is based on [18].

3.1 Problem description

A robot is mechanically constructed using a set of links and joints. These joints constrain the kinematic motion between these links. The robot might be subjected to additional constraints from its environment or high-level task specification. Thus, to analyze a robot, one must study how different rigid bodies behave under a set of kinematic constraints.

Perhaps, one of the most fundamental questions one may ask is, where is the robot? The answer is *completely* given by the robot's configuration, which is defined in the following.

Definition 3.1 (Configuration and c-space). The configuration \mathbf{q} of a robot is complete specification of position of every point on the robot. The number of real-valued coordinates n required to represent the configuration is the degrees of freedom of a robot. The space containing all the possible configurations of the robot is called configuration space (c-space) denoted by \mathcal{Q} [1].

Since, the configuration space contains the full description of the mechanism, understanding its shape can provide deep insights into the geometric behavior of a robot. The shape of the c -space is described using the c -space topology, which is a fundamental property of space itself and is independent of the choice of coordinates in the space. For example, the c -space of a single DOF prismatic joint is isomorphic to \mathbb{R}^1 . Similarly, the c -space of a single DOF revolute joint is a unit circle \mathbb{S}^1 as the self-connectedness of the rotational movement cannot be captured by \mathbb{R}^1 . The c -space of a serial robotic system can usually be obtained by taking a cartesian product of c -spaces of individual joints. For example, the c -spaces of a rotating sliding knob and a 2R planar robot arm are a cylinder $\mathbb{R}^1 \times \mathbb{S}^1$ and a two-dimensional torus $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$ respectively. The c -space of a n DOF robot with n_r revolute joints and n_p prismatic joints can be expressed as $\mathbb{V}^n = \mathbb{T}^{n_r} \times \mathbb{R}^{n_p}$. The configuration of a parallel robot is admissible if and only if it satisfies the geometric loop closure constraints. Let us define an implicit loop closure function $\phi : \mathbb{V}^n \mapsto \mathbb{R}^r \mid \phi(\mathbf{q}) = \mathbf{0}$ where r is the number of independent loop closure constraints acting on the system. Then the configuration space of the parallel robot, being the set of all admissible configurations, is

$$\begin{aligned} V &:= \{\mathbf{q} \in \mathbb{V}^n \mid \phi(\mathbf{q}) = \mathbf{0}\} \\ V &= \phi^{-1}(\mathbf{0}) \end{aligned} \quad (3.1)$$

The c -space V is a real variety in \mathbb{V}^n and locally (close to a regular configuration \mathbf{q}) a smooth manifold [2]. V comprises several connected smooth manifolds (subspaces like smooth curves or surfaces) that are separated by singular points, indicating *non-smoothness* of V at these points. The mobility of a parallel robot hence depends on the c -space topology. The local DOF of the robot is given by the local dimension of the variety $\dim_{\mathbf{q}} V$. The general mobility of a parallel robot can be estimated by Kutzbach–Grübler’s criteria [3]:

$$d_s(\mathcal{M}) = s(n - m - 1) + f = s(-c) + f, \quad (3.2)$$

where

- s – Motion parameter (= 3 for planar and spherical mechanisms, = 6 for spatial mechanisms)
- n – Number of links in the mechanism;
- m – Total number of joints;
- c – Number of independent closed loops;
- f – Sum of DOF of each joint.

Practically, one is often interested in only knowing the position of the robot's end-effector and whether it is able to perform the required task. The output motion produced by a robot's end-effector can be described in subspace of $SE(3)$ and can be parameterized by appropriate choice of coordinates. For example, the end effector configuration of a spatial 6R serial manipulator can be described by a homogeneous transformation matrix $\mathbf{P}_E \in SE(3)$. The output mapping $f_O : \mathbb{V}^n \mapsto SE(3)$ yields the end effector configuration $\mathbf{P}_E = f_O(\mathbf{q})$ as a function of joint space configuration \mathbf{q} .

Definition 3.2 (Workspace). The workspace \mathcal{W} is a set of all the configurations that the end-effector can reach.

$$\mathcal{W} := \{f_O(\mathbf{q}) \forall \mathbf{q} \in V\} \subset SE(3). \quad (3.3)$$

The workspace is usually determined by the robot's structure and chosen end-effector, but independently of the task.

Definition 3.3 (Task space). The task space $\mathcal{T} \subset SE(3)$ is a space in which the robot's task can be naturally expressed. The task space is defined by the nature of the task independently of the robot.

The robot is called task-redundant if $\dim \mathcal{T} < \dim \mathcal{W}$ and $\mathcal{T} \subseteq \mathcal{W}$, and task-deficient if $\mathcal{W} \subset \mathcal{T}$. It must be noted that the task space, the robot's workspace and its c-space are different from each other. A point in the task space might not be feasible in the workspace of the robot. When feasible in the workspace, it may correspond to more than one robot configuration, meaning that the point is not a complete specification of robot's configuration.

The robot's motion is determined by the motion of its actuators – the mechanical input. The relation between the actuator input and robot's motion is expressed by an input mapping $f_I : \mathbb{V}^n \mapsto \mathcal{U}$ that assigns any feasible robot configuration to the admissible input. This relation may not be unique as there may be different inputs corresponding to the same configuration of the robot. If there are p actuators in the robot, \mathcal{U} is p -dimensional.

Definition 3.4 (Actuation space). The actuation space \mathcal{U} is the set of all admissible actuator configurations in a robotic system. The actuation space is also dependent on robot's structure.

$$\mathcal{U} := \{f_I(\mathbf{q}) \forall \mathbf{q} \in V\} \subset \mathbb{V}^n \quad (3.4)$$

For a fixed base serial mechanism, the actuation space is the same as configuration space, i.e., $p = m = n$. However, in a parallel architecture, the

actuation space is only a subspace of \mathbb{V}^n , i.e., $p < n$. The robot is said to be fully actuated when $p = m$, redundantly actuated when $p > m$.

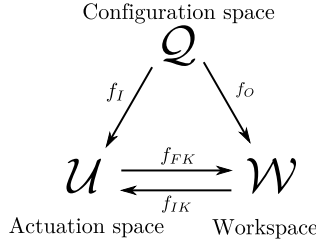


Figure 3.1 Scheme of different mappings and spaces in robotics [18].

Fig. 3.1 shows the different mappings and spaces that we commonly encounter in robotics. It is clear that c-space is the central object geometrically representing the robot. The input and output mappings yields the input, i.e., actuator configuration and output, i.e., end effector's configuration, respectively, corresponding to a given robot configuration. These mappings are not one-to-one for parallel robot in general and for redundant parallel mechanisms in particular. The notion of different spaces, input and output mappings above provides us a good abstract understanding of robot's motion. However, in practice, it is easier to work with direct mappings between the input and output spaces hiding the complete internal state of the robot. In the following, we introduce the notion of direct and inverse kinematic mappings.

Definition 3.5 (Forward Kinematic Mapping). The forward kinematic mapping yields the end-effector configuration $\mathbf{P}_E \in \mathcal{W}$ from the actuator configuration $\mathbf{u} \in \mathcal{U}$ of the robot.

$$\mathbf{P}_E = f_{FK}(\mathbf{u}) \quad (3.5)$$

Generally, it is very straight-forward to solve this problem for a serial robot, and the mapping is one-to-one in nature. For parallel architectures, this problem is very difficult to solve and often many solutions to the forward kinematic problem exist corresponding to an actuator configuration.

Definition 3.6 (Inverse Kinematic Mapping). The inverse kinematic mapping yields the actuator configuration $\mathbf{u} \in \mathcal{U}$ from the end-effector configuration $\mathbf{P}_E \in \mathcal{W}$ of the robot.

$$\mathbf{u} = f_{IK}(\mathbf{P}_E) \quad (3.6)$$

On the contrary, it is easy to solve the inverse kinematic problem for a parallel robot and difficult to solve it in case of serial architectures.

There are essentially two approaches that differ in the way the kinematics are modeled: one uses joint angles and displacements, and the other uses an algebraic parameterization of the motion of the links, such as dual quaternions or Study parameters. In the first case, the c-space V is an *analytic* variety, and in the latter it is an *algebraic* variety, which are defined as the following:

- **Analytic Variety** is defined locally as the set of common zeros of finitely many analytic functions. An analytic function is a function that is locally given by a convergent power series.
- **Algebraic Variety** is defined as the set of solutions of a system of polynomial equations. A polynomial is an expression consisting of variables and coefficients that involves only the operations addition, subtraction, multiplication, and non-negative integer exponents of variables.

In the next two sections, we will discuss the general tools for algebraic (Section 3.2) and analytic (Section 3.3) formulations and an example demonstrating the use of the two methods is provided later in Section 3.4.

3.2 Algebraic geometry

The configuration space of a large class of mechanisms can be defined using polynomial equations. At the core of it, this is possible due to two main reasons: 1) rigid body transformations are algebraic, 2) all lower pair joints except for helical joint are algebraic in nature [4]. An introduction to algebraic geometry can be found in [5]. Two important tools for formulating algebraic constraint equations for any mechanism are 1) Study's kinematic mapping and its variants, 2) tangent half-angle substitution, which are described in the following.

3.2.1 Study's kinematic mapping

Study's Kinematic Mapping maps every displacement in $SE(3)$ to a point in a 7-dimensional projective space \mathbb{P}^7 [6,7]. If the homogeneous coordinate vector of \mathbf{x} is $[x_0 : x_1 : x_2 : x_3 : \gamma_0 : \gamma_1 : \gamma_2 : \gamma_3]^T$, the kinematic pre-image of \mathbf{x} is the displacement $\mathbf{T} \in SE(3)$ described by the transformation matrix:

$$\frac{1}{\Delta} \begin{bmatrix} x_0^2 + x_1^2 - x_2^2 - x_3^2 & -2x_0x_3 + 2x_1x_2 & 2x_0x_2 + 2x_1x_3 & p \\ 2x_0x_3 + 2x_1x_2 & x_0^2 - x_1^2 + x_2^2 - x_3^2 & -2x_0x_1 + 2x_3x_2 & q \\ -2x_0x_2 + 2x_1x_3 & 2x_0x_1 + 2x_3x_2 & x_0^2 - x_1^2 - x_2^2 + x_3^2 & r \\ 0 & 0 & 0 & \Delta \end{bmatrix}$$

$$\begin{aligned}\Delta &= x_0^2 + x_1^2 + x_2^2 + x_3^2 \\ p &= -2x_0y_1 + 2x_1y_0 - 2x_2y_3 + 2x_3y_2 \\ q &= -2x_0y_2 + 2x_1y_3 + 2x_2y_0 - 2x_3y_1 \\ r &= -2x_0y_3 - 2x_1y_2 + 2x_2y_1 + 2x_3y_0.\end{aligned}$$

The parameters x_i , y_i , $i \in \{0, \dots, 3\}$ are called as the *Study's parameters*. An Euclidean transformation can be represented by a point $p \in \mathbb{P}^7$ if and only if the following equations are satisfied:

$$g_1 := x_0y_0 + x_1y_1 + x_2y_2 + x_3y_3 = 0 \quad (3.7)$$

$$g_2 := x_0^2 + x_1^2 + x_2^2 + x_3^2 - 1 = 0 \quad (3.8)$$

All the points that satisfy Eq. (3.7) belong to the 6-dimensional *Study quadric*, S_6^2 . Eq. (3.8) ensures that the points do not lie on the *exceptional generator*, $x_0 = x_1 = x_2 = x_3 = 0$. The points on S_6^2 are called *kinematic image points* of the corresponding displacement, and the seven-dimensional projective space is called *kinematic image space*. Its variants for describing rigid body displacement in the planar motion group $SE(2)$ (also known as Blaschke mapping [8]), 3-dimensional rotational motion group $SO(3)$ also exist. Rigid body displacement in $SO(2)$ can be described using complex numbers.

3.2.2 Tangent half-angle substitution

While setting up the constraint equations, one typically encounters trigonometric terms, typically due to the motion of revolute joints, which can be made algebraic using the tangent half-angle substitutions or Weierstrass substitution. For any point $(\cos \theta, \sin \theta)$ on the unit circle \mathbb{S}^1 , draw a line passing through it and the point $(-1, 0)$. This point crosses the y -axis at some point $y = t$. Using simple geometry, it is trivial to show that $t = \tan(\theta/2)$ (see Fig. 3.2). The equation for the drawn line is $y = (1 + x)t$. The equation for the intersection of the line and circle is then a quadratic equation involving t . The two solutions to this equation are $(-1, 0)$ and $(\cos \theta, \sin \theta)$. This allows us to write the latter as rational functions of t , as shown below.

$$\sin(\theta) = \frac{2t}{1+t^2}, \quad \cos(\theta) = \frac{1-t^2}{1+t^2}, \quad \tan(\theta) = \frac{2t}{1-t^2} \quad (3.9)$$

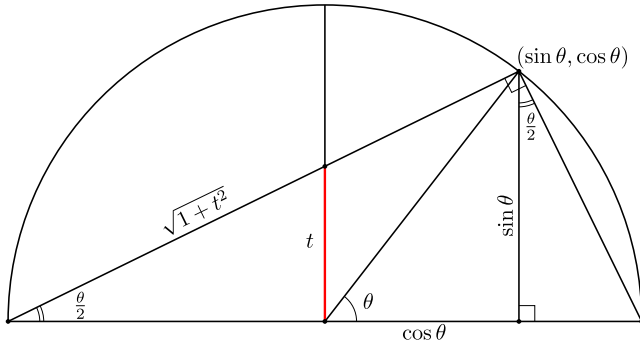


Figure 3.2 Geometric proof of tangent half-angle substitution [18].

3.2.3 Towards global kinematics

Analytic descriptions of kinematic chains lead to parametric and implicit representations. These are easy to set up but difficult to solve. Very often only a single numerical solution is obtained. Complete analysis and synthesis needs all solutions. An algebraic description of constraint equations allows the use of powerful methods and algorithms from algebraic geometry. An important first task is to find the simplest algebraic constraint equations that describe the chains. There exists always a best-adapted coordinate system for a mechanism or at least for one kinematic chain in a more complicated mechanism. When a kinematic chain is represented in its “best”-adapted coordinate system, then it is called canonical chain. Geometric and algebraic preprocessing is needed before elimination, Gröbner basis computation or numerical solution process starts. Algebraic constraint equations yield answers to the overall behavior of a kinematic chain, which provides insights into *global kinematics* of the mechanism [9]. Solutions to the inverse kinematics of a general 6R serial chain robot [10] and to the direct kinematics of the general Stewart–Gough platform [11], which yields polynomials of degree 16 and degree 40, respectively, are major advances in the last century. The disadvantage of this approach is that such an analysis is not always straightforward and the involved algorithms are NP-hard.

3.3 Screw theory and Lie group methods

Screw theory is the algebra and calculus of pairs of vectors, such as forces and moments and angular and linear velocity, that arise in the kinematics and dynamics of rigid bodies. Expressing the motion of a rigid body as a

combination of a rotation and a translation about a line was first proposed by Chasles (1830) and further developed by Poincot (1848). Then, Julius Plücker came up with a way to assign six homogeneous coordinates for a line [12]. In 1876, Sir Robert Stawell Ball developed a mathematical framework of screw theory for applications in rigid body mechanics [13]. K.H. Hunt [14] further developed screw theory with a geometrical emphasis. Using line geometry, the major contribution of Hunt was to classify the various screw systems. Screw theory has proven to be a powerful mathematical tool for the local analysis of complex mechanisms and robots. It provides a quick and efficient way to describe the configuration of a system at any given instant. There are two core reasons behind it: 1) the rigid body transformations can be analytically described in screw coordinates using exponential mapping, 2) most kinematic joints can be described as a combination of 1-DOF screw joints.

3.3.1 Fundamentals of screw theory

Theorem 3.1 (Chasles' Theorem). *The most general motion of a rigid body consists of a rotation about a line in space together with a translation along it. Such a quantity is called **twist** or **spatial velocity**.*

$$\mathbf{V} = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} \in \mathbb{R}^6. \quad (3.10)$$

Theorem 3.2 (Poincot's Theorem). *The most general force that can act on a rigid body consists of a linear force acting along a line in space, together with a moment acting about it. Such a quantity is called a **wrench** or **spatial force**.*

$$\mathbf{W} = \begin{bmatrix} \mathbf{m} \\ \mathbf{f} \end{bmatrix} \in \mathbb{R}^6. \quad (3.11)$$

Definition 3.7 (Plücker coordinates of a line). The Plücker coordinates of a line \mathcal{L} defined by two points in 3-D Euclidean space are given by the unit direction vector between those points $\hat{\mathbf{s}}$ and a moment vector $\mathbf{s}_o \times \hat{\mathbf{s}}$ (see Fig. 3.3a for a visualization).

$$\mathcal{L} = \begin{bmatrix} \hat{\mathbf{s}} \\ \mathbf{s}_o \times \hat{\mathbf{s}} \end{bmatrix} \quad (3.12)$$

Definition 3.8 (Screw). A screw \mathbf{S} is defined by a unit direction axis $\hat{\mathbf{s}}$, the position of a point \mathbf{s}_o on this axis with respect to a reference frame and

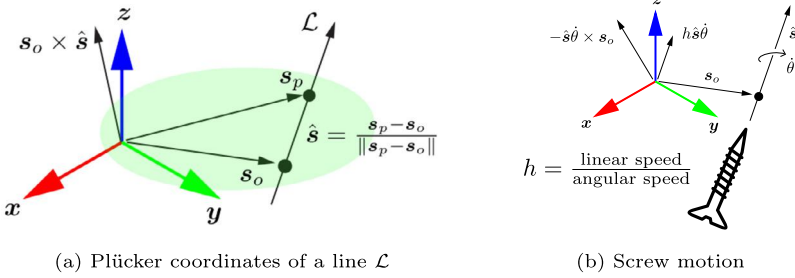


Figure 3.3 Screw representation with Plücker coordinates of a line along the screw axis and the pitch, Fig. 3.3b is adapted from [1].

pitch h (see Fig. 3.3b).

$$\mathbf{S} = \begin{bmatrix} \hat{\mathbf{s}} \\ \mathbf{s}_o \times \hat{\mathbf{s}} + h\hat{\mathbf{s}} \end{bmatrix} \in \mathbb{R}^6 \quad (3.13)$$

It is to be noted that setting $h = 0$, the vector $\begin{bmatrix} \hat{\mathbf{s}} \\ \mathbf{s}_o \times \hat{\mathbf{s}} \end{bmatrix}$ are the Plücker coordinates of a line along the screw axis. Geometrically, a screw is determined by the Plücker coordinates of a line along the screw axis and a pitch (see Fig. 3.3). In classical screw theory literature, it is often denoted as \$.

Once, we have defined the notion of a screw \mathbf{V} , a twist can be interpreted in terms of this screw and a velocity $\dot{\theta}$ about it. The expression for the twist is given by $\mathbf{V} = \mathbf{S}\dot{\theta}$.

$$\begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{s}} \\ \mathbf{s}_o \times \hat{\mathbf{s}} + h\hat{\mathbf{s}} \end{bmatrix} \dot{\theta} = \begin{bmatrix} \hat{\mathbf{s}}\dot{\theta} \\ -\hat{\mathbf{s}}\dot{\theta} \times \mathbf{s}_o + h\hat{\mathbf{s}}\dot{\theta} \end{bmatrix} \quad (3.14)$$

3.3.2 Matrix exponential and matrix logarithm maps

Any rigid-body configuration can be achieved by starting from a fixed reference frame and integrating a twist for a specified time. Such a motion resembles the motion of a screw, rotating about and translating along the same fixed screw axis. The observation that all the configurations can be achieved a screw motion motivates a six parameter representation of the configuration called the *exponential coordinates* [1].

Definition 3.9 (Matrix Exponential [1]). Let $\mathbf{S} = (\boldsymbol{\omega}, \mathbf{v})$ denote the screw coordinates. The matrix exponential is defined as $\exp : [\mathbf{S}]\theta \in se(3) \rightarrow \mathbf{T} \in$

$SE(3)$. If $\|\omega\| = 1$, then for any distance $\theta \in \mathbb{R}$ traveled along the screw axis or any angle $\theta \in \mathbb{R}$ rotated about the screw axis,

$$\mathbf{T} = \exp[\mathbf{S}]\theta = \begin{bmatrix} \exp[\omega]\theta & (\mathbf{I}\theta + (1 - \cos\theta)[\omega] + (\theta - \sin\theta)[\omega]^2)\mathbf{v} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3), \quad (3.15)$$

where

$$\exp[\omega]\theta = \mathbf{I} + \sin\theta[\omega] + (1 - \cos\theta)[\omega]^2 \in SO(3). \quad (3.16)$$

If $\|\omega\| = 0$ and $\|\mathbf{v}\| = 1$, then

$$\mathbf{T} = \exp[\mathbf{S}]\theta = \begin{bmatrix} \mathbf{I} & \mathbf{v}\theta \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3). \quad (3.17)$$

On the contrary, given an arbitrary pose $(\mathbf{R}, \mathbf{p}) \in SE(3)$, one can always find a screw axis $\mathbf{S} = (\omega, \mathbf{v})$ and a scalar θ representing it. The matrix $[\mathbf{S}]\theta$ is called as the *matrix logarithm* of the pose (\mathbf{R}, \mathbf{p}) .

Definition 3.10 (Matrix Logarithm [1]). The matrix logarithm is defined as $\log : \mathbf{T} \in SE(3) \rightarrow [\mathbf{S}]\theta \in se(3)$. If

$$\mathbf{T} = \exp[\mathbf{S}]\theta = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}, \quad (3.18)$$

then the matrix

$$[\mathbf{S}]\theta = \begin{bmatrix} [\omega]\theta & \mathbf{v}\theta \\ \mathbf{0} & 0 \end{bmatrix} \quad (3.19)$$

is the matrix logarithm of $\mathbf{T} = (\mathbf{R}, \mathbf{p})$.

The algorithm to compute the matrix logarithm can be found in [1] and is skipped here for brevity. The exponential mapping is a key ingredient in the product of exponentials (POE) formula [15], which can be used to compute the forward kinematics of serial kinematic chains or to set up geometric loop closure constraint equations for parallel mechanisms. The exponential mapping is analytic, i.e., the function is given by a convergent power series. The variety thus obtained from loop closure constraints determined using POE formula is also analytic in nature. Another advantage here is that the use of the exponential mapping in terms of screw coordinates gives rise to explicit closed-form formulae for its derivatives.

3.3.3 Screw representation of joint motion

Lower kinematic pairs with one DOF allow the interconnected bodies to perform screw motions between each other with a certain pitch h . Revolute joints can be modeled with zero pitch screws $h = 0$ and prismatic joints are modeled with infinite pitch screws $h = \infty$. If \mathbf{S} denotes the unit screw axis, then for different 1 DOF lower kinematic pairs, \mathbf{S} is given by:

$$\mathbf{S}^{\text{revolute}} = \begin{bmatrix} \hat{\mathbf{s}} \\ \mathbf{0} \end{bmatrix} \quad \mathbf{S}^{\text{screw}} = \begin{bmatrix} \hat{\mathbf{s}} \\ h\hat{\mathbf{s}} \end{bmatrix} \quad \mathbf{S}^{\text{prismatic}} = \begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{s}} \end{bmatrix}, \quad (3.20)$$

where $\hat{\mathbf{s}}$ denotes the unit vector along the joint axis resolved in the joint frame.

3.3.4 Towards local analysis

One of the biggest advantages of using methods from screw theory is that it allows an easy set up of implicit constraint equations. Further, the derivatives of these equations can be derived in closed form. If a feasible configuration of the mechanism V_q is known, an exhaustive local analysis of the c-space geometry as well as finite curves passing through this point can be performed. It provides a powerful setting for studying the mechanism behavior and classification of its singularities [16]. A disadvantage of this approach is that the mobility and singularity analysis requires a prior knowledge of the actual solution set V . This disadvantage can be leveraged by an algebraic parameterization of the constraint equations after geometric preprocessing and the use of powerful tools from computational algebraic geometry to solve them.

3.4 Example: Lambda mechanism

A Lambda mechanism is basically a planar mechanism with triangular geometry, as shown in Fig. 3.5. Body B1 forms a one-link arm, while B2 and B3 are the cylinder and piston, respectively, of a linear actuator. Joints 1, 2, and 3 participate in the kinematic loop, with joint 3 being the actuated prismatic joint. This section presents the study of this 1-RRPR mechanism, which has been used for the abstraction of a revolute joint in various robot designs (see Fig. 3.4 for its applications in robots at DFKI-RIC).

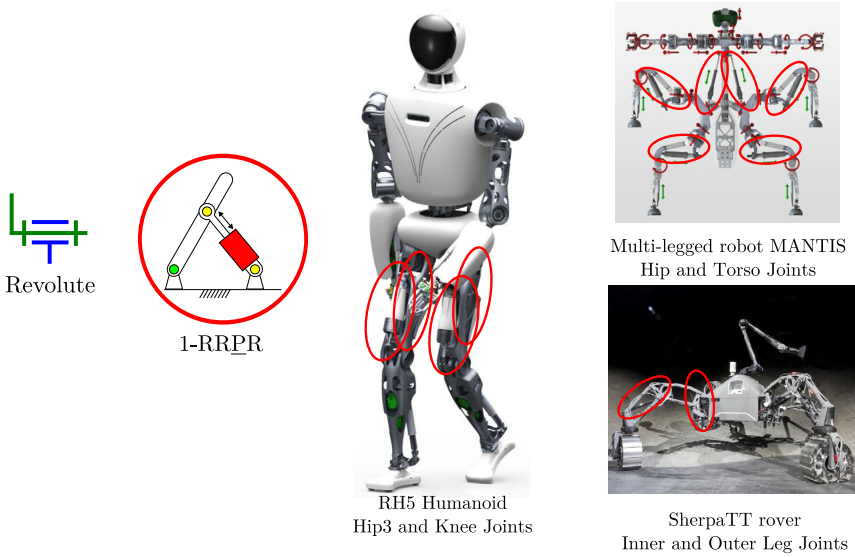


Figure 3.4 Revolute joint abstractions with Lambda mechanism in robots at DFKI-RIC.

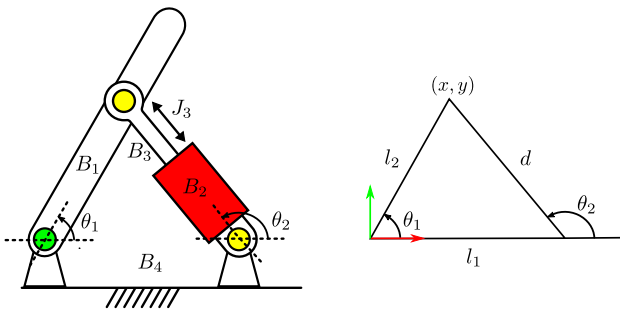


Figure 3.5 Lambda mechanism and its triangular geometry (adapted from [17]).

3.4.1 Mobility analysis

The general mobility of this mechanism can be calculated with the help of Kutzbach–Grübler criteria, see Eq. (3.2). Since it is a planar mechanism, $s = 3$. Hence, the mobility of this mechanism is $d_s(\mathcal{M}) = 3(4 - 4 - 1) + 4 = 1$.

3.4.2 Geometric analysis

In the following, we will perform the geometric analysis of this mechanism using both the approaches described previously in this chapter. First, we develop an analytic formulation using joint angles and displacements, and then develop an algebraic formulation using the polynomial description of the geometric constraint equations. We set the link parameters as $l_1 = l_2 = 1$ and hence the corresponding geometry is that of an equilateral triangle.

3.4.2.1 Analytic formulation

The displacement of joints J_1 and J_2 can be parameterized using angles θ_1 and θ_2 and the movement of the slider can be parameterized with linear displacement variable d . These are measured as absolute coordinates in the reference frame defined in Fig. 3.5. Hence, the c-space of this mechanism can be described using the choice of coordinates (θ_1, θ_2, d) and is an analytic variety in $V^n = \mathbb{T}^2 \times \mathbb{R}^1$. Using the law of cosine in trigonometry, one can establish the constraint equation for this mechanism:

$$d^2 = l_1^2 + l_2^2 - 2l_1l_2 \cos \theta_1. \quad (3.21)$$

To calculate the passive joint angle (θ_2), one can use the formula below:

$$d \cos \theta_2 = l_2 \cos \theta_1 - l_1. \quad (3.22)$$

Eq. (3.21) and Eq. (3.22) are analytic surfaces and their intersection is an analytic variety, as shown in Fig. 3.6a. However, it is to be noted that d should always be greater than or equal to 0 and hence, the part of the curve where d is negative should be disregarded. The final analytic variety of this mechanism is shown in Fig. 3.7a. It can be noticed that when $\theta_1 = 0$, then $d = 0$ and θ_2 is undefined and hence, represents a c-space singularity.

From an input-output viewpoint, d is the input variable and θ_1 is the output variable. It immediately follows that Eq. (3.21) also provides an explicit closed-form solution to the inverse geometric (or kinematic) problem. Rearranging Eq. (3.21), a closed form solution to the direct geometric (or kinematic) problem can be derived. There are two solutions to the forward geometric problem, as shown in Eq. (3.23).

$$\theta_1 = \arccos \frac{l_1^2 + l_2^2 - d^2}{2l_1l_2} \quad (3.23)$$

$$\theta_1 = \text{atan2}(\pm \sin \theta_1, \cos \theta_1)$$

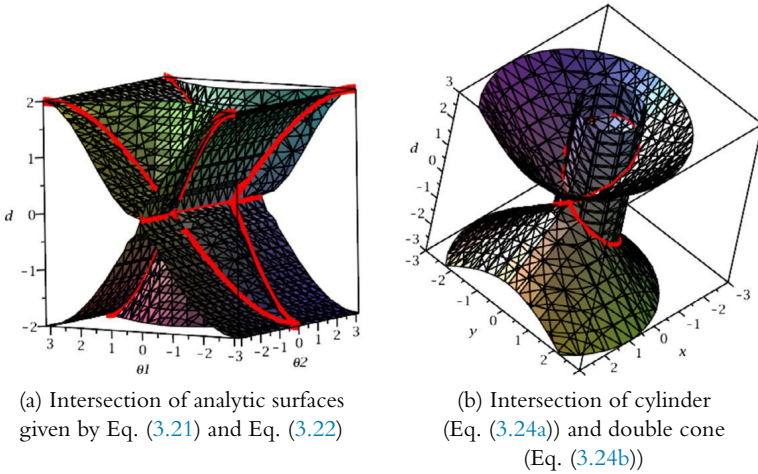


Figure 3.6 C-space of lambda mechanism [18].

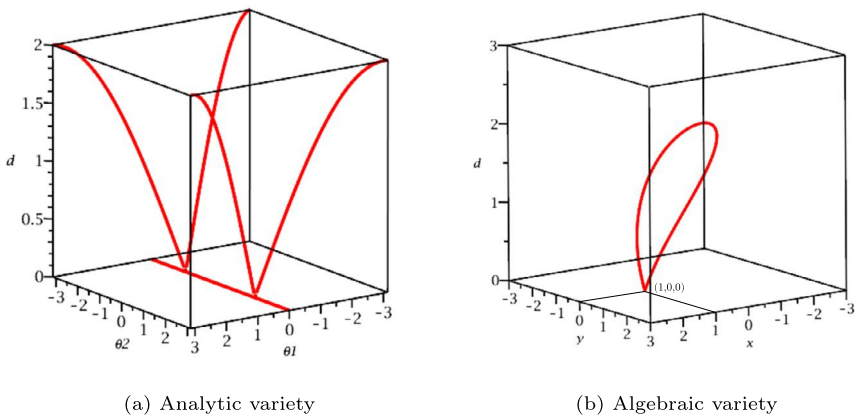


Figure 3.7 C-space of lambda mechanism with $d \geq 0$ [18].

3.4.2.2 Algebraic formulation

Since, it is a planar mechanism, one could formulate the constraint equations in the form of a set of polynomials by choosing an algebraic parametrization of the links in the XY plane. To this end, let $(x = l_2 \cos \theta_1, l_2 \sin \theta_1)$ denote the coordinates of the crank link B_1 and $(l_1, 0)$ denote the coordinates of the point on the ground where cylinder B_2 is at-

tached. The loop constraint equations are given as the following:

$$x^2 + y^2 = l_2^2, \quad (3.24a)$$

$$(x - l_1)^2 + y^2 = d^2. \quad (3.24b)$$

In this case, the c -space coordinates are $(x, y, d) \in \mathbb{R}^3$ and any feasible values of these coordinates satisfying Eq. (3.24) fully describe the mechanism. The first constraint equation represents the surface of a cylinder (Eq. (3.24a)) while the second equation represents a double cone (Eq. (3.24b)). The c -space is in this case is the set of all points lying on the intersection of these two surfaces, which looks like a bent infinity shaped curve (or lemniscate curve) in 3-space of c -space coordinates. However, it should be noted that $d \geq 0$, so the part of the curve where d is negative should be disregarded. The algebraic variety representing the c -space of the mechanism is shown in Fig. 3.7b. A cusp can be noticed at the point $(x, y, d) = (1, 0, 0)$, which shows the singular configuration of the c -space as V is not a smooth manifold at this point.

From an input-output viewpoint, d is the input variable and the pair (x, y) is the output variable. In the algebraic formulation, the solution to the inverse kinematics problem can be derived by substituting Eq. (3.24a) in Eq. (3.24b) and eliminating the variable y . The solution is unique and is given by:

$$\begin{aligned} (x - l_1)^2 + l_2^2 - x^2 &= d^2 \\ d &= \sqrt{l_1^2 + l_2^2 - 2l_1x}. \end{aligned} \quad (3.25)$$

Similarly, the forward geometric problem can be solved by manipulating the constraint equations to write the variables (x, y) as a function of d . As noted earlier, it can be noted that this problem has two solutions given algebraically by Eq. (3.26).

$$\begin{aligned} x(d) &= \frac{l_1^2 + l_2^2 - d^2}{2l_1} \\ y(d) &= \pm \sqrt{l_2^2 - \left(\frac{l_1^2 + l_2^2 - d^2}{2l_1} \right)^2} \end{aligned} \quad (3.26)$$

The above equation is also a parametric equation describing the c -space variety in \mathbb{R}^3 , as shown in Fig. 3.7b.

3.5 Conclusion

This chapter presents a summary of modern geometric approaches in the analysis of robots and mechanisms. Two approaches, namely screw theory and algebraic geometry, are briefly discussed and their corresponding advantages in the local and global kinematic analysis of the mechanisms are highlighted. Lastly, a simple one DOF planar mechanism which converts the linear motion of an actuator to the rotary motion is studied with both analytic and algebraic approaches. It can be observed that for simple cases, like that of the lambda mechanism, the two approaches are equivalent in terms of their ease of use and the insights they provide in the mechanism analysis. In the upcoming chapters, where more complex mechanisms have been studied, we will take the geometric approach which suits better to the particular geometry or type of the mechanism being studied.

References

- [1] K.M. Lynch, F.C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st edition, Cambridge University Press, New York, NY, USA, 2017.
- [2] A. Mueller, On the terminology and geometric aspects of redundant parallel manipulators, *Robotica* 31 (1) (2013) 137–147, <https://doi.org/10.1017/S0263574712000173>.
- [3] G. Gogu, Chebychev–Grübler–Kutzbach’s criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations, *European Journal of Mechanics. A, Solids* 24 (3) (2005) 427–441, <https://doi.org/10.1016/j.euromechsol.2004.12.003>, <https://www.sciencedirect.com/science/article/pii/S0997753805000033>.
- [4] C.W. Wampler, A.J. Sommese, Applying numerical algebraic geometry to kinematics, in: J.M. McCarthy (Ed.), *21st Century Kinematics*, Springer London, London, 2013, pp. 125–159.
- [5] D.A. Cox, J. Little, D. O’Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, 3rd ed., Undergraduate Texts in Mathematics, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [6] M.L. Husty, H.-P. Schröcker, Kinematics and algebraic geometry, in: *21st Century Kinematics*, Springer-Verlag London, London, 2013, pp. 85–123.
- [7] M.L. Husty, M. Pffurner, H.-P. Schröcker, K. Brunthaler, Algebraic methods in mechanism analysis and synthesis, *Robotica* 25 (6) (2007) 661–675.
- [8] O. Bottema, B. Roth, *Theoretical Kinematics*, Dover Books on Physics, Dover Publications, 1990, <https://books.google.de/books?id=AMIIAwAAQBAJ>.
- [9] M.L. Husty, Analysis of parallel manipulators using algebraic tools, *Lecture Notes, Singularity Summer School*, September 2017.
- [10] M. Raghavan, B. Roth, Inverse kinematics of the general 6R manipulator and related linkages, *Journal of Mechanical Design* 115 (3) (1993) 502–508.
- [11] M. Husty, An algorithm for solving the direct kinematics of general Stewart-Gough platforms, *Mechanism and Machine Theory* 31 (4) (1996) 365–379, [https://doi.org/10.1016/0094-114X\(95\)00091-C](https://doi.org/10.1016/0094-114X(95)00091-C), <http://www.sciencedirect.com/science/article/pii/0094114X9500091C>.
- [12] J. Plücker, XVII. On a new geometry of space, *Philosophical Transactions of the Royal Society of London* 155 (1865) 725–791, <https://doi.org/10.1098/rstl.1865>.

- 0017, <https://royalsocietypublishing.org/doi/pdf/10.1098/rstl.1865.0017>, <https://royalsocietypublishing.org/doi/abs/10.1098/rstl.1865.0017>.
- [13] R.S. Ball, The theory of screws: a study in the dynamics of a rigid body, *Mathematische Annalen* 9 (4) (1876) 541–553, <https://doi.org/10.1007/BF01442479>.
 - [14] J. Davidson, K. Hunt, *Robots and Screw Theory: Applications of Kinematics and Statics to Robotics*, Oxford University Press, 2004.
 - [15] R.W. Brockett, Robotic manipulators and the product of exponentials formula, in: P.A. Fuhrmann (Ed.), *Mathematical Theory of Networks and Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1984, pp. 120–129.
 - [16] A. Müller, *Local Investigation of Mobility and Singularities of Linkages*, Springer International Publishing, Cham, 2019, pp. 181–229, https://doi.org/10.1007/978-3-030-05219-5_5.
 - [17] R. Featherstone, *Rigid Body Dynamics Algorithm*, Springer, 2008.
 - [18] S. Kumar, *Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots*, Ph.D. thesis, Universität Bremen, 2019.

This page intentionally left blank

CHAPTER 4

2-DOF orientational parallel mechanisms

Shivesh Kumar^a, Christoph Stoeffler^a, Heiner Peters^a,
Andreas Müller^b, and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

^cWorking Group Robotics, University of Bremen, Bremen, Germany

This chapter presents three different abstractions of a universal joint, where traditionally the variant $2SPU + 1U$ is used. We discuss a novel design of type $2SPRR + 1U$ and show that the aforementioned variant can be seen as special case of this novel type. Moreover, the $2SPU + 2RSU + 1U$ mechanism is presented, which was devised to overcome workspace restrictions of the traditional $2SPU + 1U$ design. These mechanisms have been used in various robot designs (see Fig. 4.1), e.g., to abstract hip, ankle, and wrist joints in humanoid robots. The chapter is organized as follows: Section 4.1 presents the motivation for the mechanism's design and highlights its novelty. Section 4.2 presents the manipulator's architecture and constraint equations. Section 4.3 presents the solutions to the direct and inverse kinematic problems by utilizing traditional vector calculus, but also tools from computational algebraic geometry. Section 4.4 presents the workspace characterization, description of its singularity curves, and performance analysis and Section 4.5 concludes this chapter. The content of this chapter is based on [1], [2] and [3].

4.1 Introduction

Orientation parallel mechanisms with 2 degrees of freedom are extremely important for the design of anthropomorphic or animal-inspired robots. These are required for developing wrist, ankle, torso joints in the robots and serve as an abstraction to 2-DOF universal joints. Since they are often integrated in the distal links, these parallel mechanisms should be light-weight in construction and should provide good force-velocity characteristics. Fig. 4.2 shows the novel two degrees of freedom orientational

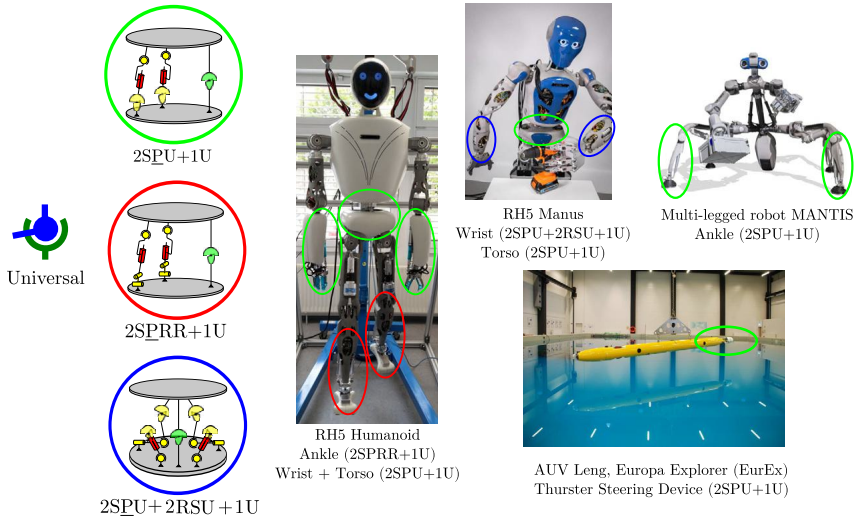


Figure 4.1 Universal joint abstractions in robots at DFKI-RIC [1].

parallel mechanism of type 2SPRR+1U which is used as an ankle joint in the RH5 humanoid robot [4] developed at DFKI-RIC. The kinematic actuation principle of this mechanism comprises of a motion constraint generator leg with a universal joint (U) and two auxiliary actuation legs of type SPRR, i.e., they contain a spherical (S), prismatic (P), and two revolute (RR) joints in series, as shown in Fig. 4.3. It is well known that during walking, the torque required for the pitch movement is larger than the torque required for the roll movements [5]. When the two motors are actuated in the same direction, the mechanism produces a pitch only movement demonstrating good torque transmission characteristics. It has been shown in biomechanics studies that during the ankle pitch movement of human gait, a peak torque between 105 Nm and 120 Nm is required when flexion/extension angle is between -6° and -12° [6]. To reflect this in the ankle design, the foot attachment points of the two linear actuators may be displaced along the z-axis by 30 mm. Utilizing a common universal joint at the offset points, as in the case of $2SPU+1U$ mechanism, reduces the workspace of the roll movement. Instead, two skew revolute joints, with axes parallel to the axes of universal joint on constraint generator leg, connected by an intermediate offset link are used to provide the desired torque characteristics in the pitch movement with minimal influences on the motion range of roll movement.



Figure 4.2 CAD prototype of Ankle joint [1].

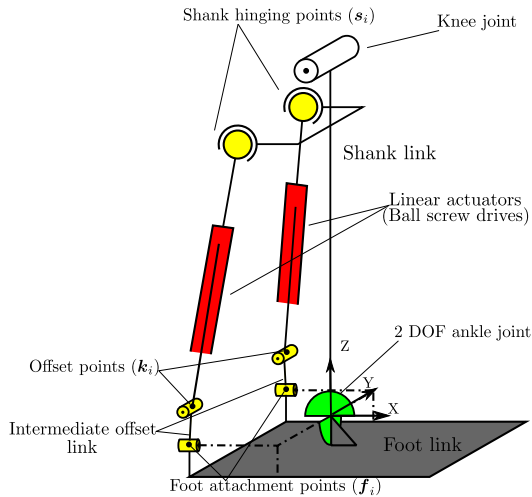
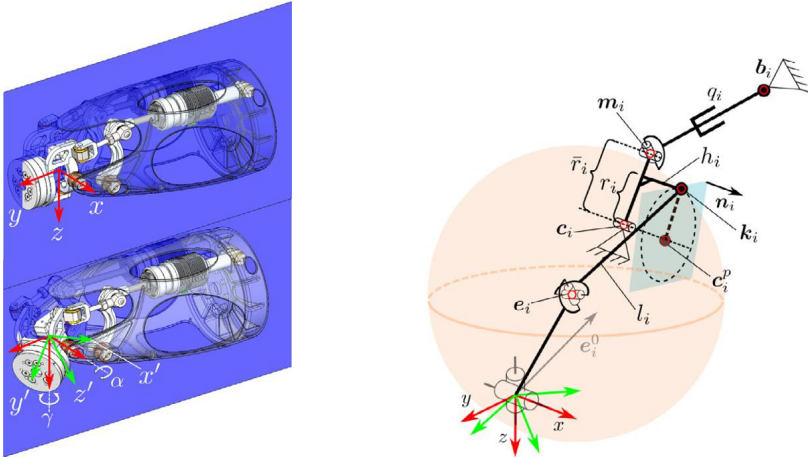


Figure 4.3 Scheme of the mechanism. Original Source: [2] (Reproduced with permission from Springer Nature).

A common problem for PKMs is their reduced workspace size that often arises due to an increased number of constraints and a higher number of possible collisions. Avoiding collisions in the $2SPU + 1U$ mechanism can be achieved by adding intermediate linkages that transmit the prismatic actuator forces to the universal joint and extending the workspace of the overall mechanism. This results into the $2SPU + 2RSU + 1U$ variant that is depicted in Fig. 4.4a and is implemented as wrist mechanism in

RH5 Manus humanoid [7]. The intermediate linkages also shift occurring singularities to higher pitch angles and make this mechanism suitable for humanoid wrist applications, bringing it closer to human motion ranges. However, increasing the *flexion* and *extension* movement of the wrist mechanism comes at the cost of reduced tilting capabilities.



(a) Depiction of the $2SPU + 2RSU + 1U$ mechanism as humanoid wrist mechanism with its symmetry plane

(b) Schematics with geometric parameters of one side of the wrist mechanism.

Figure 4.4 CAD model of a humanoid wrist and geometric overview. Original Source: [3] (Reproduced with permission from Springer Nature).

4.2 Architecture and constraint equations

In this section, we will elaborate on the constraint equations defining the three different architectures $2SPU + 1U$, $2SPRR + 1U$ and $2SPU + 2RSU + 1U$, which we also denote *torso*, *ankle*, and *wrist* mechanism, respectively. This is in accordance with their application as joint abstractions in humanoid robots. Since the $2SPU + 1U$ topology arises as a special case of the $2SPRR + 1U$ mechanism, only the latter is treated in this presentment. However, in practice the equations for the $2SPU + 1U$ case can be derived independently in a simpler manner.

4.2.1 Ankle ($2SPRR + 1U$) and torso ($2SPU + 1U$) mechanism

The mobility of a mechanism (\mathcal{M}) can be calculated with the help of Kutzbach–Grübler criteria as follows: $d_s(\mathcal{M}) = s(n - m - 1) + f = s(-c) + f$,

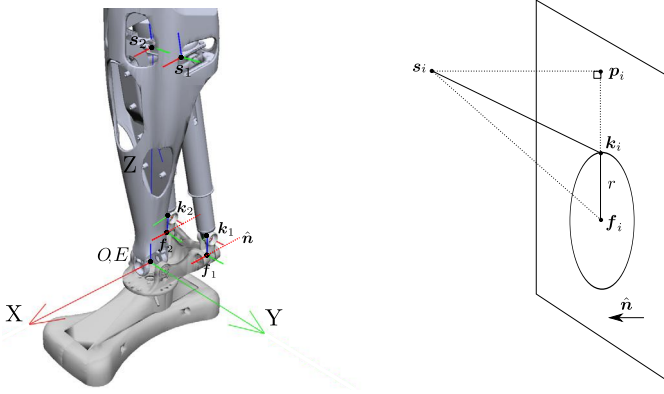


Figure 4.5 Geometry of the ankle mechanism $2SPRR + 1U$ (RGB colors denote XYZ axes) [2]. Original Source: [2] (Reproduced with permission from Springer Nature).

where n is number of links in the mechanism $3 + 3 + 2 = 8$, m is total number of joints $4 + 4 + 1 = 9$, f is total dof of joints $2 + 6 + 6 = 14$, and s is the motion parameter. Since, it is a spatial mechanism, $s = 6$. Hence, the mobility can be calculated as $d_s(\mathcal{M}) = 6(8 - 9 - 1) + 14 = 2$.

The manipulator architecture and geometry is shown in Fig. 4.5. Let us define a set of three points: shank point (s_i), foot attachment point (f_i) and the offset point (k_i) on the two auxiliary actuation legs of the mechanism. The base frame O is attached to the shank link and is coincident with the end-effector (EE) frame E attached to the foot link in zero configuration. The intermediate offset link $f_i k_i$ rotates about the x-axis (denoted as \hat{n}) of the frame defined at f_i , thus point k_i moves on a circle of radius r , which is equal to the length of the link, $\|f_i - k_i\|$. The length of the linear actuators (d_i) is the norm of the vector ($k_i - s_i$). We also define a vector $\delta_i := (s_i - f_i)$.

The constraint equations of the manipulator are the following:

$$d_i^2 = \|k_i - s_i\|^2 = \|p_i - s_i\|^2 + \|k_i - p_i\|^2, \quad i \in \{1, 2\}. \quad (4.1)$$

We can rewrite Eq. (4.1) purely as a function of (s_i, \hat{n}, f_i) .

$$\begin{aligned} d_i^2 &= \|\hat{n} \cdot \delta_i\|^2 + (\|f_i - p_i\| - r)^2 \\ d_i^2 &= \|\hat{n} \cdot \delta_i\|^2 + (\sqrt{\|\delta_i\|^2 - \|\hat{n} \cdot \delta_i\|^2} - r)^2 \\ d_i^2 &= \|\hat{n} \cdot \delta_i\|^2 + (\|\hat{n} \times \delta_i\| - r)^2. \end{aligned} \quad (4.2)$$

For the purpose of visualization or computing passive joint angles, it is necessary to compute the point \mathbf{k}_i , which is given by Eq. (4.3).

$$\mathbf{k}_i = \mathbf{f}_i + r \frac{\boldsymbol{\delta}_i - (\hat{\mathbf{n}} \cdot \boldsymbol{\delta}_i) \hat{\mathbf{n}}}{\|\boldsymbol{\delta}_i - (\hat{\mathbf{n}} \cdot \boldsymbol{\delta}_i) \hat{\mathbf{n}}\|}. \quad (4.3)$$

The orientation of the moving platform is parameterized by roll (θ , around X axis) and pitch (ϕ , around Y axis) angles such that ${}^O\mathbf{R}_E = \text{Rot}(X, \theta) \cdot \text{Rot}(Y, \phi)$. The revolute joint axis vector ($\hat{\mathbf{n}}$) and the foot attachment point (\mathbf{f}_i) are expressed in global coordinate frame using $\hat{\mathbf{n}} = {}^O\mathbf{R}_E \cdot \hat{\mathbf{n}}_E$ and $\mathbf{f}_i = {}^O\mathbf{R}_E \cdot \mathbf{f}_i^E$ respectively, where $\hat{\mathbf{n}}_E$ and \mathbf{f}_i^E denote the revolute joint axis and foot attachment point vector in EE frame.

4.2.2 Wrist mechanism (2SPU + 2RSU + 1U)

Like for the ankle and torso mechanism the Kutzbach–Grübler criteria is used to compute the mobility of the design that possess 10 bodies and 13 joints. With the joints having 26 degrees of freedom in total, such that $d_s(\mathcal{M}) = 6(10 - 13 - 1) + 26 = 2$. For the discussion to follow, these two degrees of freedom are called *inclination* (α) and *tilt* (γ). We will consider the base frame of the mechanism at the universal joint of the end-effector and describe joint locations by Cartesian vectors, as to be seen in Fig. 4.4b. The point \mathbf{k}_i at the center of the spherical joint of the intermediate crank gives rise to the constraint equations involving actuator length and end-effector configuration.

$$\left\| \mathbf{c}_i + \frac{\bar{r}_i}{r_i} (\mathbf{k}_i - \mathbf{c}_i^p) - \mathbf{b}_i \right\|^2 - q_i^2 = 0, \quad (4.4)$$

$$h_i := \|\mathbf{R}(\alpha, \gamma) \mathbf{e}_i^0 - \mathbf{k}_i\|^2 - l_i^2 = 0, \quad (4.5)$$

with $i \in \{1, 2\}$ for both sides of the mechanism. Inclination α and tilt γ parameterize the rotation matrix $\mathbf{R}(\alpha, \gamma)$ that can either be *intrinsic* or *extrinsic* thus $\mathbf{R}_z(\gamma)\mathbf{R}_x(\alpha)$ or $\mathbf{R}_z(\alpha)\mathbf{R}_x(\gamma)$, respectively.

4.3 Solving forward and inverse kinematics

4.3.1 Ankle (2SPRR + 1U) and torso (2SPU + 1U) mechanism

Algebraic geometry techniques have proven to be useful in solving the forward kinematics of parallel manipulators, but they require the constraint equations to be algebraic. Tangent half angle substitutions might leave the

Table 4.1 Geometric dimensions (in mm) of the ankle mechanism. Original Source: [2] (Reproduced with permission from Springer Nature).

i	s_i	f_i^E	\hat{n}_E	$\ f_i k_i\ $
1	$(-22.30, 25, 291.27)^T$	$(-70, 40, 0)^T$	$(1, 0, 0)^T$	30
2	$(-22.30, -25, 291.27)^T$	$(-70, -40, 0)^T$	$(1, 0, 0)^T$	30

constraint equations undefined for π orientations. Hence, in order to have an algebraic description of the mechanism's constraint equations, cosines and sines are replaced by $\cos(\theta) = x$, $\sin(\theta) = y$, $\cos(\phi) = u$, and $\sin(\phi) = v$ in ${}^O\mathbf{R}_E$, but at the cost of adding two more equations to the ideal set. To this end, rearranging Eq. (4.2) and squaring to avoid the square root term $\|\hat{n} \times \delta_i\|$ leads to four algebraic constraint equations:

$$g_1 := (d_1^2 - \|\hat{n} \cdot \delta_1\|^2 - \|\hat{n} \times \delta_1\|^2 - r^2)^2 - 4 \|\hat{n} \times \delta_1\|^2 r^2 = 0, \quad (4.6a)$$

$$g_2 := (d_2^2 - \|\hat{n} \cdot \delta_2\|^2 - \|\hat{n} \times \delta_2\|^2 - r^2)^2 - 4 \|\hat{n} \times \delta_2\|^2 r^2 = 0, \quad (4.6b)$$

$$g_3 := x^2 + y^2 - 1 = 0, \quad (4.6c)$$

$$g_4 := u^2 + v^2 - 1 = 0. \quad (4.6d)$$

After substituting the geometric dimensions provided in Table 4.1, the constraint equations are only a function of variables x , y , u , v , d_1 , and d_2 . g_1 and g_2 are 16 degree polynomials and are quite long to be shown here.

The solution to inverse kinematics problem (IKP) of the manipulator is straightforward and unique for a given orientation of the moving platform, as the joint variables d_i can be easily calculated from Eq. (4.2). This can be used to implement the analytical loop closure function for the complete mechanism in HyRoDyn software framework [8]. It is noteworthy that when the roll angle is zero, Eq. (4.2) yields $d_1 = d_2$.

The direct kinematics problem (DKP) aims to find the variables x , y , u , and v when the prismatic joint lengths are specified. In search of maximum number of solutions to DKP (assembly modes), an ideal of the constraint polynomials g_i is defined: $\mathcal{I} = \langle g_1, g_2, g_3, g_4 \rangle \mid \mathcal{I} \subseteq k[u, v, x, y]$. Finding the Gröbner basis with a pure lexicographic ordering of the orientation parameters in any order leads to a univariate polynomial of degree 32. Since squaring two of the four constraint equations quadruples the number of solutions, the number of solutions must be quartered. Hence, the upper limit to DKP solutions of the manipulator under study is eight. To investigate the number of real solutions, **RootFinding[Isolate]** function of Maple[®] is used. The algorithm behind this function finds out the rational univariate

representation of the set of polynomials and isolates the real roots of these univariate polynomials based on Descartes' rule of sign and the bisection strategy in a unified framework. The variables d_1 and d_2 are varied from 221 to 331 mm (physical motion range of linear actuators) with an increment of 6 mm and the percentage of the number of real DKP solutions is listed in Table 4.2. It is evident that the maximum number of real solutions for the considered set of prismatic joint lengths is six. Fig. 4.6 shows six such assembly modes when $d_1 = 221$ mm and $d_2 = 228.3$ mm. It is speculated that a different choice of design parameters may lead to eight real solutions to DKP. In the physical construction of the ankle joint, passive joint limits are chosen such that there exists a unique solution to forward kinematics for a given input of actuator lengths in their feasible motion range (for instance, Fig. 4.6e).

Table 4.2 Percentage of real solutions to direct kinematics. Original Source: [2] (Reproduced with permission from Springer Nature).

Number of solutions	0	2	4	6	8
Number of poses (/2601)	124	268	2146	63	0
Percentage	4.77	10.30	82.51	2.42	0

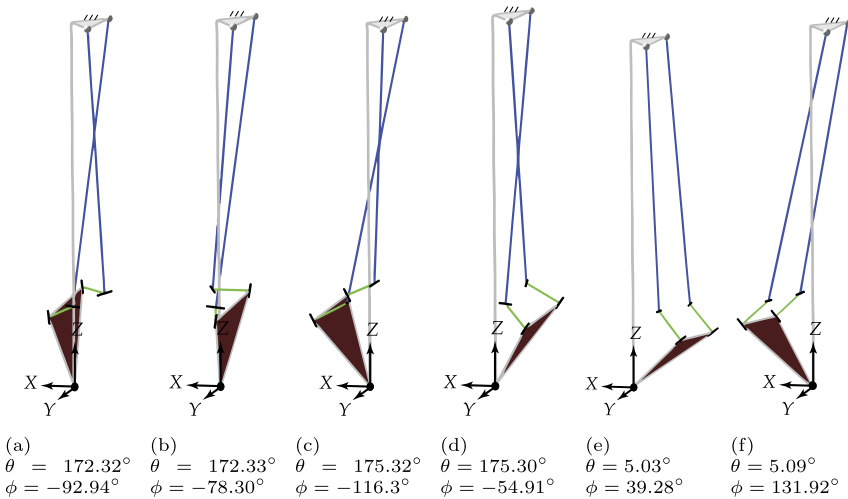


Figure 4.6 Assembly modes for $d_1 = 221$ mm and $d_2 = 228.3$ mm. Original Source: [2] (Reproduced with permission from Springer Nature).

4.3.2 Wrist mechanism ($2SPU + 2RSU + 1U$)

Solving the circle–sphere intersection is crucial to the problem of solving forward and inverse kinematics of the $2SPU + 2RSU + 1U$ variant. It generally writes $\mathcal{C}(\mathbf{c}_i^p, \mathbf{n}_i, r_i) \cap \mathcal{S}(\mathbf{s}_i, t_i) \rightarrow \mathbf{i}_i^+, \mathbf{i}_i^-$. The problem be simplified by projecting it to a circle–circle intersection in the intersecting plane, making use of

$$\boldsymbol{\eta}_i = ((\mathbf{c}_i^p - \mathbf{s}_i) \cdot \mathbf{n}_i) \mathbf{n}_i. \quad (4.7)$$

The circle radius of the intersected sphere then becomes $\rho_i^2 = t_i^2 - \boldsymbol{\eta}_i \cdot \boldsymbol{\eta}_i$ and the distance between both circle centers then writes $\boldsymbol{\delta}_i = \mathbf{s}_i + \boldsymbol{\eta}_i - \mathbf{c}_i^p$. The vector pointing from the circle \mathcal{C} to the midline of intersection is

$$\begin{aligned} \boldsymbol{\delta}'_i &= \frac{\boldsymbol{\delta}_i \cdot \boldsymbol{\delta}_i + r_i^2 - \rho_i^2}{2\|\boldsymbol{\delta}_i\|} \frac{\boldsymbol{\delta}_i}{\|\boldsymbol{\delta}_i\|} \\ &= \frac{\boldsymbol{\delta}_i}{2} + \boldsymbol{\delta}_i \frac{r_i^2 - \rho_i^2}{2\boldsymbol{\delta}_i \cdot \boldsymbol{\delta}_i} \\ &= \frac{\boldsymbol{\delta}_i}{2} + \boldsymbol{\delta}_i \frac{r_i^2 - t_i^2 + \boldsymbol{\eta}_i \cdot \boldsymbol{\eta}_i}{2\boldsymbol{\delta}_i \cdot \boldsymbol{\delta}_i}. \end{aligned} \quad (4.8)$$

Additionally, one can compute the orthogonal vector to $\boldsymbol{\delta}'_i$ that points from the midline center to the intersection points

$$\mathbf{p}_i = \frac{\boldsymbol{\delta}'_i}{\|\boldsymbol{\delta}'_i\|} \times \mathbf{n}_i \sqrt{r_i^2 - \boldsymbol{\delta}'_i \cdot \boldsymbol{\delta}'_i}. \quad (4.9)$$

This yields the intersection points with

$$\mathbf{i}_i = \mathbf{c}_i^p + \boldsymbol{\delta}'_i \pm \mathbf{p}_i. \quad (4.10)$$

Solving the forward kinematics for the wrist joint also makes use of algebraic geometry approaches that can be applied upon knowing \mathbf{k}_i (see Fig. 4.4b). Applying the circle–sphere intersection on the wrist mechanism, allows to obtain \mathbf{m}_i , such that $\mathbf{m}_i^+, \mathbf{m}_i^- \leftarrow \mathcal{C}(\mathbf{c}_i^p, \mathbf{n}_i, r_i) \cap \mathcal{S}(\mathbf{b}_i, q_i)$, which can be used to obtain \mathbf{k}_i for given actuator lengths q_i as

$$\mathbf{k}_i = \mathbf{c}_i + \frac{r_i}{\tilde{r}_i} (\mathbf{m}_i - \mathbf{c}_i) + h_i \mathbf{n}_i. \quad (4.11)$$

To obtain the configuration of the wrist mechanism from \mathbf{k}_i , it remains to solve a sphere–sphere intersection. It can be solved by means of *Gröbner bases* of an ideal $\mathcal{I} = \langle h_1, h_2, h_3, h_4 \rangle | \mathcal{I} \subseteq k[t, u, v, w]$ The first two equations

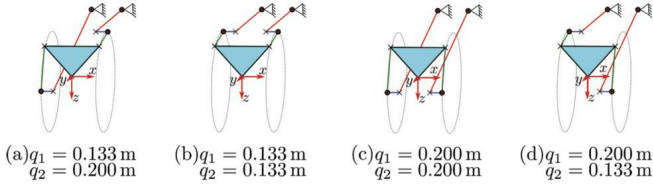


Figure 4.7 Four inverse solutions of the zero configuration: $\alpha = 0$, $\gamma = 0$. Constraints arising from the revolute joints located in c_i are indicated by dashed circles. Original Source: [3] (Reproduced with permission from Springer Nature).

arise from Eq. (4.5) and the latter two from substitution in the rotation matrices, like already done in Section 4.3.1:

$$h_3 := t^2 + u^2 - 1 = 0 \quad \text{where} \quad t = \cos(\alpha), \quad u = \sin(\alpha), \quad (4.12)$$

$$h_4 := v^2 + w^2 - 1 = 0 \quad \text{where} \quad v = \cos(\gamma), \quad w = \sin(\gamma). \quad (4.13)$$

In contrast, computing the inverse kinematic problem, i.e., actuator lengths from end-effector configuration, can be done by computing $\mathbf{k}_i^+, \mathbf{k}_i^- \leftarrow \mathcal{C}(c_i^p, \mathbf{n}_i, r_i) \cap \mathcal{S}(\mathbf{R}(\alpha, \gamma) \mathbf{e}_i^0, l_i)$ and inserting the result into Eq. (4.4). Rearranging this quadratic constraint equation for q_i doubles the solution of \mathbf{k}_i what leaves us with a maximum number of four inverse solutions. With the wrist geometry specified by the parameters:

$$\begin{aligned} \mathbf{b}_1 &= [15 \quad -178 \quad -34]^T, & \mathbf{b}_2 &= [-15 \quad -178 \quad -34]^T, \\ \mathbf{c}_1 &= [15 \quad -32 \quad 11]^T, & \mathbf{c}_2 &= [-15 \quad -32 \quad 11]^T, \\ \mathbf{e}_1^0 &= [27 \quad 0 \quad -30]^T, & \mathbf{e}_2^0 &= [-27 \quad 0 \quad -30]^T, \\ \mathbf{n}_1 &= [1 \quad 0 \quad 0]^T, & \mathbf{n}_2 &= [-1 \quad 0 \quad 0]^T, \\ \bar{r}_i &= 49, & r_i &= 49, & h_i &= 12, & l_i &= 45 \end{aligned}$$

and inside the feasible actuation space $q_i \in [113, 178]$ mm eight real forward solutions and four inverse solutions (Fig. 4.7) can be computed.¹

4.4 Workspace, singularity, and performance analysis

For all three joint designs, the *constraint Jacobian* of the mechanism is used to discuss the velocity and force transmissions from joint to task space and vice

¹ Software available at <https://github.com/dfki-ric/NovelWrist/>.

versa. The constraint equation (see Section 4.2) of the form $\mathbf{g}(\mathbf{x}, \mathbf{q}) = \mathbf{0}$ can be differentiated

$$\frac{\partial \mathbf{g}(\mathbf{x}, \mathbf{q})}{\partial \mathbf{x}} \dot{\mathbf{x}} = -\frac{\partial \mathbf{g}(\mathbf{x}, \mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} \quad (4.14)$$

$$\begin{aligned} \dot{\mathbf{x}} &= -\mathbf{J}_x^{-1} \mathbf{J}_q \dot{\mathbf{q}} \\ &= \mathbf{J} \dot{\mathbf{q}}. \end{aligned} \quad (4.15)$$

By conservation of power for joint and task space, one can also obtain the force transmission of a parallel mechanism with

$$\mathbf{f} = \mathbf{J}^{-T} \boldsymbol{\tau}, \quad (4.16)$$

where $\boldsymbol{\tau}$ are the joint forces and \mathbf{f} the end-effector forces, respectively.

The quality of velocity or force transmission of a parallel manipulator can be measured by plotting the inverse of condition number of the kinematic Jacobian matrix (\mathbf{J}) over the manipulator's workspace. The inverse of condition number of the Jacobian is calculated with $c(\mathbf{J}) = \frac{1}{\|\mathbf{J}\| \|\mathbf{J}^{-1}\|}$, where $\|\cdot\|$ represents the Euclidean norm of the matrix.

4.4.1 Ankle (2SPRR + 1U) and torso (2SPU + 1U) mechanism

To demonstrate the suitability of the novel 2-SPRR+1U mechanism as a humanoid ankle joint, it is important to compute and characterize its feasible workspace in orientation and configuration domains. The feasible configuration space is calculated by varying the orientation variables describing foot rotation, roll (θ) and pitch (ϕ) angles, in the range $[-\pi, \pi]$. Then the physical limits of the linear actuators ($d_i \in [221, 331]$ mm) are imposed to compute the workspace of the mechanism under actuator constraints. The resulting configuration space and orientation workspace are shown in Fig. 4.8. It is possible to take into account physical limits of passive joints in the mechanism to further compute the physically realizable workspace, which is indicated with a closed curve in the figure. The final range of motion (ROM) of the proposed ankle mechanism is more than that of an average human and is presented in Table 4.3 (compare with [6]). Hence, the available range of motion (ROM) in the humanoid ankle is between -57° and 57° for the roll angle (ϕ) and between -51.5° and 45° for the pitch angle (θ).

The ankle mechanism under study does not have any limb singularities, since the auxiliary actuation legs do not generate any constraints on the

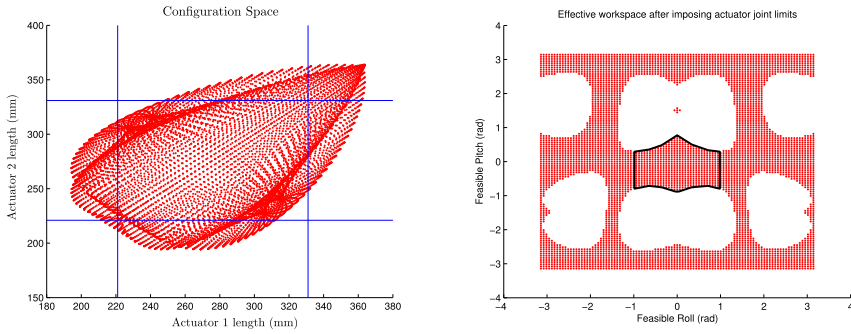


Figure 4.8 Configuration space and orientation workspace under actuator physical limits. Original Source: [2] (Reproduced with permission from Springer Nature).

moving platform. Nonetheless, the actuation scheme results in the so called actuation singularities, which can be determined through the kinematic Jacobian matrix of the manipulator obtained by the partial differentiation of the constraint polynomials in Eq. (4.6) with respect to the orientation parameters:

$$J = \begin{bmatrix} \frac{\partial g_1}{\partial \theta} & \frac{\partial g_1}{\partial \phi} \\ \frac{\partial g_2}{\partial \theta} & \frac{\partial g_2}{\partial \phi} \end{bmatrix} \tag{4.17}$$

The configurations for which the determinant of the Jacobian matrix J vanishes are called actuation singularities. The determinant of J depends only on θ and ϕ . An implicit plot of the equation $\det(J) = 0$ in terms of the orientation variables θ and ϕ is shown in Fig. 4.9, which shows the singularity curves in the mechanism’s workspace. Also, it can be observed that there exist four singularities each for pure roll ($\phi = 0$) and pure pitch ($\theta = 0$) movements. Fig. 4.11 shows the singular poses for the pure roll and pure pitch movements which are closest to the zero configuration of the mechanism.

The inverse of the condition number is plotted over the feasible orientation workspace of the ankle, as shown in Fig. 4.10. From Fig. 4.10, it is evident that the kinematic Jacobian matrix is well-conditioned in the feasible orientation workspace of the ankle mechanism.

For practical purposes, it is crucial to calculate the maximum absolute velocity and torque available at the EE from the maximum force and ve-

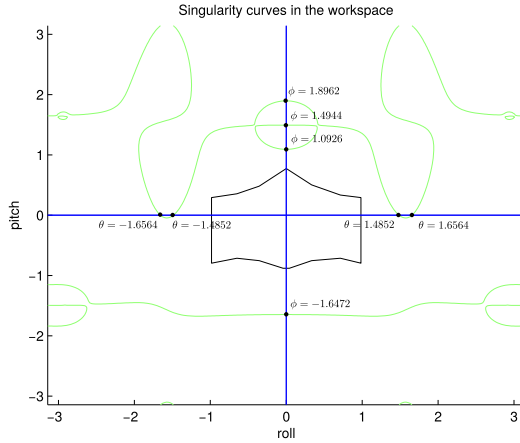


Figure 4.9 Singularity curve. Original Source: [2] (Reproduced with permission from Springer Nature).

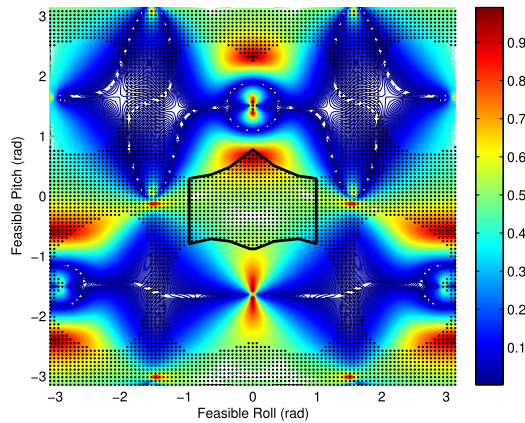


Figure 4.10 Inverse of condition number. Original Source: [2] (Reproduced with permission from Springer Nature).

locity that can be delivered by the actuators. These are computed with the help of kinematic Jacobian matrix and actuator specification (see Table 4.3).

Velocities for pure pitch movement

It is trivial to compute the pure pitch velocity, as we know that for pure pitch movement ($\theta = 0$ and $\dot{\theta} = 0$), the movement required in the linear actuators is identical, i.e., $d_1 = d_2$ and $\dot{d}_1 = \dot{d}_2$. To compute the maximum

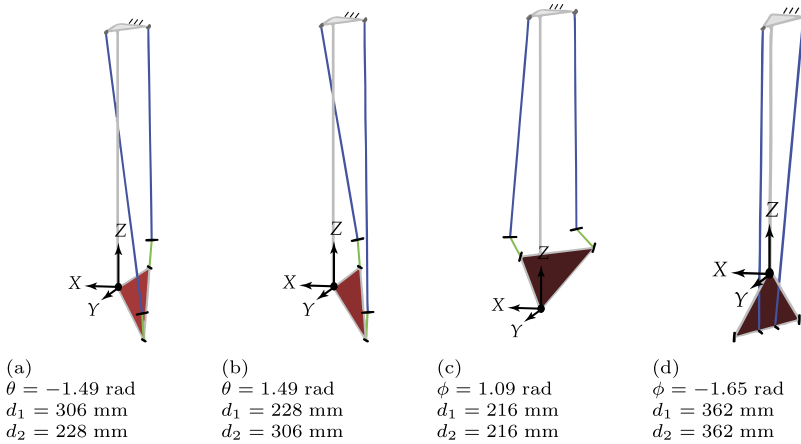


Figure 4.11 Singularity configurations for pure roll (θ) and pure pitch (ϕ) movements. Original Source: [2] (Reproduced with permission from Springer Nature).

Table 4.3 Ankle joint specification (total weight of lower leg = 3.2 kg, weight of one actuator = 0.44 kg). Original Source: [2] (Reproduced with permission from Springer Nature).

Range (min. to max.)	Position	Max. abs. force	Max. abs. velocity
Ankle pitch	-51.5° to 45°	43.8 Nm to 110.1 Nm	61°s^{-1} to 154°s^{-1}
Ankle roll	-57° to 57°	30.6 Nm to 57 Nm	118°s^{-1} to 222°s^{-1}
Linear actuator	221 mm to 331 mm	754 N	81 mm s^{-1}

pitch velocity, we vary the pitch angle ϕ in $[-\pi, \pi]$ in the following formula:

$$\dot{\mathbf{x}}_{max} = \mathbf{J}(0, \phi)\dot{\mathbf{q}}_{max}, \tag{4.18}$$

where $\dot{\mathbf{q}}_{max} = \begin{bmatrix} \dot{d}_{max} & \dot{d}_{max} \end{bmatrix}$. The maximum speed of the linear actuators, i.e., \dot{d}_{max} can be found in Table 4.3. Fig. 4.12 shows the pitch velocity component of $\dot{\mathbf{x}}_{max}$ in complete and feasible working range of the mechanism. The discontinuities in the curve in Fig. 4.12a show the singular points. Obviously, the roll velocity component is zero and hence not shown in

the plots. Fig. 4.12b shows the singularity-free pitch velocity transmission curve in physically realizable workspace of the mechanism.

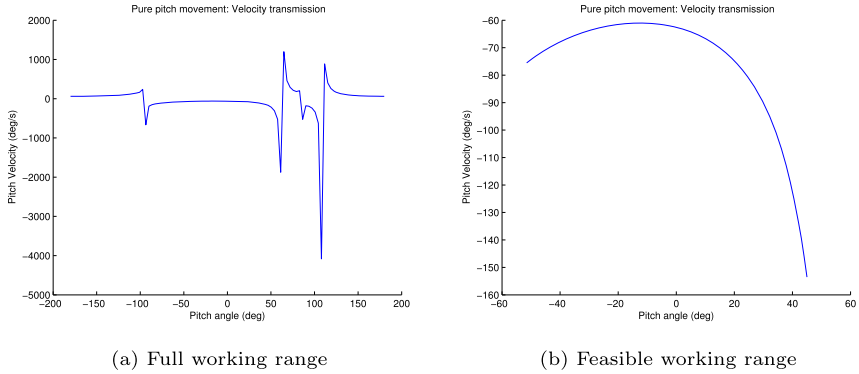


Figure 4.12 Velocity transmission in pure pitch movement [1].

Velocities for pure roll movement

For pure roll movement ($\phi = 0$ and $\dot{\phi} = 0$), such an analysis is not so straightforward because an explicit relation between d_1 and d_2 is not known. However, the ratio of \dot{d}_1 and \dot{d}_2 can be computed with the help of inverse kinematic Jacobian matrix using the fact that $\phi = 0$ for the pure roll movement:

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{J}^{-1}(\theta, 0)\dot{\mathbf{x}} \\ \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ 0 \end{bmatrix} \\ \frac{\dot{d}_1}{\dot{d}_2} &= \frac{a_{11}}{a_{21}} \end{aligned} \quad (4.19)$$

where $a_{11}, a_{12}, a_{21}, a_{22}$ are elements of the inverse kinematic Jacobian and are functions of output variables (θ, ϕ) . In Eq. (4.19), one has the freedom to provide maximum velocity to one of the actuator and calculate the maximum possible speed in the other actuator. For example, if the second actuator is chosen to work at maximum speed, i.e., $\dot{d}_2 = \dot{d}_{max}$, then the first actuator speed can be calculated as $\dot{d}_1 = \frac{a_{11}}{a_{21}}\dot{d}_{max}$. Once the two actuator speeds are known, one can substitute them in Eq. (4.18) to compute the maximum roll velocity. Fig. 4.13 shows the roll and pitch velocity components of $\dot{\mathbf{x}}_{max}$ in complete and feasible working range of the mechanism.

The two set of curves in Fig. 4.13a show which actuator was chosen to operate at the maximum speed. The discontinuities in the roll curves show the four singular points. As one can see, the pitch velocity component is zero for a pure roll movement. Fig. 4.13b shows the maximum roll velocity in the feasible working range of the mechanism.

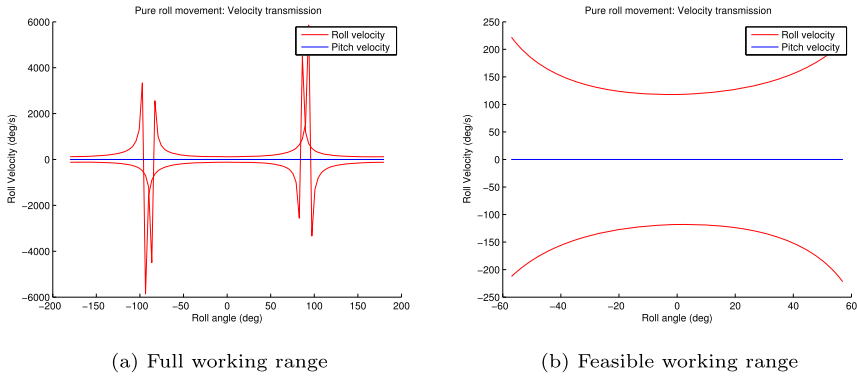


Figure 4.13 Velocity transmission in pure roll movement [1].

In both cases, it can be noticed that the maximum output velocity is not constant and depends on the configuration of the mechanism. The range of the maximum output velocity has been documented in Table 4.3.

Forces for pure pitch movement

It is trivial to compute the pure pitch torque, as we know that for pure pitch movement ($\theta = 0$), the movement required in the linear actuators is identical, i.e., $f_1 = f_2$. To compute the maximum pitch moment, we vary the pitch angle ϕ in $[-\pi, \pi]$ in the following formula:

$$\mathbf{f}_{max} = \mathbf{J}^{-T} \boldsymbol{\tau}_{max}, \quad (4.20)$$

where $\bar{\boldsymbol{\tau}}_{max} = \begin{bmatrix} f_{max} & f_{max} \end{bmatrix}$. Fig. 4.14a and Fig. 4.14b show the pure pitch velocity transmission in complete and feasible working range of the mechanism respectively. The points at which the pure pitch moment becomes zero are the actuation singularities. It can also be noticed that the maximum pitch torque of 110 N m is available when the pitch angle is between -6° and -12° , which is the main motivation during the design process.

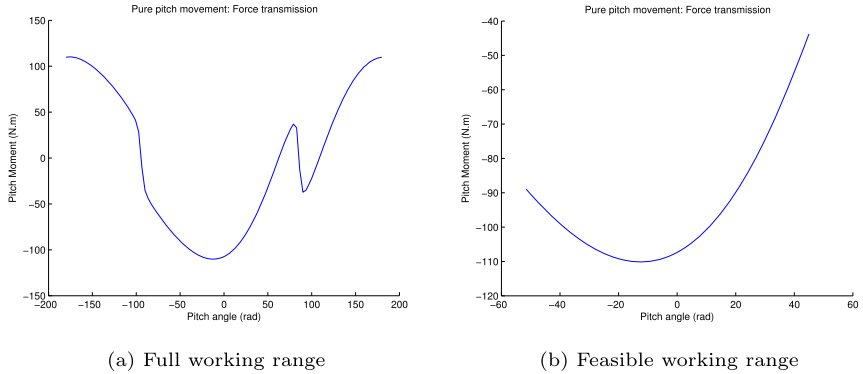


Figure 4.14 Force transmission in pure pitch movement [1].

Forces for pure roll movement

For pure roll movement ($\phi = 0$), again such an analysis is not so straightforward. However, the ratio of f_1 and f_2 can be computed with the help of forward kinematic Jacobian matrix using the fact that $\tau_{pitch} = 0$ for the pure roll movement.

$$\boldsymbol{\tau} = \mathbf{J}^T(\theta, 0)\mathbf{f}$$

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} \tau_{roll} \\ 0 \end{bmatrix} \quad (4.21)$$

$$\frac{f_1}{f_2} = \frac{b_{11}}{b_{21}}$$

In Eq. (4.21), one has the freedom to provide maximum force to one of the actuator and calculate the maximum possible force in the other actuator. For example, if the second actuator is chosen to work at maximum force, i.e., $f_2 = f_{max}$, then the first actuator force can be calculated as $f_1 = \frac{b_{11}}{b_{21}}f_{max}$. Once the two actuator forces are known, one can substitute them in Eq. (4.20) to compute the maximum roll moment. Fig. 4.15 shows the roll and pitch moment components of $\boldsymbol{\tau}_{max}$ in complete and feasible working range of the mechanism. The two set of curves in Fig. 4.15a and Fig. 4.15b show which actuator was chosen to operate at the maximum force. The points where roll moment drops to zero are the singular points. As one can see, the pitch moment component is zero for a pure roll movement. Fig. 4.15b shows the maximum roll moment in the feasible working range of the mechanism.

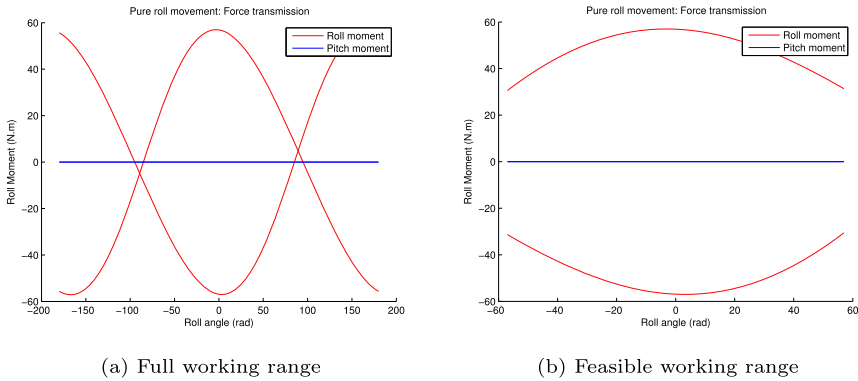


Figure 4.15 Force transmission in pure roll movement [1].

Table 4.4 Geometric dimensions (in mm) of the torso mechanism [1].

i	s_i	f_i^E
1	$(-80.49, 20, -210)^T$	$(-91.22, 76.81, 23)^T$
2	$(-80.49, -20, -210)^T$	$(-91.22, -76.81, 23)^T$

Again, in both cases, it can be noticed that the maximum output force is not constant and depends on the configuration of the mechanism. The range of the maximum output force for pure pitch and pure roll movements has been documented in Table 4.3. The proposed ankle design provides good force and velocity transmission along pure pitch and roll movements which are highly desired in modern humanoids.

A variant of $2SPRR + 1U$ linkage with $r = 0$ has been used to construct the torso joint in RH5 humanoid. The physical dimensions of the $2SPU+1U$ mechanism applied to a torso joint is provided in Table 4.4 based on Fig. 4.16. Fig. 4.17 shows the configuration space and workspace of the torso, taking into account the physical limits of the actuators. Fig. 4.18 shows the singularity curve and inverse of $\text{cond}(\mathbf{J})$ over workspace for $2SPU+1U$ mechanism. Table 4.5 presents the overall joint specification of the torso joint in RH5 humanoid.

4.4.2 Wrist mechanism ($2SPU + 2RSU + 1U$)

By inserting the rearranged form of Eq. (4.4) into Eq. (4.15) yields an expression for the mechanisms Jacobian. An impression of the work space conditioning of the mechanism can be seen in Fig. 4.22a for the assembly

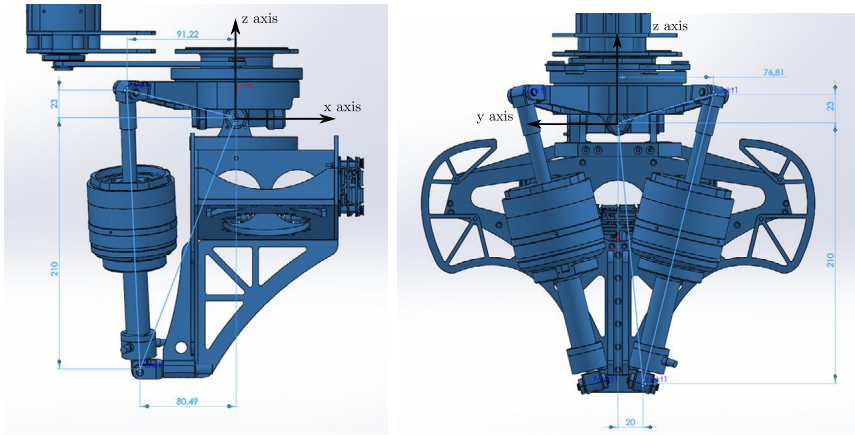


Figure 4.16 Geometric dimensions of RH5 torso joint [1].

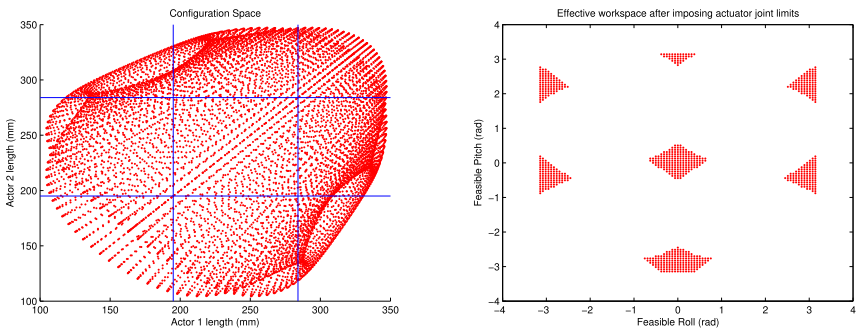


Figure 4.17 Configuration space and orientation workspace for 2SPU+1U mechanism applied to RH5 torso [1].

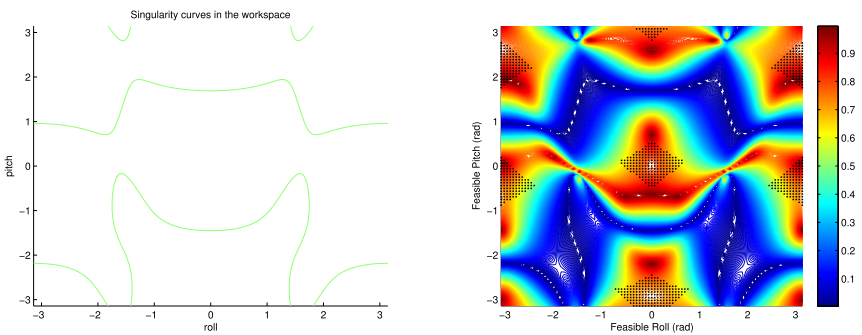
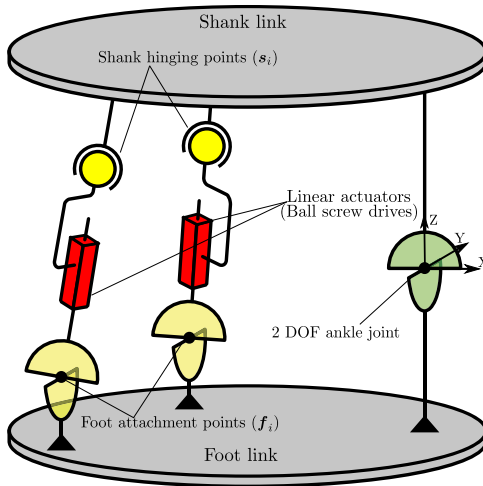


Figure 4.18 Singularity curve and inverse of condition number over workspace for 2SPU+1U mechanism applied to RH5 torso joint [1].

Table 4.5 RH5 torso joint specification [1].

Range (min. to max.)	Position	Max. abs. force	Max. abs. velocity
Torso pitch	-25° to 29°	380 N m to 493 N m	184°s^{-1} to 238°s^{-1}
Torso roll	-36° to 36°	285 N m to 386 N m	208°s^{-1} to 400°s^{-1}
Linear actuator	195 mm to 284 mm	2716 N	291 mm s^{-1}

**Figure 4.19** Manipulator architecture of 2SPU+1U ankle mechanism [1].

mode of the wrist shown in Fig. 4.7. The area of feasible configurations due to actuator limits is highlighted as dashed line. Fig. 4.22a also reveals that there is a wide range of tilt angles that cannot be reached due to an increased number of constraints arising from the intermediate linkages in the mechanism. Moreover, the actuator limits have been placed close to the limits of the mechanism, see Fig. 4.22b.

4.4.3 Comparison between 2SPRR + 1U and 2SPU + 1U designs

2SPU+1U mechanism is a special case of 2SPRR+1U mechanism with intersecting revolute joint axes. The manipulator architecture applied to an ankle design is shown in Fig. 4.19. The analysis presented in this chapter can

Table 4.6 Geometric dimensions (in mm) of the 2SPU+1U ankle mechanism [1].

i	s_i	f_i^E
1	$\begin{pmatrix} -22.30 \\ 25 \\ 291.27 \end{pmatrix}$	$\begin{pmatrix} -70 \\ 40 \\ 0 \end{pmatrix}$
2	$\begin{pmatrix} -22.30 \\ -25 \\ 291.27 \end{pmatrix}$	$\begin{pmatrix} -70 \\ -40 \\ 0 \end{pmatrix}$

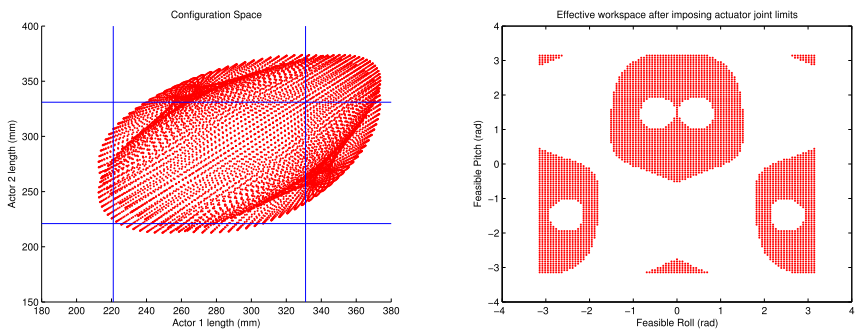


Figure 4.20 Configuration space and orientation workspace for ankle design based on 2SPU+1U mechanism [1].

be easily applied to 2SPU+1U mechanism by substituting $r = 0$. To make a comparison between the 2SPRR+1U and 2SPU+1U designs, we perform the kinematic analysis of 2SPU+1U mechanism based on the design parameters provided in Table 4.6. Same attachment points are used for this ankle design to compare against the ankle based on 2SPRR+1U architecture (compare with Table 4.1). For example, Fig. 4.20 shows the configuration space and orientation workspace of this mechanism. In comparison to the workspace of 2SPRR+1U mechanism, as shown in Fig. 4.8, it can be noticed that 2SPU+1U mechanism has a poor orientation workspace, especially for pure roll movements. Hence, the 2SPRR+1U architecture is the preferred solution for the ankle design since it provides the ideal force transmission without compromising on the ankle workspace. Fig. 4.21 shows the singularity curve and inverse of $\text{cond}(\mathbf{J})$ over workspace for 2SPU+1U mechanism.

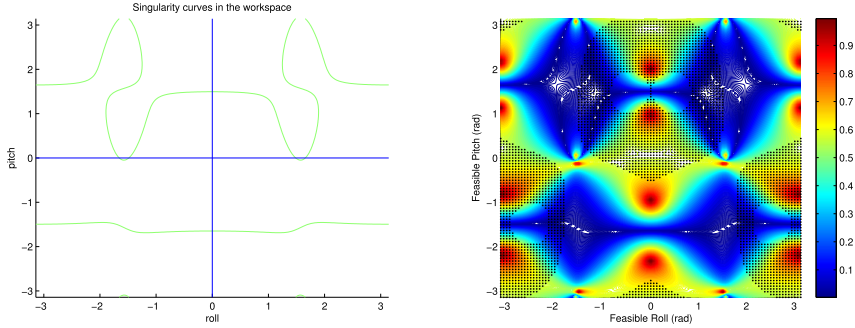


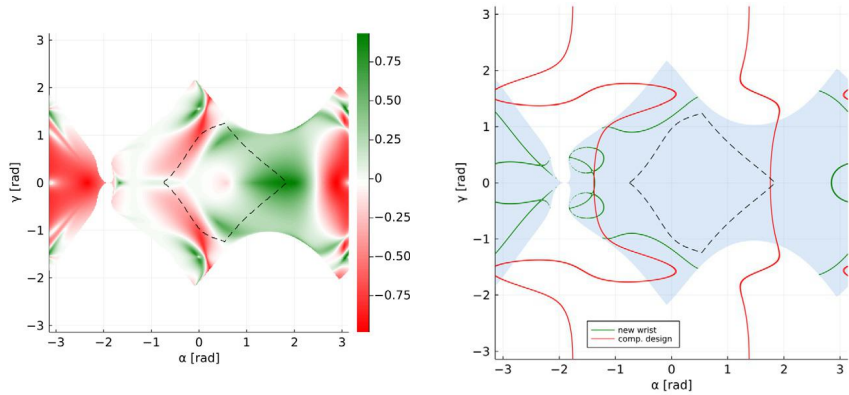
Figure 4.21 Singularity curve and inverse of condition number over workspace for ankle design based on 2SPU+1U mechanism [1].

4.4.4 Comparison between $2SPU + 2RSU + 1U$ and $2SPU + 1U$ designs

We again compare the design $2SPU + 2RSU + 1U$ to the classical $2SPU + 1U$ mechanism, where the assumption is made that the prismatic actuators act directly from their base locations onto the end-effector hinge points. This can be seen as if the intermediate linkage is left out from the wrist mechanism. Improvements in terms of bigger usable work space and improved conditioning can be seen from Fig. 4.22. The $2SPU + 2RSU + 1U$ leaves us with an overall reduced work space, but is capable to reach bigger inclination angles under a higher dexterity.

4.5 Conclusion

This chapter presents three different mechanisms to create active universal joints, where their implementations are highlighted as joint abstractions in humanoid robots. We showed that the classical $2SPU + 1U$ arrangement can be seen as special case of the $2SPRR + 1U$ design, when the joint axes of the revolute joints intersect. These designs are implemented as torso and ankle joint respectively. As such, in depth kinematic analysis of these mechanisms is given and the relevant equations are detailed. Another variant with an additional linkage, the $2SPU + 2RSU + 1U$ mechanism, is shown alongside and the same analysis, comprising workspace conditioning, singularities, and performance, is presented. This latter arrangement finds its application as a humanoid wrist joint. We showed that these three designs are beneficial in their uses as joint abstractions for humanoid robots, especially when considering ankle and wrist joint as extensions of the more



(a) Color map of the difference between wrist mechanism and comparative design, where green and red indicate superior inferior conditioning, respectively.

(b) Singularity curves wrist design and comparative mechanism, where the reduced workspace of the wrist is indicated by the gray area.

Figure 4.22 Comparison between $2SPU + 2RSU + 1U$ and $2SPU + 1U$ mechanism. Original Source: [3] (Reproduced with permission from Springer Nature).

traditional torso joint. This is demonstrated by comparative analysis of these designs, regarding also human-like data.

References

- [1] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
- [2] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Mueller, F. Kirchner, Kinematic analysis of a novel parallel $2SPRR + 1U$ ankle mechanism in humanoid robot, in: M. Carricato (Ed.), *Advances in Robot Kinematics*, Springer Verlag GmbH, Cham, 2018.
- [3] C. Stoeffler, A. del Rio Fernandez, H. Peters, M. Schilling, S. Kumar, Kinematic analysis of a novel humanoid wrist parallel mechanism, in: O. Altuzarra, A. Kecskeméthy (Eds.), *Advances in Robot Kinematics 2022*, Springer International Publishing, Cham, 2022, pp. 348–355.
- [4] J. Eßer, S. Kumar, H. Peters, V. Bargsten, J.d.G. Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid robot, in: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021, pp. 400–407, <https://doi.org/10.1109/HUMANOIDS47582.2021.9555770>.
- [5] S. Lohmeier, T. Buschmann, H. Ulbrich, F. Pfeiffer, Modular joint design for performance enhanced humanoid robot LOLA, in: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006*, pp. 88–93.
- [6] A.B. Zoss, H. Kazerooni, A. Chu, Biomechanical design of the Berkeley lower extremity exoskeleton (BLEEX), *IEEE/ASME Transactions on Mechatronics* 11 (2) (2006) 128–138.
- [7] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in:

2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 01–07, <https://doi.org/10.1109/ICRA46639.2022.9811843>.

- [8] S. Kumar, A. Mueller, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, in: Vol. 5A: 43rd Mechanisms and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2019, v05AT07A054, <https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2019/59230/V05AT07A054/6453674/v05at07a054-detc2019-97115.pdf>, <https://doi.org/10.1115/DETC2019-97115>.

CHAPTER 5

3-DOF orientational parallel mechanism

Shivesh Kumar^a, Bertold Bongardt^b, and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute for Robotics and Process Informatics, Technical University of Braunschweig, Braunschweig, Germany

^cWorking Group Robotics, University of Bremen, Bremen, Germany

This chapter presents the study of the novel 3R-[2SS] mechanism, also called as ACTIVE ANKLE, which has been used for the abstraction of a spherical joint in the design of Recupera Reha exoskeleton (see Fig. 5.1). The chapter is organized as follows: In Section 5.2, the design and the construction of the ACTIVE ANKLE is reflected in comparison to the state-of-the-art and its general mobility is determined. In Section 5.4, the inverse kinematic problems and solution methods suitable for its kinematic control are presented. Finally, conclusions are drawn in Section 5.5. The content of this chapter is based on [2] (reproduced with permission from Springer Nature).

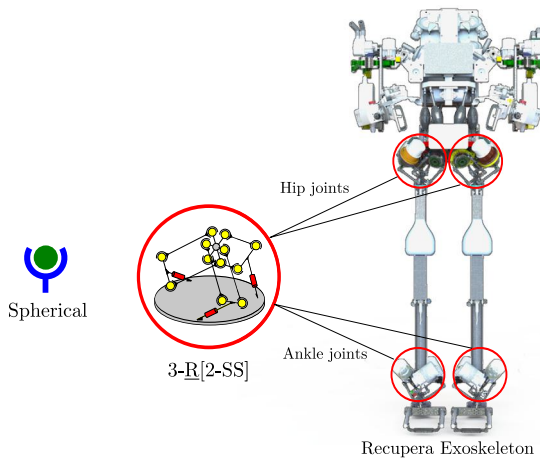


Figure 5.1 Spherical joint abstraction in RECUPERA exoskeleton [3] at DFKI-RIC [1].

5.1 Introduction

If the location of the end-effector of a PM remains constant, the device is called a spherical parallel manipulator (SPM). The AGILE EYE [4] and its improved variant AGILE WRIST [5] are prominent examples of SPMs with three degrees of freedom (DOF). The joint axes of this class of spherical manipulators are required to intersect in a single point. However, due to machining and assembling errors, it is difficult to achieve an accurate intersection of all joint axes [6]. Misalignments may lead to undesirable reaction forces in the structure and hence to a reduced service life of the mechanism or sometimes make it difficult to assemble the complete system [7]. Moreover, the use of C-shaped links in the system prevents the design from being used in high payload applications. Due to the kinematic layout that requires an exact intersection of all rotation axes, a high-precision manufacturing is indispensable for these SPMs [8]. The ARGOS mechanism, an SPM with three DOF, was developed by Vischer and Clavel [7] to overcome these shortcomings. Their 3[R[RR/SS]S]-design consists of three identical legs containing a revolute joint at the base, whose axis is pointing to a virtual rotation center.

A novel, almost-spherical parallel manipulator (ASPM) ACTIVE ANKLE (Fig. 5.2) has recently been introduced in [9] and [10]. Due to its unique, simple and compact 3[R2[SS]] design, the constraint of moving the end-effector about an exact center (of rotation) in case of spherical parallel manipulators (SPM) is relaxed to almost spherical motions that includes a shift of the end effector about a tolerated, very small domain. Due to the presence of a closed loop in each leg, the mechanism offers high stiffness and orientation accuracy. The mechanism features a low link diversity and its simple, robust, and modular design makes it highly suitable for many applications. While the primary application of the ACTIVE ANKLE is an active ankle joint in an exoskeleton or a humanoid, it could also be integrated as a submechanism into a regional manipulator for obtaining precise 6-DOF motions if the constrained translations of the ASPM are compensated by the previous joints of the overall device.

5.2 Mechanism's design description

5.2.1 Type synthesis

The geometric type of the spatial almost-spherical parallel mechanism ACTIVE ANKLE are set into context in Table 5.1 and Table 5.2. The vari-

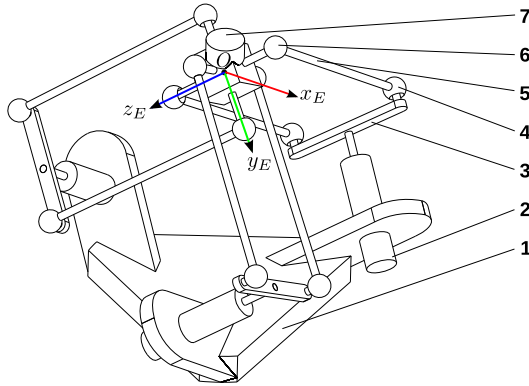


Figure 5.2 Sketch of the ACTIVE ANKLE [9] including (1) base, (2) rotative actuator, (3) crank, (4 and 6) ball and socket joints, (5) rod, (7) end-effector. Original Source: [2] (reproduced with permission from Springer Nature).

ous possible leg configurations can be derived using the Kutzbach–Grübler formula. Table 5.1 describes the possibilities by using the relation between the desired degree of freedom of the parallel manipulator d , the number of kinematic chains k , and the sum of the joint DOF of each chain f . Each kinematic leg can be realized by a serial arrangement of links and joints or with closed loops. The latter comes with an inherent advantage of increased stiffness. For example, in the famous DELTA robot with 3 DOF, each of its three legs is realized by a closed parallelogram (4S) mechanism, which makes it a stiff positioning system. This is an inspiration for finding a novel parallel manipulator which can produce spherical movements while still keeping the topological arrangement of DELTA robot. With a homogeneous distribution of five DOF to all three legs (Table 5.1), the type of the ACTIVE ANKLE matches those of the DELTA robot. The topological setup of both mechanisms also equals on the level of each of the three identical legs. Both consist of one rotative actuator in series with one closed loop with four spherical joints (Fig. 5.3). For these reasons, the ACTIVE ANKLE can be classified as the (almost) rotative counterpart of the DELTA robot. In comparison to the DELTA robot, which provides a stiff positioning functionality, the ACTIVE ANKLE provides a stiff orientating feature, due to the employment of parallel structures within the three kinematic chains.

Table 5.1 Overview of spatial parallel manipulators with general mobility d with distributions of degrees of freedom to k kinematic chains (legs), in accordance to [11]. Original Source: [2] (reproduced with permission from Springer Nature).

	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
$k = 2$					
$k = 3$	—				
$k = 4$	—	—			
$k = 5$	—	—	—		
$k = 6$	—	—	—	—	

Table 5.2 Examples of mechanisms with respect to type and mobility. *Watt’s and Chebyshev’s linkages are almost prismatic [12]. Original Source: [2] (reproduced with permission from Springer Nature).

Mechanism type			General mobility d		
Motion	Group	Dim.	1	3	6
Position	P^2	2	Peaucellier–Lipkin*	—	—
Flat	P^2R	3	Planar 4R	Planar Stewart	—
Spherical	R^3	3	Spherical 4R	Agile Eye, Argos	—
Position	P^3	3	Sarrus	Delta robot	—
Spatial	P^3R^3	6	Bennett 4R	Active Ankle	Stewart

5.2.2 Design and construction

The mechanical layout of ACTIVE ANKLE is modular and depicted in Fig. 5.2: the device features three rotative actuators fixed to the base. Each of the three motors drives a spatial quadrilateral consisting of a symmetric crank, two rods, and a line segment on the mobile platform. The three line segments mutually intersect orthogonally and together form a spatial cross on the end-effector link. The total weight of the mechanism including the three actuators is 1.8 kg. With regard to the electronics, the device features three actuator modules which include a brushless DC motor coupled with harmonic gear drives (nominal torque 28 Nm, weight 0.392 kg), FPGA based control, and power electronics. Each actuator module is capable of

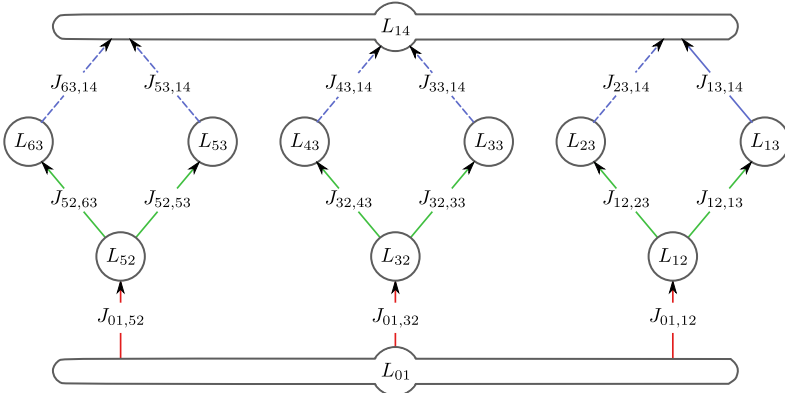


Figure 5.3 Link graph of the parallel manipulator ACTIVE ANKLE, including $n = 11$ links and $m = 15$ joints [2] (reproduced with permission from Springer Nature).

a cascaded position, velocity, and current-based torque control [10]. The presented prototype of this mechanism is designed to carry static loads up to 30 kg in the zero configuration.

5.2.3 Topology and general mobility

The topology of the mechanism is equivalent to DELTA robot, as depicted in Fig. 5.3. The $n = 11$ links L_i are enumerated as L_{01} , L_{12} , L_{13} , L_{14} , L_{23} , L_{32} , L_{33} , L_{43} , L_{52} , L_{53} , and L_{63} . The $m = 15$ joints $J_{i,j}$ are distinguished using double indices, as indicated in Fig. 5.3. The number of independent loops of the ACTIVE ANKLE is computed with $c = m - n + 1 = 15 - 11 + 1 = 5$. The general mobility of the mechanism can be estimated by means of the Kutzbach–Grübler formula: $d_s(\mathcal{M}) = s(n - m - 1) + f = s(-c) + f$, where the total number of freedoms $f = \sum_{ij} f_{ij}$ is determined by considering three rotative joints, six spherical joints, and six universal joints, which results in $f = 3 \cdot 1 + 6 \cdot 3 + 6 \cdot 2 = 3 + 18 + 12 = 33$. Since the device is *almost* spherical, the motion parameter $s = 6$ (spatial) and $s = 3$ (spherical). Hence, the mobility of the device can be computed as: $d_s(\mathcal{M}) = 6 \cdot (11 - 15 - 1) + 33 = 3$.

5.2.4 Design features

The mechanism's homogeneous and simple design leads to a low link diversity, permits a low-cost construction, and provides robustness against

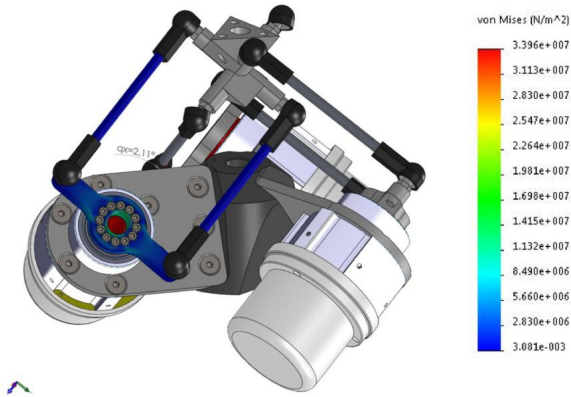


Figure 5.4 FEM analysis of the ACTIVE ANKLE. Original Source: [2] (reproduced with permission from Springer Nature).

production inaccuracies. A crucial feature of the mechanism's design is the stress distribution among the structure. The six rods that transmit the forces from the cranks to the platform are only loaded with forces along their axes due to the spherical joints attached to them. Moreover, any force applied along the direction of its platform's torsional axis can be supported without an active torque in the motors.

A multibody dynamics simulation analysis and a subsequent FEM analysis have been performed to check the deformation of the critical parts as rods and cranks under desired loads (Fig. 5.4). A force corresponding to the weight of the exoskeleton is applied to the end effector and the forces in the spherical joints are measured. In the zero configuration, this force—equivalent to 350 N perpendicular to the end effector's top plate—leads to a reaction force of approximately 100 N in each spherical joint. The selected ball and socket joints are designed for a maximum axial tensile force of 600 N in housing axis and a pivot angle of maximum of $\pm 25^\circ$. The same magnitude of force occurs in the rods and this force has been found to be less than the buckling force of the rods (i.e., 2120 N). Thus, it is ensured that the mechanism resists from buckling in all possible configurations [10].

5.2.5 Design comparison

In this section, the design of the almost-spherical mechanism Active Ankle is analyzed from a principal and from an application-motivated point of

Table 5.3 A comparison of mechanisms, in terms of their members, links n , joints m , and number of independent loops $c = m - n + 1$; quoted from [10]. Original Source: [2] (reproduced with permission from Springer Nature).

Mechanism	Ref.	Links n	Joints m	Loops c
RRR / Cardan	[13]	4	3 (6)	0 (3)
Agile Eye / Wrist	[4]	8	9	2
AsySPM	[14]	11	13	3
CamSPM3	[15]	8	10	4
Hexasphere	[16]	14	19	6
Active Ankle	[9]	11	15	5

view: First, its design is compared to that of spherical mechanisms, and second, its design is set into contrast with devices intended to interoperate with the human ankle.

Spherical mechanisms

In Table 5.3, the almost-spherical ACTIVE ANKLE is briefly compared to a set of (purely) spherical devices.¹ The RRR chain and the Cardan mechanism [13] with three intersecting axes represent the simplest spherical devices: due to their serial construction, they lack the stiffness that is offered by their parallel counterparts. AGILE EYE and its variants are Spherical Parallel Manipulators (SPM) that offer high speeds for low payloads. Due to their design, they require high manufacturing and assembly accuracies. The design of the Asymmetrical Spherical Parallel Manipulator ASYSPM [14] involves the use of large number of different parts due to its asymmetrical leg configuration. In comparison to the Active Ankle, the 3-SPS manipulator (CAMSPM3 in Table 5.3) [15] follows a complementary actuation approach: prismatic, instead of revolute joints are employed to actuate the platform. The HEXASPHERE [16] is a redundant SPM that features six motors to achieve the three rotative degrees of freedom of the platform.

Ankle exoskeletons

The ACTIVE ANKLE is primarily designed to work as an active interface to three DOF human joints. Its application at the hip and the ankle joints

¹ The presented comparison is an outline of a more detailed argumentation [10].

within the novel full body RECUPERA exoskeleton [3,17]. While the exoskeleton is primarily designed for upper body rehabilitation,² the main purpose of the legs is to transfer the load of the upper body exoskeleton system to the ground and provide some mobility features (e.g., sitting, standing, walking, etc.) to the human subject: the RECUPERA legs and the integrated ACTIVE ANKLE instances are considered as *load transfer* devices according to the classification by Herr.³ In contrast to the similar load carrying exoskeleton BLEEX [23], which only features four active DOF per leg, the RECUPERA exoskeleton provides seven active DOF in each leg, due to the role of ACTIVE ANKLE as a modular spherical unit. The hydraulically-damped ankle-foot orthosis by Yamamoto et al. [24] and the knee-ankle-foot exoskeleton KAFO, driven by artificial pneumatic muscles [25] both only provide a single DOF at the ankle joint of the human.

5.3 Mechanism architecture and constraint equations

In this section, the parameterizations of the end effector and crank points are presented and the constraint equations of the mechanism are derived. The six points (e_1, \dots, e_6) on the end effector lie on a sphere with radius d . The points c_i and c_j rotate around b_{ij} in circles of radius r for $ij \in \{12, 34, 56\}$. The length of the six rods is denoted by l . The global frame O is coincident with the end effector position e when the mechanism is in its zero-configuration (Fig. 5.2). The unit vectors \hat{s} , \hat{n} and \hat{a} are vectors along the x_E , y_E , and z_E axes, expressed in the global frame O .

End-effector points

The points e_i , $i \in \{1, \dots, 6\}$ are rigidly attached to the end-effector. Fig. 5.5 shows that the pair of points (e_1, e_2) lies on a line $\mathcal{L}_{12} = (e, \hat{n})$ along unit vector \hat{n} passing through point e . Similarly, the pairs (e_3, e_4) and (e_5, e_6) lie

² The exoskeleton designs for upper body rehabilitation are usually attached to a fixed base (e.g., ARMIN [18], Recupera wheelchair system [19]) or to the patient's torso (e.g., RUPERT [20]), which either reduces the mobility of patients or forces the patient to carry the weight of the exoskeleton which might be difficult for weaker stroke patients. A more detailed survey of exoskeletons for upper body rehabilitation can be found in [21].

³ Herr [22] distinguishes parallel-limb exoskeletons according to their function, "load transfer to the ground", "torque and work augmentation", and "increase human endurance". Active devices are named "exoskeletons", passive devices are named "orthoses".

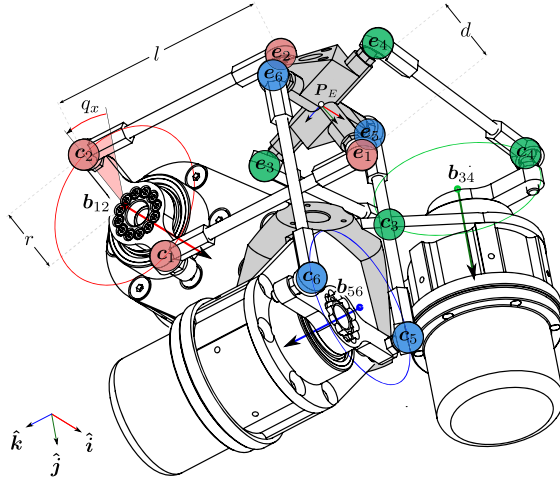


Figure 5.5 A posture of the ACTIVE ANKLE corresponding to the configuration $\mathbf{q} = (q_x, q_y, q_z) \approx (-25^\circ, 0^\circ, 0^\circ)$. The design parameters are $d = r = 35$ mm and $l = 100$ mm. Original Source: [2] (reproduced with permission from Springer Nature).

on lines $\mathcal{L}_{34} = (e, \hat{\mathbf{a}})$ and $\mathcal{L}_{56} = (e, \hat{\mathbf{s}})$ respectively. The coordinates of these points in terms of end effector position (e) and orientation ($\hat{\mathbf{s}}, \hat{\mathbf{n}}, \hat{\mathbf{a}}$) are expressed as:

$$\begin{aligned}
 e_1 &= e + d\hat{\mathbf{n}} & e_2 &= e - d\hat{\mathbf{n}} \\
 e_3 &= e + d\hat{\mathbf{a}} & e_4 &= e - d\hat{\mathbf{a}} \\
 e_5 &= e + d\hat{\mathbf{s}} & e_6 &= e - d\hat{\mathbf{s}}
 \end{aligned} \tag{5.1}$$

The position vectors of six end effector points are stored column-wise in matrix $\mathbf{E} = (e_1 \dots e_6)$. The parameterization of six end-effector points using the end effector pose is implemented in the method Calculate End-effector Points (CEP) (Algorithm 5.5).

Crank points

The crank points c_i , $i \in \{1, \dots, 6\}$ are allowed to move on the circles defined by the motion of three actuators. The pair of points (c_i, c_j) lie diametrically opposite to each other on a circle of radius r with center \mathbf{b}_{ij} , $ij \in \{12, 34, 56\}$. The position vector of six crank points is parameterized

using input joint angles (q_x, q_y, q_z) with the set of equations

$$\begin{aligned} \mathbf{c}_1(q_x) &= \mathbf{b}_{12} + \mathbf{c}_{12}(q_x) & \mathbf{c}_2(q_x) &= \mathbf{b}_{12} - \mathbf{c}_{12}(q_x) \\ \mathbf{c}_3(q_y) &= \mathbf{b}_{34} + \mathbf{c}_{34}(q_y) & \mathbf{c}_4(q_y) &= \mathbf{b}_{34} - \mathbf{c}_{34}(q_y) \\ \mathbf{c}_5(q_z) &= \mathbf{b}_{56} + \mathbf{c}_{56}(q_z) & \mathbf{c}_6(q_z) &= \mathbf{b}_{56} - \mathbf{c}_{56}(q_z). \end{aligned} \quad (5.2)$$

In Eq. (5.2), centers $(\mathbf{b}_{12}, \mathbf{b}_{34}, \mathbf{b}_{56})$ lie on (yz, zx, xy) planes at a distance of l units along (z, x, y) axes respectively. The general points $(\mathbf{c}_{12}, \mathbf{c}_{34}, \mathbf{c}_{56})$ on these circles are described as

$$\begin{aligned} \mathbf{b}_{12} &= l\hat{\mathbf{k}}, & \mathbf{c}_{12}(q_x) &= r\cos(q_x)\hat{\mathbf{j}} + r\sin(q_x)\hat{\mathbf{k}} \\ \mathbf{b}_{34} &= l\hat{\mathbf{i}}, & \mathbf{c}_{34}(q_y) &= r\cos(q_y)\hat{\mathbf{k}} + r\sin(q_y)\hat{\mathbf{i}} \\ \mathbf{b}_{56} &= l\hat{\mathbf{j}}, & \mathbf{c}_{56}(q_z) &= r\cos(q_z)\hat{\mathbf{i}} + r\sin(q_z)\hat{\mathbf{j}}. \end{aligned}$$

The position vectors of six crank points are stored column-wise in matrix $\mathbf{C} = (\mathbf{c}_1 \dots \mathbf{c}_6)$. The parameterization of six crank points using the input joint angles is implemented in the method Calculate Crank Points (CCP) (Algorithm 5.6).

Kinematic constraint equations

The length of the line segment joining the crank points (\mathbf{c}_i) to the end effector points (\mathbf{e}_i) equals the rod length l .

$$\|\mathbf{e}_i - \mathbf{c}_i\| = l, \quad i \in \{1, \dots, 6\}. \quad (5.3)$$

Substituting Eq. (5.1) and Eq. (5.2) in Eq. (5.3), and squaring both sides, the six distance constraints equations are derived:

$$(e_x + dn_x)^2 + (e_y + dn_y - r\cos(q_x))^2 + (e_z + dn_z - l - r\sin(q_x))^2 = l^2 \quad (5.4)$$

$$(e_x - dn_x)^2 + (e_y - dn_y + r\cos(q_x))^2 + (e_z - dn_z - l + r\sin(q_x))^2 = l^2 \quad (5.5)$$

$$(e_x + da_x - l - r\sin(q_y))^2 + (e_y + da_y)^2 + (e_z + da_z - r\cos(q_y))^2 = l^2 \quad (5.6)$$

$$(e_x - da_x - l + r\sin(q_y))^2 + (e_y - da_y)^2 + (e_z - da_z + r\cos(q_y))^2 = l^2 \quad (5.7)$$

$$(e_x + ds_x - r\cos(q_z))^2 + (e_y + ds_y - l - r\sin(q_z))^2 + (e_z + ds_z)^2 = l^2 \quad (5.8)$$

$$(e_x - ds_x + r\cos(q_z))^2 + (e_y - ds_y - l + r\sin(q_z))^2 + (e_z - ds_z)^2 = l^2 \quad (5.9)$$

Problem overview

In Table 5.4, an overview of the nature of kinematics problems is presented based on the dimensionality of the input and output variables and

Table 5.4 Overview of problem characteristics, $\dim(\mathbf{R}_E) = 3$, $\dim(\mathbf{e}) = 3$, $\dim(\mathbf{q}) = 3$. * denotes the relaxed problem. Original Source: [2] (reproduced with permission from Springer Nature).

Type	Well-determined				Over-determined			
Direction	Name	In	Eqs	Out	Name	In	Eqs	Out
Inverse	RIKP	\mathbf{R}_E	$\xrightarrow{6}$	(\mathbf{q}, \mathbf{e})	IKP	$(\mathbf{R}_E, \mathbf{e})$	$\xrightarrow{6}$	\mathbf{q}
	IKP*	$(\mathbf{R}_E, \mathbf{e})$	$\xrightarrow{3}$	\mathbf{q}				
Forward	FKP	\mathbf{q}	$\xrightarrow{6}$	$(\mathbf{R}_E, \mathbf{e})$	TFKP	$(\mathbf{q}, \mathbf{R}_E)$	$\xrightarrow{6}$	\mathbf{e}
	TFKP*	$(\mathbf{q}, \mathbf{R}_E)$	$\xrightarrow{3}$	\mathbf{e}				

the number of constraint equations. With regard to the dimensionality of the unknown variables and number of equations, it can be noticed that the inverse kinematics problem (IKP) for this mechanism is over-determined, while the forward kinematics problem (FKP) is well determined. From the point of view of kinematic control, inverse kinematics problem in its original form is not relevant due to almost spherical nature of this mechanism. It is intended to be used as a spherical device, and hence this demands the solution to a rotative inverse kinematics problem (RIKP) which involves finding a joint configuration from a given platform orientation in $SO(3)$ instead of a given platform pose in $SE(3)$. This problem is again well-determined.

5.4 Inverse kinematics

In this section, the inverse and rotational inverse kinematics problems are presented along with their solution methods.

5.4.1 Inverse kinematics

Problem 5.1 (Inverse Kinematics). The Inverse Kinematics Problem (IKP) is defined as the problem of finding the input joint angles needed to achieve a specific pose of the end effector [26], formally,

$$[q_x, q_y, q_z] = \text{IKP}(\mathbf{P}_E), \quad \mathbf{P}_E \in SE(3),$$

where \mathbf{P}_E is the homogeneous transformation matrix of the end effector E with respect to the global frame O and $[q_x, q_y, q_z]$ are the active revolute

Algorithm 5.1 Inverse kinematic model (IKM). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Target pose \mathbf{P}_e

(out) Joint configuration (q_x, q_y, q_z)

```

1: function IKM( $\mathbf{P}_e$ )
2:    $(\mathbf{e}_1, \dots, \mathbf{e}_6) \leftarrow \text{CEP}(\mathbf{P}_e)$  ▷ Platform coords
3:   for  $ij \in \{12, 34, 56\}$  do
4:      $\mathbf{p}_{i+}, \mathbf{p}_{i-} \leftarrow \mathcal{S}(\mathbf{e}_i, d_i) \cap \mathcal{C}(\mathbf{b}_{ij}, \frac{c}{2}, \hat{\mathbf{z}}_{ij})$  ▷ Sphere-circle intersections
       for  $i$ 
5:      $\mathbf{p}_{j+}, \mathbf{p}_{j-} \leftarrow \mathcal{S}(\mathbf{e}_j, d_j) \cap \mathcal{C}(\mathbf{b}_{ij}, \frac{c}{2}, \hat{\mathbf{z}}_{ij})$  ▷ Sphere-circle intersections
       for  $j$ 
6:      $\mathcal{I} \leftarrow \{\mathbf{p}_{i+}, \mathbf{p}_{i-}\}, \mathcal{J} \leftarrow \{\mathbf{p}_{j+}, \mathbf{p}_{j-}\}$ 
7:      $\mathbf{p}_+, \mathbf{p}_- \leftarrow \underset{p_i \in \mathcal{I}, p_j \in \mathcal{J}}{\text{argmax}}((\mathbf{p}_i - \mathbf{b}_{ij})^\odot \cdot (\mathbf{b}_{ij} - \mathbf{p}_j)^\odot)$  ▷ Operator
        $\odot : \mathbf{a} \mapsto \hat{\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}$ 
8:      $\mathbf{r}_{ij} \leftarrow \mathbf{p}_+ - \mathbf{p}_-$  ▷ Antipodes
9:      $\mathbf{d}_{ij} \leftarrow \mathbf{c}_i^{(0)} - \mathbf{c}_j^{(0)}$  ▷ Zero Posture
10:     $q_x, q_y, q_z \leftarrow \angle(\mathbf{d}_{12}, \mathbf{r}_{12}), \angle(\mathbf{d}_{34}, \mathbf{r}_{34}), \angle(\mathbf{d}_{56}, \mathbf{r}_{56})$ 
11:    return  $(q_x, q_y, q_z)$ 

```

joint angles.

$$\mathbf{P}_E = \begin{bmatrix} s_x & n_x & a_x & e_x \\ s_y & n_y & a_y & e_y \\ s_z & n_z & a_z & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As noted in Table 5.4, in context of the IKP, the system of non-linear equations is overdetermined, as the number of unknowns (three) is less than the number of equations (six).

The method IKM in Algorithm 5.1 provides an analytical solution method to IKP. The computation of the intersections of sphere and circle in Line 4 and Line 5 of Algorithm 5.1 can be conducted by means of the intersection method SPHINT (Algorithm 5.4) for three spheres.⁴ In Line 7, a pair of antipodal points is selected from the set of four intersection

⁴ For a given a circle $\mathcal{C}(\mathbf{m}_C, r_C, \hat{\mathbf{n}}_C)$ with midpoint \mathbf{m}_C , radius r_C , and unit normal $\hat{\mathbf{n}}_C$, two substituting spheres $\mathcal{S}_A(\mathbf{m}_A, r_A)$ and $\mathcal{S}_B(\mathbf{m}_B, r_B)$ are given by the midpoints $\mathbf{m}_A, \mathbf{m}_B = \mathbf{m}_C \pm \frac{4}{3}r_C\hat{\mathbf{n}}_C$ and the radii $r_A = r_B = \frac{5}{3}r_C$.

points by maximizing the cosine similarity between two normalized difference vectors. The line segment between the selected two points represents the current alignment of the rod. The angle between the current alignment and the zero reference alignment (Line 9) of one rod yields the angle of one input joint, determined in Line 10 of Algorithm 5.1.

Since the IKM solution depends on the knowledge of the end effector shift (e_x, e_y, e_z) , it is not sufficient for achieving a kinematic control of the mechanism in spherical task space $SO(3)$. Therefore, it is required to calculate the input joint angles only from the desired orientation of the end-effector.

Algorithm 5.2 Matrix minor (MINOR). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, row indices $R = (r_1, r_2, \dots, r_p)$, column indices $C = (c_1, c_2, \dots, c_q)$, $x \in \{0, 1\}$

(out) Determinant of the submatrix $\mathbf{A}_{[R][C]} \in \mathbb{R}^{p \times q}$

```

1: function MINOR( $\mathbf{A}$ ,  $R$ ,  $C$ ,  $x$ )
2:   if  $x = 0$  then                                     ▷ Index handling
3:     do  $r_i \leftarrow r_i + 1$  for  $r_i \in R$ 
4:     do  $c_j \leftarrow c_j + 1$  for  $c_j \in C$ 
5:      $\mathbf{A}_{[R][C]} \leftarrow \text{EXTRACT}(\mathbf{A}, R, C)$            ▷ Submatrix
6:      $m \leftarrow \det(\mathbf{A}_{[R][C]})$                        ▷ Minor
7:   return  $m$ 

```

Algorithm 5.3 Submatrix extraction (EXTRACT). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, row indices $R = (r_1, r_2, \dots, r_p)$ with $1 \leq r_i \leq m$ for $1 \leq i \leq p$, and column indices $C = (c_1, c_2, \dots, c_q)$ with $1 \leq r_j \leq n$ for $1 \leq j \leq q$

(out) Submatrix $\mathbf{A}_{[R][C]} \in \mathbb{R}^{p \times q}$ extracted by R and C

```

1: function EXTRACT( $\mathbf{A}$ ,  $R$ ,  $C$ )
2:    $\mathbf{R} \leftarrow (\mathbf{e}_{r_1}^m \ \mathbf{e}_{r_2}^m \ \dots \ \mathbf{e}_{r_p}^m)^T$            ▷  $\mathbf{R} \in \mathbb{R}^{p \times m}$ 
3:    $\mathbf{C} \leftarrow (\mathbf{e}_{c_1}^n \ \mathbf{e}_{c_2}^n \ \dots \ \mathbf{e}_{c_q}^n)$            ▷  $\mathbf{C} \in \mathbb{R}^{n \times q}$ 
4:    $\mathbf{A}_{[R][C]} \leftarrow \mathbf{RAC}$ 
5:   return  $\mathbf{A}_{[R][C]}$ 

```

Algorithm 5.4 Intersection of three spheres (SPHINT). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Spheres; midpoints $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ and radii r_1, r_2, r_3

(out) Intersection; points \mathbf{p}_+ and \mathbf{p}_- , with $\|\mathbf{p}_+ - \mathbf{s}_i\| = r_i$ and $\|\mathbf{p}_- - \mathbf{s}_i\| = r_i$ for $i \in \{1, 2, 3\}$, or empty set

```

1: function SPHINT( $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, r_1, r_2, r_3$ )
2:    $R_1 \leftarrow r_1^2, R_2 \leftarrow r_2^2, R_3 \leftarrow r_3^2$ 
3:    $\mathbf{Q} \leftarrow \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & Q(\mathbf{m}_1, \mathbf{m}_2) & Q(\mathbf{m}_1, \mathbf{m}_3) & R_1 \\ 1 & Q(\mathbf{m}_2, \mathbf{m}_1) & 0 & Q(\mathbf{m}_2, \mathbf{m}_3) & R_2 \\ 1 & Q(\mathbf{m}_3, \mathbf{m}_1) & Q(\mathbf{m}_3, \mathbf{m}_2) & 0 & R_3 \\ 1 & R_1 & R_2 & R_3 & 0 \end{pmatrix}$ 
4:    $D_{(1234)} \leftarrow \frac{1}{8} \det(\mathbf{Q})$  ▷ CM determinant
5:   if  $D_{(1234)} < 0$  then ▷ Empty intersection
6:     return  $\emptyset$ 
7:    $D_{(123)} \leftarrow -\frac{1}{4} \text{MINOR}(\mathbf{Q}, (0, 1, 2, 3), (0, 1, 2, 3), 0)$ 
8:    $D_{(123;124)} \leftarrow -\frac{1}{4} \text{MINOR}(\mathbf{Q}, (0, 1, 2, 3), (0, 1, 2, 4), 0)$ 
9:    $D_{(123;134)} \leftarrow -\frac{1}{4} \text{MINOR}(\mathbf{Q}, (0, 1, 2, 3), (0, 1, 3, 4), 0)$ 
10:   $\mathbf{v}_1 \leftarrow \mathbf{m}_2 - \mathbf{m}_1$ 
11:   $\mathbf{v}_2 \leftarrow \mathbf{m}_3 - \mathbf{m}_1$ 
12:   $\mathbf{v}_0 \leftarrow -D_{(123;134)}\mathbf{v}_1 + D_{(123;124)}\mathbf{v}_2$ 
13:   $\mathbf{v}_\Delta \leftarrow \sqrt{D_{(1234)}}(\mathbf{v}_1 \times \mathbf{v}_2)$ 
14:   $\mathbf{p}_+ \leftarrow \mathbf{m}_1 + \frac{1}{D_{(123)}}(\mathbf{v}_0 + \mathbf{v}_\Delta)$ 
15:   $\mathbf{p}_- \leftarrow \mathbf{m}_1 + \frac{1}{D_{(123)}}(\mathbf{v}_0 - \mathbf{v}_\Delta)$ 
16:  return  $\mathbf{p}_+, \mathbf{p}_-$ 

```

Algorithm 5.5 Calculation of effector points (CEP). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Homogeneous transformation of end effector \mathbf{P}_E

(out) End effector point matrix \mathbf{E}

```

1: function CEP( $\mathbf{P}_E$ )
2:    $\begin{bmatrix} \hat{\mathbf{s}} & \hat{\mathbf{n}} & \hat{\mathbf{a}} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix} \leftarrow \mathbf{P}_E$  ▷ Extraction
3:    $\mathbf{e}_1 \leftarrow \mathbf{e} + d\hat{\mathbf{n}}, \mathbf{e}_2 \leftarrow \mathbf{e} - d\hat{\mathbf{n}}$ 
4:    $\mathbf{e}_3 \leftarrow \mathbf{e} + d\hat{\mathbf{a}}, \mathbf{e}_4 \leftarrow \mathbf{e} - d\hat{\mathbf{a}}$ 
5:    $\mathbf{e}_5 \leftarrow \mathbf{e} + d\hat{\mathbf{s}}, \mathbf{e}_6 \leftarrow \mathbf{e} - d\hat{\mathbf{s}}$ 
6:    $\mathbf{E} \leftarrow (\mathbf{e}_i : 1 \leq i \leq 6)$ 
7:   return  $\mathbf{E}$ 

```

Algorithm 5.6 Calculation of crank points (CCP). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Input joint angles $[q_x, q_y, q_z]$

(out) Crank point matrix \mathbf{C}

```

1: function CCP( $q_x, q_y, q_z$ )
2:    $\mathbf{c}_1 \leftarrow (0, r \cos q_x, l + r \sin q_x)^T$ 
3:    $\mathbf{c}_2 \leftarrow (0, -r \cos q_x, l - r \sin q_x)^T$ 
4:    $\mathbf{c}_3 \leftarrow (l + r \sin q_y, 0, r \cos q_y)^T$ 
5:    $\mathbf{c}_4 \leftarrow (l - r \sin q_y, 0, -r \cos q_y)^T$ 
6:    $\mathbf{c}_5 \leftarrow (r \cos q_z, l + r \sin q_z, 0)^T$ 
7:    $\mathbf{c}_6 \leftarrow (-r \cos q_z, l - r \sin q_z, 0)^T$ 
8:    $\mathbf{C} \leftarrow (\mathbf{c}_i : 1 \leq i \leq 6)$ 
9:   return  $\mathbf{C}$ 

```

5.4.2 Rotative inverse kinematic model

Problem 5.2 (Rotative Inverse Kinematics). The Rotative Inverse Kinematic Problem (RIKP) is defined as the problem of finding the input joint angles needed to achieve a desired orientation of the end effector without having the knowledge of end effector position, formally

$$[q_x, q_y, q_z, e_x, e_y, e_z] = \text{RIKP}(\mathbf{R}_E), \quad \mathbf{R}_E \in SO(3),$$

where \mathbf{R}_E is the rotation matrix of the end effector w.r.t the global frame, $[q_x, q_y, q_z]$ and $[e_x, e_y, e_z]$ are the active revolute joint angles and end effector shift, respectively.

In this case, the system of nonlinear equations Eqs. (5.4)–(5.9) is well determined, as the number of unknowns is equal to the number of equations. To the best knowledge of the authors, it is not possible to derive a closed form solution to this problem due to coupled nature of the constraint equations. Instead of employing standard nonlinear solvers, a novel tailored and efficient algorithm is presented which is suitable for real-time control of this mechanism. Its core idea is to decompose the overall equation system into two different equation sets and orthogonally iterate between their solutions to achieve the required overall solution with a desired accuracy. For concrete explanation, two subproblems related to the geometry of ACTIVE ANKLE are presented, namely, the Relaxed Inverse Kinematic

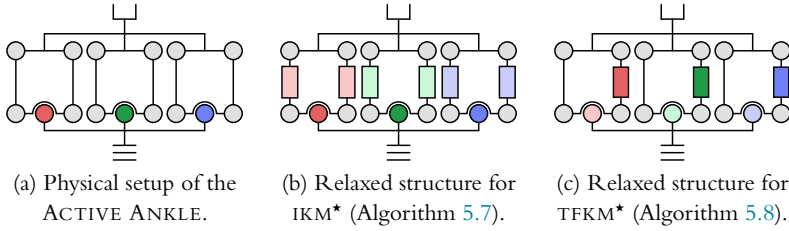


Figure 5.6 The mechanism ACTIVE ANKLE and relaxed variants featuring additional freedoms (virtual prismatic joints). Original Source: [2] (reproduced with permission from Springer Nature).

Problem (IKP*) and the Relaxed Translative Forward Kinematic Problem (TFKP*). Based on their analytical solutions, the solution to the rotational inverse kinematic problem is presented.

5.4.2.1 Relaxed inverse kinematic model

Since the nature of inverse kinematic problem is overdetermined (see Table 5.4), the two rod equations in each leg are subtracted to obtain a well-determined system of leg equations. Problem 5.1 is relaxed in the sense that it ensures $l_{e_1c_1} = l_{e_2c_2}$, $l_{e_3c_3} = l_{e_4c_4}$, $l_{e_5c_5} = l_{e_6c_6}$ and not $l_{e_i c_i} = l$, $i \in \{1, \dots, 6\}$. A geometric interpretation of this relaxation is shown in Fig. 5.6b: the rods can be interpreted as virtual prismatic joints which change their lengths in pair in each leg.

Three leg equations

Subtracting Eq. (5.5) from Eq. (5.4), Eq. (5.7) from Eq. (5.6), Eq. (5.9) from Eq. (5.8), the three leg equations are derived.

$$\begin{aligned}
 r e_y \cos q_x + r(e_z - l) \sin q_x + d(ln_z - \mathbf{e} \cdot \mathbf{n}) &= 0 \\
 r e_z \cos q_y + r(e_x - l) \sin q_y + d(la_x - \mathbf{e} \cdot \mathbf{a}) &= 0 \\
 r e_x \cos q_z + r(e_y - l) \sin q_z + d(ls_y - \mathbf{e} \cdot \mathbf{s}) &= 0.
 \end{aligned} \tag{5.10}$$

The three leg equations, with the leg index $j \in \{1, 2, 3\}$, are of the form:

$$E_j \cos(q_j) + F_j \sin(q_j) + G_j = 0. \tag{5.11}$$

Table 5.5 Parameters for IKM* solution. Original Source: [2] (reproduced with permission from Springer Nature).

Leg Index (j)	E_j	F_j	G_j
$j = 1$	re_y	$r(e_z - l)$	$d(ln_z - \mathbf{e} \cdot \mathbf{n})$
$j = 2$	re_z	$r(e_x - l)$	$d(la_x - \mathbf{e} \cdot \mathbf{a})$
$j = 3$	re_x	$r(e_y - l)$	$d(ls_y - \mathbf{e} \cdot \mathbf{s})$

Relaxed IKP solution

Using the tangent half angle substitution,

$$t_j = \tan\left(\frac{q_j}{2}\right), \quad \cos(q_j) = \frac{1 - t_j^2}{1 + t_j^2}, \quad \sin(q_j) = \frac{2t_j}{1 + t_j^2},$$

a quadratic equation in t is obtained

$$(G_j - E_j)t_j^2 + 2F_jt_j + (G_j + E_j) = 0. \quad (5.12)$$

The two solutions of the above quadratic equation is given by:

$$t_{j1,2} = \frac{-F_j \pm \sqrt{E_j^2 + F_j^2 - G_j^2}}{G_j - E_j} \quad (5.13)$$

$$q_{j+}, q_{j-} = 2 \operatorname{atan2}(-F_j \pm H_j, G_j - E_j),$$

where $H_j = \sqrt{E_j^2 + F_j^2 - G_j^2}$. The expressions for E_j , F_j , and G_j for the three legs are given in Table 5.5.

The absolute minimum of the two solutions is chosen so that the mechanism stays close to the zero configuration and respects the physical constraints imposed by either link intersection or limits of passive spherical joints. The solution is implemented in the method IKM* in Algorithm 5.7.

5.4.2.2 Relaxed translative forward kinematic model

Translative Forward Kinematic Problem (TFKP) is defined as the problem of finding the end effector shift from the input joint configuration and desired orientation of the end effector, formally

$$\mathbf{e} = \text{TFKP}(q_x, q_y, q_z, \mathbf{R}_E). \quad (5.14)$$

The solution to this problem provides the parasitic motion of the end effector. As noted in Table 5.4, this problem is an over-determined problem, as

Algorithm 5.7 Relaxed inverse kinematic model (IKM*). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Homogeneous transformation of end-effector \mathbf{P}_E

(out) Input joint angles $[q_x, q_y, q_z]$

```

1: function IKM*( $\mathbf{P}_E$ )
2:   for  $j \in (1, 2, 3)$  do
3:      $H_j \leftarrow \sqrt{E_j^2 + F_j^2 - G_j^2}$  ▷ Table 5.5
4:      $q_{j+}, q_{j-} \leftarrow 2 \operatorname{atan2}(-F_j \pm H_j, G_j - E_j)$ 
5:      $q_{j+} \leftarrow \operatorname{atan2}(\sin q_{j+}, \cos q_{j+})$  ▷ Wrap to  $\pm\pi$ 
6:      $q_{j-} \leftarrow \operatorname{atan2}(\sin q_{j-}, \cos q_{j-})$  ▷ Wrap to  $\pm\pi$ 
7:      $q_j \leftarrow \min(|q_{j+}|, |q_{j-}|)$ 
8:    $[q_x, q_y, q_z] \leftarrow (q_j : 1 \leq j \leq 3)$ 
9:   return  $[q_x, q_y, q_z]$ 

```

Algorithm 5.8 Relaxed translative forward kinematic model (TFKM*). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Input joint angles $[q_x, q_y, q_z]$ and rotation matrix \mathbf{R}_E

(out) End-effector position \mathbf{e}

```

1: function TFKM*( $q_x, q_y, q_z, \mathbf{R}_E$ )
2:    $(r_1, r_2, r_3) \leftarrow l$ 
3:    $\mathbf{s}_1 \leftarrow (-dn_x, r \cos q_x - dn_y, l - dn_z + r \sin q_x)^T$ 
4:    $\mathbf{s}_2 \leftarrow (l - da_x + r \sin q_y, -da_y, r \cos q_y - da_z)^T$ 
5:    $\mathbf{s}_3 \leftarrow (r \cos q_z - ds_x, l - ds_y + r \sin q_z, -ds_z)^T$ 
6:    $\mathbf{e}_+, \mathbf{e}_- \leftarrow \text{SPHINT}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, r_1, r_2, r_3)$  ▷ Algorithm 5.4
7:   if  $\|\mathbf{e}_+\| < d$  then
8:      $\mathbf{e} \leftarrow \mathbf{e}_+$ 
9:   else
10:     $\mathbf{e} \leftarrow \mathbf{e}_-$ 
11:   return  $\mathbf{e}$ 

```

the number of unknowns are three, while the number of constraint equations equals to six. Each rod length constraint Eqs. (5.4)–(5.9) represents the equation of a sphere where the end-effector point $[e_x, e_y, e_z]$ moves on its surface. They represent the system of equations of six spheres and the end-effector of ACTIVE ANKLE must lie at their intersection point. However, while solving the IKM* it is already ensured that the two rod lengths forming a leg should be the same. So, end effector coordinates can

be computed by solving the three rod equations, including one from each leg which makes the problem well-determined. The problem is relaxed in the sense that it ensures either $l_{e_1c_1} = l_{e_3c_3} = l_{e_5c_5} = l$ or $l_{e_2c_2} = l_{e_4c_4} = l_{e_6c_6} = l$ and not $l_{e_i c_i} = l, i \in \{1, \dots, 6\}$. A geometric interpretation of this relaxation is shown in Fig. 5.6c: the unchosen rods can be interpreted as virtual prismatic joints that adjust their lengths so that the chosen rod length becomes equal to l after solving the problem. Overall, the six sphere intersection problem reduces to a three sphere intersection problem.

Relaxed TFKP solution

Three spheres intersect in maximally two points [27]. Without the loss of generality, one can choose to solve for spheres represented by Eq. (5.4), Eq. (5.6), and Eq. (5.8). This particular choice of sphere centers (\mathbf{s}_i) and radii $r_i, i \in \{1, 2, 3\}$ is shown in the method TFKM* in Algorithm 5.8. The end-effector coordinates are estimated using

$$\mathbf{e}_+, \mathbf{e}_- = \text{SPHINT}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, r_1, r_2, r_3). \quad (5.15)$$

The method SPHINT is specified in Algorithm 5.4. The solution with a norm less than equal to d is selected to avoid the mechanism to leave its assembly. This is implemented in the method TFKM* in Algorithm 5.8.

5.4.2.3 Solution approach

The estimated end effector coordinates are defined as $\tilde{\mathbf{e}} = [\tilde{e}_x, \tilde{e}_y, \tilde{e}_z]$. In the sequel, the approximate nature of a variable x is expressed by using a tilde \tilde{x} . The homogeneous transformation matrix of the end effector w.r.t. the global frame is given by

$$\tilde{\mathbf{P}}_E = \begin{bmatrix} \mathbf{R}_E & \tilde{\mathbf{e}}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}. \quad (5.16)$$

With an estimated homogeneous transformation matrix ($\tilde{\mathbf{P}}_E$), the estimated positions of the six end-effector points stored in matrix $\tilde{\mathbf{E}}$ are calculated with the help of Algorithm 5.5 as

$$\tilde{\mathbf{E}} = \text{CEP}(\tilde{\mathbf{P}}_E). \quad (5.17)$$

The IKM* solution as presented in Section 5.4.2.1 is used to calculate the estimated input joint angles $\tilde{\mathbf{q}} = [\tilde{q}_x, \tilde{q}_y, \tilde{q}_z]$ required to achieve the estimated end effector position and desired orientation. It must be recalled that for

Algorithm 5.9 Rotative inverse kinematic model (RIKM). Original Source: [2] (reproduced with permission from Springer Nature).

(in) Desired orientation of the end effector, \mathbf{R}_E

(out) Joint angles $[q_x, q_y, q_z]$ and end effector shift $[e_x, e_y, e_z]$

```

1: function RIKM( $\mathbf{R}_E, \epsilon$ )
2:    $\tilde{\mathbf{P}}_E \leftarrow \begin{bmatrix} \mathbf{R}_E & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$  ▷ Initialization
3:   while  $E_{\text{rgd}} < \epsilon$  do
4:      $(\tilde{\mathbf{e}}_1 \dots \tilde{\mathbf{e}}_6) \leftarrow \text{CEP}(\tilde{\mathbf{P}}_E)$  ▷ Algorithm 5.5
5:      $[\tilde{q}_x, \tilde{q}_y, \tilde{q}_z] \leftarrow \text{IKM}^*(\tilde{\mathbf{P}}_E)$  ▷ Algorithm 5.7
6:      $(\tilde{\mathbf{c}}_1 \dots \tilde{\mathbf{c}}_6) \leftarrow \text{CCP}(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z)$  ▷ Algorithm 5.6
7:      $E_{\text{rgd}} \leftarrow \sum_i^6 (\|\tilde{\mathbf{e}}_i - \tilde{\mathbf{c}}_i\| - l)^2$  ▷ Rigidity error
8:      $\tilde{\mathbf{e}} \leftarrow \text{TFKM}^*(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z, \mathbf{R}_E)$  ▷ Algorithm 5.8
9:      $\tilde{\mathbf{P}}_E \leftarrow \begin{bmatrix} \mathbf{R}_E & \tilde{\mathbf{e}}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$  ▷ Update
10:     $[q_x, q_y, q_z] \leftarrow [\tilde{q}_x, \tilde{q}_y, \tilde{q}_z]$ 
11:     $[e_x, e_y, e_z] \leftarrow [\tilde{e}_x, \tilde{e}_y, \tilde{e}_z]$ 
12:  return  $[q_x, q_y, q_z, e_x, e_y, e_z]$ 

```

the derivation of three leg equations Eq. (5.10), the two distance constraint equations of each rod constituting a leg are subtracted from each other and hence forcing the two virtual rod lengths of each leg to be equal. Thus, any approximate solution to the inverse kinematic model comes at a cost of incorrect leg lengths.

$$[\tilde{q}_x, \tilde{q}_y, \tilde{q}_z] = \text{IKM}^*(\tilde{\mathbf{P}}_E). \quad (5.18)$$

The estimated input joint angles are now used to estimate the position of six crank points using Algorithm 5.6.

$$\tilde{\mathbf{C}} = \text{CCP}(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z). \quad (5.19)$$

The estimated position vectors of the six end effector points ($\tilde{\mathbf{e}}_i$) and the six crank points ($\tilde{\mathbf{c}}_i$) are extracted from end effector points matrix $\tilde{\mathbf{E}}$ Eq. (5.17) and crank points matrix $\tilde{\mathbf{C}}$ Eq. (5.19) respectively. The length of six virtual rods is calculated from $\tilde{\mathbf{e}}_i$ and $\tilde{\mathbf{c}}_i$ using:

$$\|\tilde{\mathbf{e}}_i - \tilde{\mathbf{c}}_i\| = \tilde{l}_i, \quad i \in \{1, \dots, 6\}. \quad (5.20)$$

A least square error function to minimize the change in virtual rod lengths is defined as follows⁵:

$$E_{\text{rgd}}(\tilde{\mathbf{e}}_i, \tilde{\mathbf{c}}_i) = \sum_{i=1}^6 (\tilde{l}_i - l)^2. \quad (5.21)$$

To minimize the least square error, the new end effector coordinates ($\tilde{\mathbf{e}} = [\tilde{e}_x, \tilde{e}_y, \tilde{e}_z]$) are estimated. Solving for the new end effector position in each iteration is equivalent to solving the relaxed forward kinematic model in translative domain (see Section 5.4.2.2). The solution ensures that the leg lengths become equal to l .

$$\tilde{\mathbf{e}} = \text{TFKM}^*(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z, \mathbf{R}_E). \quad (5.22)$$

The two solutions in TFKM* lead to two distinct solutions for the RIKM, out of which we are primarily interested in the solution with norm less than d . Each estimation of the end effector position using the method TFKM* minimizes the least squared error function in the next iteration. Hence, the estimated end effector coordinates are substituted back into Eq. (5.16) and the subsequent calculations are iterated until the $E_{\text{rgd}}(\tilde{\mathbf{e}}_i, \tilde{\mathbf{c}}_i) < \epsilon$ is achieved. The overall rotational inverse kinematic model is implemented in the method RIKM (Algorithm 5.9). It must be noted that $\tilde{\mathbf{e}}$ is initialized as $\mathbf{0}$ at the beginning of the algorithm (Line 2) but this choice does not affect the convergence of the algorithm.⁶ The two almost spherical working modes (solutions to the RIKM) for an axis $\mathbf{u} \approx (0.2127, 0.5344, 0.8180)^T$ and angle $\phi \approx 0.3140$ are shown in Fig. 5.7 and Fig. 5.8. The numerical convergence towards normal working mode is depicted in Table 5.6.

Benchmarking and convergence

The solution strategy presented above to solve RIKP is compared with some standard non-linear solvers like Levenberg–Marquardt (LM) and Trust Region Dog Leg (TRDL) implemented in `fsolve` function of MATLAB[®] as

⁵ For improving computational efficiency, the error function can also be chosen to minimize the change in three instead of six rod lengths: the solution of IKM* already ensures that the two rod lengths equal in each leg.

⁶ Instead, the convergence to the correct physical configuration is guaranteed by selecting the appropriate intersection point within TFKM*.

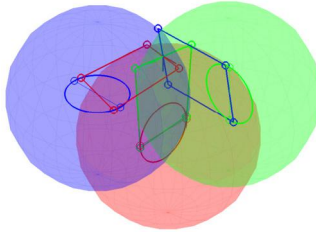


Figure 5.7 Normal working mode, active joint angles: $(q_x, q_y, q_z) \approx (0.0872, 0.1748, 0.2614)$ and end effector shift: $(e_x, e_y, e_z) \approx (0.0127, 0.1515, 0.3807)$. Original Source: [2] (reproduced with permission from Springer Nature).

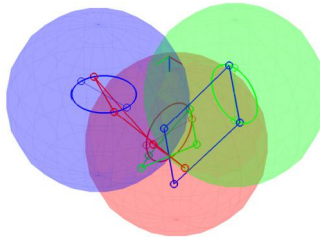


Figure 5.8 Upside – down working mode, active joint angles: $(q_x, q_y, q_z) \approx (0.4566, 0.2377, 0.4663)$ and end effector shift: $(e_x, e_y, e_z) \approx (65.6274, 65.9876, 66.7599)$. Original Source: [2] (reproduced with permission from Springer Nature).

Table 5.6 A numerical example showing the convergence of RIKM for an axis $\mathbf{u} \approx (0.2127, 0.5344, 0.8180)^T$ and an angle $\phi \approx 0.3140$. Original Source: [2] (reproduced with permission from Springer Nature).

Iteration	0	1	2
q_x	0.0872	0.0872	0.0872
q_y	0.1745	0.1748	0.1748
q_z	0.2612	0.2613	0.2614
e_x	0.0	0.0071	0.0127
e_y	0.0	0.1499	0.1515
e_z	0.0	0.3678	0.3807
E_{rgd}	0.3370	4.1210^{-04}	4.2910^{-07}

well as constrained optimization solver using Active Set algorithm implemented in MATLAB function called `fmincon`. A total of 1000 random orientation samples are chosen from the *physically* feasible workspace of

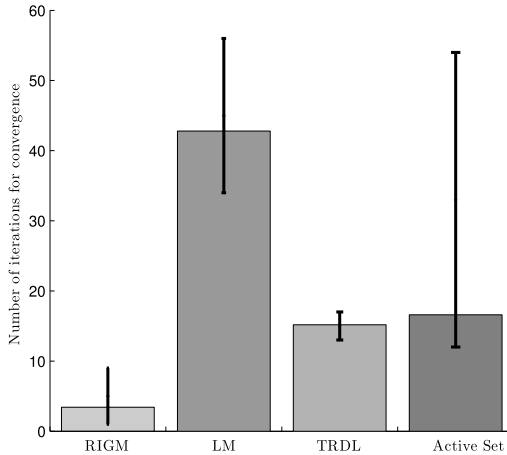


Figure 5.9 Comparison of number of iterations for convergence between different RIKM solution strategies. Original Source: [2] (reproduced with permission from Springer Nature).

the mechanism and provided as the input to this problem. RIKM solver demonstrates robust convergence inside the physically feasible workspace of the mechanism. The number of iterations for convergence and the CPU time⁷ of RIKM are recorded for benchmarking its efficiency in comparison to standard solvers for a tolerance of $\epsilon = 1.e^{-06}$ mm (see Fig. 5.9 and Fig. 5.10). Error bars used in the two figures are asymmetric and represent min.-max. and average values. With average iterations for convergence equal to 3.42 and CPU time equals to 2.58 milliseconds, it was found that RIKM performed 21 times faster than TRDL-based solver.

Discussion

The computation scheme of the novel RIKM algorithm—that solves the problem of coupled motion kinematics in context of the ACTIVE ANKLE efficiently, as displayed in diagram in Fig. 5.11. From that scheme, it can be observed that the auxiliary variables $\tilde{\mathbf{l}}$ —that reflect violations of structural (rigidity) constraints—can be interpreted as virtual joints. The method RIKM ensures that at termination after a few iterations (see Table 5.6), the values of $\tilde{\mathbf{l}}$ equal zero. From the viewpoint of kinematic synthesis, this consideration opens a perspective for extending the ACTIVE ANKLE from an

⁷ Intel Core i7 CPU 950 @ 3.07 GHz x 8PC, 6GB RAM.

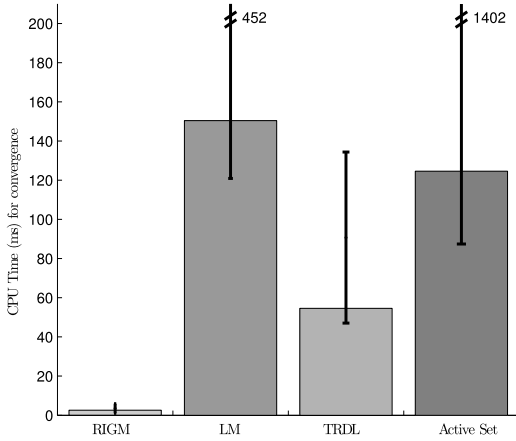


Figure 5.10 Comparison of CPU time for convergence between different RIKM solution strategies. Original Source: [2] (reproduced with permission from Springer Nature).

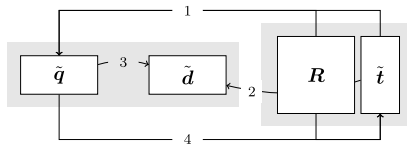


Figure 5.11 Computation scheme of the rotational inverse method RIKM. The matrix of workspace variables is the end-effector pose $\tilde{\mathbf{P}}_E \cong (\mathbf{R}, \tilde{\mathbf{t}})$. The vector of configuration variables $\tilde{\mathbf{q}}$ is given by (q_x, q_y, q_z) . The vector of design variables, denoted by $\tilde{\mathbf{d}}$, includes the vector $\tilde{\mathbf{t}}$ that is checked for constraint violation in the abortion criterion of RIKM in Algorithm 5.9. The computation in RIKM consists of the steps (1) IKM*, (2) CEP, (3) CCP, and (4) TFKM*. Original Source: [2] (reproduced with permission from Springer Nature).

almost spherical design to a fully-controllable six-DOF mechanism, which in particular could also act as a perfect spherical mechanism (compare [28]).

5.5 Conclusion

This chapter presents a thorough study of the ACTIVE ANKLE, a novel parallel manipulator with mobility three that moves in an almost spherical manner. The type synthesis, mobility analysis and design considerations are

presented, unveiling its distinctive features and its suitability as a spherical joint module in various applications. Then, a thorough inverse kinematics analysis is then performed, deriving an analytical solution to the full inverse kinematic problem. Also, it is identified that the inverse kinematic problem is not sufficient for its kinematic control due to the almost spherical nature of the device. Subsequently, a rotative inverse kinematic problem is solved with a novel tailored iterative technique which exploits the geometric properties of this mechanism. The solution to the rotative inverse kinematic problem enables the kinematic control of this mechanism in its spherical task space. The discussion of its forward kinematics can be found in [29].

References

- [1] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
- [2] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, *Journal of Intelligent & Robotic Systems* 94 (2019) 303–325, <https://doi.org/10.1007/s10846-018-0792-x>.
- [3] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the RECUPERA exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019), <https://doi.org/10.3390/app9040626>, <http://www.mdpi.com/2076-3417/9/4/626>.
- [4] C.M. Gosselin, E.S. Pierre, M. Gagne, On the development of the agile eye, *IEEE Robotics & Automation Magazine* 3 (4) (1996) 29–37, <https://doi.org/10.1109/100.556480>.
- [5] A. Niyetkaliyev, A. Shintemirov, An approach for obtaining unique kinematic solutions of a spherical parallel manipulator, in: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2014, pp. 1355–1360.
- [6] J. Gallardo-Alvarado, *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*, 1st edition, Springer, 2016.
- [7] P. Vischer, R. Clavel, Argos: a novel 3-DoF parallel wrist mechanism, *The International Journal of Robotics Research* 19 (1) (2000) 5–11, <https://doi.org/10.1177/02783640022066707>.
- [8] K. Al-Widyan, X.Q. Ma, J. Angeles, The robust design of parallel spherical robots, *Mechanism and Machine Theory* 46 (3) (2011) 335–343.
- [9] M. Simnofske, Ausrichtungsvorrichtung zum Ausrichten einer Plattform in drei rotatorischen Freiheiten, Patent application, DE102013018034A1, 2015.
- [10] M. Simnofske, S. Kumar, B. Bongardt, F. Kirchner, Active ankle – an almost-spherical parallel mechanism, in: *47th International Symposium on Robotics (ISR)*, 2016.
- [11] M. Frindt, *Modulbasierte Synthese von Parallelstrukturen für Maschinen in der Produktionstechnik*, Ph.D. thesis, Technical University of Braunschweig, 2001.
- [12] A.B. Kempe, How to Draw a Straight Line, 1877.
- [13] R.K.G. Temple, The Cardan suspension, in: *The UNESCO Courier*, 1988.
- [14] G. Wu, S. Caro, J. Wang, Design and transmission analysis of an asymmetrical spherical parallel manipulator, *Mechanism and Machine Theory* 94 (2015) 119–131.
- [15] T. Villgratner, E. Schneider, P. Andersch, H. Ulbrich, Compact high dynamic 3 DoF camera orientation system: development and control, *Journal of System Design and Dynamics* 5 (5) (2011) 819–828.

- [16] M. Valasek, J. Zicha, M. Karasek, R. Hudec, Hexasphere – redundantly actuated parallel spherical mechanism as a new concept of agile telescope, *Advances in Astronomy* 2010 (2010) 348286.
- [17] E.A. Kirchner, N. Will, M. Simnofske, L.M.V. Benitez, B. Bongardt, M.M. Krell, S. Kumar, M. Mallwitz, A. Seeland, M. Tabie, H. Woehrl, M. Yueksel, A. Hess, R. Buschfort, F. Kirchner, Recupera-Reha: exoskeleton technology with integrated biosignal analysis for sensorimotor rehabilitation, in: *Transdisziplinäre Konferenz SmartASSIST*, 2016, pp. 504–517.
- [18] T. Nef, M. Guidali, V. Klamroth-Marganska, R. Riener, ARMin – exoskeleton robot for stroke rehabilitation, in: O. Dössel, W.C. Schlegel (Eds.), *World Congress on Medical Physics and Biomedical Engineering*, September 7–12, 2009, Munich, Germany, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 127–130.
- [19] S. Kumar, M. Simnofske, B. Bongardt, A. Müller, F. Kirchner, Integrating mimic joints into dynamics algorithms: exemplified by the hybrid Recupera exoskeleton, in: *Proceedings of the Advances in Robotics, AIR '17*, ACM, New York, NY, USA, 2017, pp. 27:1–27:6, <https://doi.org/10.1145/3132446.3134891>, <http://doi.acm.org/10.1145/3132446.3134891>.
- [20] J. Huang, X. Tu, J. He, Design and evaluation of the RUPERT wearable upper extremity exoskeleton robot for clinical and in-home therapies, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46 (7) (2016) 926–935, <https://doi.org/10.1109/TSMC.2015.2497205>.
- [21] R.A.R.C. Gopura, K. Kiguchi, D.S.V. Bandara, A brief review on upper extremity robotic exoskeleton systems, in: *2011 6th International Conference on Industrial and Information Systems*, 2011, pp. 346–351, <https://doi.org/10.1109/ICIINFS.2011.6038092>.
- [22] H. Herr, Exoskeletons and orthoses: classification, design challenges and future directions, *Journal of NeuroEngineering and Rehabilitation* 6 (1) (2009) 21.
- [23] A.B. Zoss, H. Kazerooni, A. Chu, Biomechanical design of the Berkeley lower extremity exoskeleton (BLEEX), *IEEE/ASME Transactions on Mechatronics* 11 (2) (2006) 128–138.
- [24] S. Yamamoto, A. Hagiwara, T. Mizobe, O. Yokoyama, T. Yasui, Development of an ankle-foot orthosis with an oil damper, *Prosthetics and Orthotics International* 29 (3) (2005).
- [25] G.S. Sawicki, D.P. Ferris, A pneumatically powered knee-ankle-foot orthosis (KAFO) with myoelectric activation and inhibition, *Journal of NeuroEngineering and Rehabilitation* 6 (1) (2009) 23.
- [26] W. Khalil, E. Dombre, Chapter 4 - inverse geometric model of serial robots, in: *Modeling, Identification and Control of Robots*, Butterworth-Heinemann, Oxford, 2002, pp. 57–84, <https://doi.org/10.1016/B978-190399666-9/50004-X>, <https://www.sciencedirect.com/science/article/pii/B978190399666950004X>.
- [27] F. Thomas, L. Ros, Revisiting trilateration for robot localization, *IEEE Transactions on Robotics* 21 (1) (2005) 93–101.
- [28] L. Luzzi, N. Sancisi, V. Parenti Castelli, A new direct position analysis solution for an over-constrained Gough-Stewart platform, in: S. Zeghloul, L. Romdhane, M.A. Laribi (Eds.), *Computational Kinematics*, Springer International Publishing, Cham, 2018, pp. 585–592.
- [29] S. Kumar, A. Nayak, B. Bongardt, A. Mueller, F. Kirchner, Kinematic analysis of active ankle using computational algebraic geometry, in: S. Zeghloul, L. Romdhane, M.A. Laribi (Eds.), *Computational Kinematics*, Springer International Publishing, Cham, 2018, pp. 117–125.

PART 3

Kinematics, dynamics, and control

This page intentionally left blank

CHAPTER 6

Kinematics and dynamics of tree type systems

Shivesh Kumar^a and Andreas Müller^b

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

This chapter provides the preliminaries for the Kinematics and Dynamics part of the book by visiting the screw theory and Lie group methods for modeling the kinematics and dynamics of tree type systems. The content presented here is largely based on [1] and is included for the completeness of the book. For a more detailed treatment, the readers are referred to [2–6]. The reader with some background in this area may consider skipping this chapter. It starts with graph-based topological description of tree type robotic systems in Section 6.1 and then in Section 6.2 presents the recursive relations for computing the kinematics of different bodies in the system. Section 6.3 presents the equations of motion to describe the dynamics of a single rigid body. Section 6.4 presents the recursive Newton–Euler algorithm for computing the spatial velocities and wrenches for the robot in order to provide the solution to the inverse dynamics problem. Section 6.5 presents the equations of motion in closed form.

6.1 Graph based topological description

The first step in developing the equations of motion is to start with a system model which describes the existence of various components and how they are connected. This is also referred to as topological description of the multi-body system. The topology of a system is best described with the help of graphs [5,7,8]. The topological or connectivity graph \mathcal{G} has the following properties:

1. The nodes represent bodies and the edges represent joints.
2. The graph is directed, i.e., edge is directed from the parent node to its child.
3. The graph is connected, i.e., a path exists between any two nodes.

4. Exactly one body represents fixed body, which is also known as the root link or base.

Graphs can be used to describe the kinematic topology of any kind of mechanical system, including free-floating systems, closed-loop mechanisms, etc. However, when there is no closed loop in the system, the graph becomes a tree \mathcal{T} , i.e., there exists only one path between any two nodes in the graph.

6.1.1 Numbering scheme for topological graphs

A convenient way to identify bodies and joints in the topological graph is to number them. A well-suited numbering scheme suitable for multi-body dynamics algorithms based on [5] called the *regular numbering* is presented here. According to this scheme, the bodies are numbered from 0 to N_B and the joints from 1 to N_J , and respecting the following rules:

1. Choose a spanning tree \mathcal{T} .
2. Number 0 is assigned to the fixed root link.
3. Number the remaining nodes from 1 to N_B in any order such that each node has a higher number than its parent in \mathcal{T} .
4. Number the edges in \mathcal{T} from 1 to N_B such that the edge i connects between node i and its parent.
5. Number all the remaining edges from $N_B + 1$ to N_J in any order.
6. Each body gets the same number as its node, each joint gets the same number as its edge.

This scheme is only unique for a serial kinematic chain, in all other cases, it is non-unique. Fig. 6.1 shows the examples of regular numbering scheme applied to a serial and tree type mechanism. It is clear from Fig. 6.1a that the numbering scheme is unique. However, it can be noticed in Fig. 6.1b that the numbering scheme is non-unique at the branching, i.e., both left and right branch could start from 2 (in the figure it is the left branch).

6.1.2 Representation of topological graphs

It is useful to derive a certain representation for topological graphs that can provide information about connectivity in the graph. One of the simplest ways to this is to record the predecessor and successor body numbers in a pair of arrays p and s such that their elements $p(i)$ and $s(i)$ are the predecessor and successor body numbers of joint i . Together, p and s provide a complete description of the topological graph, from which many other useful quantities can be calculated.

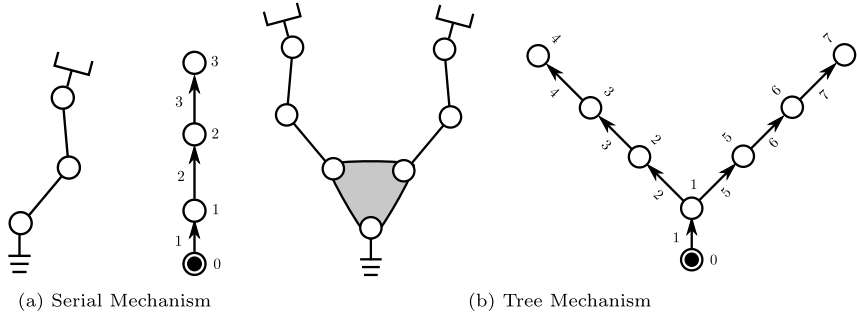


Figure 6.1 Examples for regular numbering scheme [1].

Table 6.1 Examples for topological graph representation and its properties.

Property	Serial mechanism in Fig. 6.1a	Tree mechanism in Fig. 6.1b
p	{0, 1, 2}	{0, 1, 2, 3, 1, 5, 6}
s	{1, 2, 3}	{1, 2, 3, 4, 5, 6, 7}
λ	{0, 1, 2}	{0, 1, 2, 3, 1, 5, 6}
μ	{{1}, {2}, {3}}	{{1}, {2, 5}, {3}, {4}, {}, {6}, {7}, {}}}
κ	{{1}, {1, 2}, {1, 2, 3}}	{{1}, {1, 2}, {1, 2, 3}, {1, 2, 3, 4}, {1, 5}, {1, 5, 6}, {1, 5, 6, 7}}

- **Parent array** λ identifies the parent of each body in the spanning tree. According to rules 3 and 4 of the regular numbering scheme, the tree joint i connects the two bodies i and $\lambda(i)$ such that $\lambda(i) < i$, so

$$\lambda(i) = \min(p(i), s(i)), \quad 1 \leq i \leq N_B. \quad (6.1)$$

- **Child array** μ identifies the children of each body in the spanning tree.
- **Support array** κ : For any body i except base, $\kappa(i)$ is the set of all tree joints between body i and base.

Table 6.1 shows the representation of topological graphs and their properties for the serial and tree-type mechanism examples introduced in Fig. 6.1a and Fig. 6.1b.

6.2 Recursive kinematics computation

This section presents the formulations for the computation of position, velocity and acceleration of different bodies in a kinematic chain from

its topological description. These formulations are based on screw theory, which provides the geometric setting and Lie group theory, which forms the analytic foundation for an overall intuitive and efficient modeling of rigid body mechanisms. An extensive mathematical introduction to Lie groups and screw theory can be found in [2–4,6]. To keep the description simple, we restrict our attention to unbranched kinematic chains or serial robots.

Notation

In this book, bold letters denote vectors and matrices. However, in the remainder of this chapter, whenever there is a chance of ambiguity, $[\]$ denotes the matrix form of the variable (similar to the notation in [4]). For example, a screw represented by $\mathbf{S} = [\boldsymbol{\omega}, \boldsymbol{\nu}]^T \in \mathbb{R}^6$, when represented as an element of $se(3)$ is given by:

$$[\mathbf{S}] = \begin{bmatrix} [\boldsymbol{\omega}] & \boldsymbol{\nu} \\ \mathbf{0} & 0 \end{bmatrix}. \quad (6.2)$$

In above equation, again the angular velocity vector $\boldsymbol{\omega} \in \mathbb{R}^3$ when represented as an element of $so(3)$, is given by:

$$[\boldsymbol{\omega}] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \quad (6.3)$$

6.2.1 Position of a kinematic chain

The matrix exponential mapping, defined as $\exp : [\mathbf{S}]q \in se(3) \rightarrow \mathbf{T} \in SE(3)$, can be used to compute the configuration of a rigid body \mathbf{T} from the description of its screw axis \mathbf{S} and screw coordinate q . The forward kinematics of a kinematic chain can be computed recursively exploiting the matrix exponential mapping using the product of exponential (POE) formula [9]. The key concept behind it is to regard each joint as applying screw motion to all the outward links.

Definition 6.1 (Product of Exponentials). Given the zero configuration of the end-effector of the kinematic chain ${}^0\mathbf{T}_n(\mathbf{0}) \in SE(3)$ in base frame, its forward kinematics $\mathbf{q} \in \mathcal{Q} \mapsto {}^0\mathbf{T}_n(\mathbf{q}) \in SE(3)$ can be computed using the **product of exponentials** formula as:

$${}^0\mathbf{T}_n(\mathbf{q}) = \exp([\mathbf{S}_1]q_1) \exp([\mathbf{S}_2]q_2) \dots \exp([\mathbf{S}_n]q_n) {}^0\mathbf{T}_n(\mathbf{0}), \quad (6.4)$$

where \mathbf{S}_i represents the screw coordinates of the i th joint expressed in the base frame (also known as spatial representation) and q_i is the respective joint displacement relative to the zero configuration.

The POE formula is not minimal, i.e., it requires $6n$ scalars to describe n screw axes in addition to n joint coordinate values. However, the advantage of this formula is that it does not require the definition of any link frames unlike the Denavit–Hartenberg convention [10] and geometric data can be extracted easily from any CAD software. Further, it is very simple to compute the inverse of the homogeneous transformation matrix¹ representing the end effector transformation using:

$${}^0T_n^{-1}(\mathbf{q}) = {}^0T_n^{-1}(\mathbf{0}) \exp(-[\mathbf{S}_n]q_n) \dots \exp(-[\mathbf{S}_2]q_2) \exp(-[\mathbf{S}_1]q_1). \quad (6.5)$$

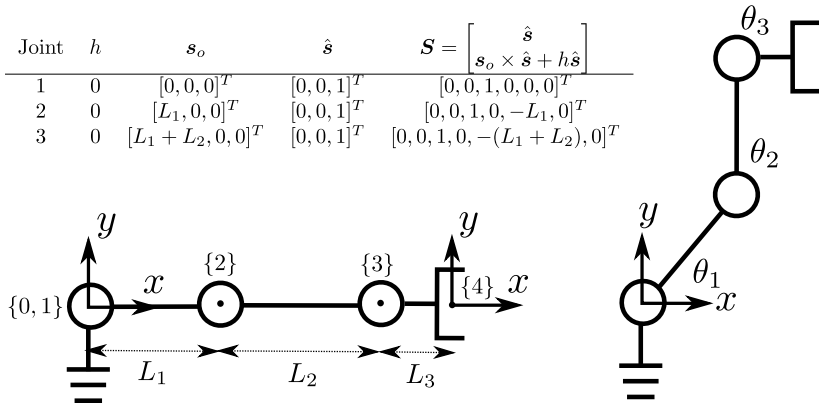


Figure 6.2 Screw description for an unbranched kinematic chain [1].

Example 6.1. Consider the unbranched kinematic chain shown in Fig. 6.2. The zero-pose configuration of the end effector in the base frame is given by:

$${}^0T_4(\mathbf{0}) = \begin{bmatrix} 1 & 0 & 0 & (L_1 + L_2 + L_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6.6)$$

¹ Recall that the inverse of the matrix exponential function is given by $(\exp(\mathbf{A}\theta))^{-1} = \exp(-\mathbf{A}\theta)$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$.

Using Eq. (6.2), the screw axes in matrix screw representation $[\mathbf{S}] \in se(3)$ are given by:

$$\begin{aligned} [\mathbf{S}_1] &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & [\mathbf{S}_2] &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -L_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\ & & [\mathbf{S}_3] &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (6.7)$$

The forward kinematics of the mechanism is then given by

$${}^0\mathbf{T}_4(\mathbf{q}) = \exp([\mathbf{S}_1]\theta_1) \exp([\mathbf{S}_2]\theta_2) \exp([\mathbf{S}_3]\theta_3) {}^0\mathbf{T}_4(\mathbf{0}). \quad (6.8)$$

For $\mathbf{q} = [\pi/4, \pi/4, -\pi/2]$, the forward kinematics of the robot is given by $\mathbf{X} = [x, y, \phi] = [L_1/\sqrt{2} + L_3, L_1/\sqrt{2} + L_2, 0]$, which represents the right configuration of Fig. 6.2.

6.2.2 Velocity of a kinematic chain

The POE formula² (Eq. (6.4)), which is used to describe the position of a kinematic chain, can be differentiated with respect to time to establish a relationship between joint velocity $\dot{\mathbf{q}}$ and end-effector's spatial twist \mathbf{V} . The subscript and superscript are dropped in Eq. (6.4) to simplify the notation.

$$\begin{aligned} \mathbf{T}(\mathbf{q}) &= \exp([\mathbf{S}_1]q_1) \exp([\mathbf{S}_2]q_2) \dots \exp([\mathbf{S}_n]q_n) \mathbf{T}(\mathbf{0}) \\ \dot{\mathbf{T}} &= \left(\frac{d(\exp([\mathbf{S}_1]q_1))}{dt} \exp([\mathbf{S}_2]q_2) \dots \exp([\mathbf{S}_n]q_n) \mathbf{T}(\mathbf{0}) \right) + \\ &\quad \left(\exp([\mathbf{S}_1]q_1) \frac{d(\exp([\mathbf{S}_2]q_2))}{dt} \dots \exp([\mathbf{S}_n]q_n) \mathbf{T}(\mathbf{0}) \right) + \dots \end{aligned}$$

The spatial twist \mathbf{V} is given by $[\mathbf{V}] = \dot{\mathbf{T}}\mathbf{T}^{-1} \in se(3)$. Using Eq. (6.5), one could compute the spatial twist $[\mathbf{V}]$ as:

² Recall that the derivative of a matrix exponential function is given by $\frac{d(\exp(\mathbf{A}\theta))}{dt} = \mathbf{A} \exp(\mathbf{A}\theta) \dot{\theta} = \exp(\mathbf{A}\theta) \mathbf{A} \dot{\theta}$ where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a constant matrix and $\theta(t)$ is a scalar function of t .

$$[V] = \exp([\mathbf{S}_1]q_1)[\mathbf{S}_1]\exp(-[\mathbf{S}_1]q_1)\dot{q}_1 + \exp([\mathbf{S}_1]q_1)\exp([\mathbf{S}_2]q_2)[\mathbf{S}_2]\exp(-[\mathbf{S}_2]q_2)\exp(-[\mathbf{S}_1]q_1)\dot{q}_2 + \dots$$

The above can be expressed in vector form $V \in \mathbb{R}^6$ with the help of adjoint mapping:

$$V = \underbrace{\mathbf{Ad}_{\exp([\mathbf{S}_1]q_1)}(\mathbf{S}_1)}_{J_1} \dot{q}_1 + \underbrace{\mathbf{Ad}_{\exp([\mathbf{S}_1]q_1)\exp([\mathbf{S}_2]q_2)}(\mathbf{S}_2)}_{J_2} \dot{q}_2 + \dots, \quad (6.9)$$

where $J_i = \mathbf{Ad}_{T_i}(\mathbf{S}_i)$ with $T_i = \exp([\mathbf{S}_1]q_1) \dots \exp([\mathbf{S}_i]q_i)$ is the instantaneous screw coordinate vector of the i th joint.³ It can be observed that the above twist is a sum of n spatial twists and can be written in the following matrix form:

$$V = \begin{bmatrix} J_1 & J_2 & \dots & J_n \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J(q)\dot{q}, \quad (6.10)$$

where J is the spatial Jacobian of the kinematic chain with dimension $n \times 6$.

Recursive nature

Inspecting Eq. (6.10), the velocity of any body i in the kinematic chain can be expressed in the following summand form.

$$V_i = \sum_{j \leq i} \mathbf{Ad}_{T_j}(\mathbf{S}_j)\dot{q}_j \quad (6.11)$$

This also reveals the recursive nature of the velocity computation, i.e., the velocity of any body i can be expressed as a sum of the velocity of previous body $i - 1$ and velocity across the joint \dot{q}_i .

$$V_i = V_{i-1} + \mathbf{Ad}_{T_i}(\mathbf{S}_i)\dot{q}_i \quad (6.12)$$

³ An equivalent formula for computing instantaneous screw coordinates is

$$J_i = \begin{cases} \mathbf{S}_1, & i = 1 \\ \mathbf{Ad}_{T_{i-1}}(\mathbf{S}_i), & i \geq 2 \end{cases},$$

which arises when we use $\frac{d(\exp(A\theta))}{dt} = A \exp(A\theta)\dot{\theta}$ instead of $\exp(A\theta)A\dot{\theta}$. For details, see the derivation in [4,6].

The recursive nature of Eq. (6.12) is also exploited in the dynamics algorithms.

Geometric construction

The spatial twist of the end effector can be geometrically constructed using the *instantaneous* screw coordinates expressed in the base frame. It is to be noted that they are configuration-dependent, i.e., a function of \mathbf{q} and are equal to the \mathbf{S}_i only in the zero configuration.

$$\mathbf{V}_n = \begin{bmatrix} \hat{\mathbf{s}}_1 \\ \mathbf{s}_{o1} \times \hat{\mathbf{s}}_1 + h_1 \hat{\mathbf{s}}_1 \end{bmatrix} \dot{q}_1 + \begin{bmatrix} \hat{\mathbf{s}}_2 \\ \mathbf{s}_{o2} \times \hat{\mathbf{s}}_2 + h_2 \hat{\mathbf{s}}_2 \end{bmatrix} \dot{q}_2 + \dots + \begin{bmatrix} \hat{\mathbf{s}}_n \\ \mathbf{s}_{on} \times \hat{\mathbf{s}}_n + h_n \hat{\mathbf{s}}_n \end{bmatrix} \dot{q}_n \quad (6.13)$$

Example 6.1 (Continued). Consider the unbranched kinematic chain shown in Fig. 6.2 again. The spatial Jacobian of the kinematic chain in the zero configuration can be built by vertical concatenation of the joint screws.

$$\mathbf{J}(\mathbf{0}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & -L_1 & -(L_1 + L_2) \\ 0 & 0 & 0 \end{bmatrix}_{3 \times 6} \quad (6.14)$$

As noted above, the spatial Jacobian is configuration-dependent, i.e., the instantaneous joint screws as a function of \mathbf{q} needs to be derived (as opposed to zero configuration joint screw description from Fig. 6.2). These are provided in Table 6.2. Thus, the expression of the spatial Jacobians in any configuration \mathbf{q} is the following.

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & L_1 \sin q_1 & L_1 \sin q_1 + L_2 \sin(q_1 + q_2) \\ 0 & -L_1 \cos q_1 & -L_1 \cos q_1 - L_2 \cos(q_1 + q_2) \\ 0 & 0 & 0 \end{bmatrix}_{3 \times 6} \quad (6.15)$$

Table 6.2 Instantaneous Screw Description $\mathcal{S}(q)$. $c_1 = \cos q_1, s_1 = \sin q_1, c_{12} = \cos(q_1 + q_2), s_{12} = \sin(q_1 + q_2)$.

Joint (i)	s_{oi}	s_i	$S_i = \begin{bmatrix} \hat{s}_i \\ s_{oi} \times \hat{s}_i + h_i \hat{s}_i \end{bmatrix}$
1	$[0, 0, 0]^T$	$[0, 0, 0]^T$	$[0, 0, 1, 0, 0, 0]^T$
2	$[L_1 c_1 s_1, 0]^T$	$[0, 0, 0]^T$	$[0, 0, 1, L_1 s_1, -L_1 c_1, 0]^T$
3	$[L_1 c_1 + L_2 c_{12}, L_1 s_1 + L_2 s_{12}, 0]^T$	$[0, 0, 0]^T$	$[0, 0, 1, L_1 s_1 + L_2 s_{12}, -L_1 c_1 - L_2 c_{12}, 0]^T$

System level composition

The spatial twists of every body can be summarized in overall spatial twist vector $V^{\text{sys}} = [V_1, \dots, V_n]^T \in \mathbb{R}^{6n}$ and can be computed as

$$V^{\text{sys}} = J^{\text{sys}} \dot{q}, \quad (6.16)$$

where $J^{\text{sys}} \in \mathbb{R}^{6n \times n}$ is the *spatial system Jacobian*. The spatial system Jacobian can be factorized and written as

$$J^{\text{sys}} = A^{\text{sys}} S^{\text{sys}}, \quad (6.17)$$

where $S^{\text{sys}} = \text{diag}(S_1, \dots, S_n)$ is a block diagonal matrix and A^{sys} is a lower triangular matrix with the following expressions:

$$A^{\text{sys}} = \begin{bmatrix} [\mathbf{Ad}_{T_1}] & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ [\mathbf{Ad}_{T_1}] & [\mathbf{Ad}_{T_2}] & \mathbf{0} & \mathbf{0} \\ [\mathbf{Ad}_{T_1}] & [\mathbf{Ad}_{T_2}] & [\mathbf{Ad}_{T_3}] & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ [\mathbf{Ad}_{T_1}] & [\mathbf{Ad}_{T_2}] & [\mathbf{Ad}_{T_3}] & [\mathbf{Ad}_{T_n}] \end{bmatrix}_{6n \times 6n}, \quad (6.18)$$

$$S^{\text{sys}} = \begin{bmatrix} S_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & S_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & S_3 & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & S_n \end{bmatrix}_{6n \times n}$$

where $[\mathbf{Ad}_{T_i}]$ is a 6×6 adjoint matrix for the screw transformation of the i th joint (also called as *Plücker transformation matrix* in [5]).

6.2.3 Acceleration of a kinematic chain

The benefit of POE formula is not only an easy computation of the velocity of a kinematic chain, but it also facilitates the recursive computation of any higher-order derivatives. Here, the acceleration computation of the kinematic chain will be discussed. Differentiating Eq. (6.9) with respect to time, one gets:

$$\begin{aligned} \dot{\mathbf{V}} = & \underbrace{\mathbf{Ad}_{\exp(\{\mathbf{S}_1\}_{q_1})}(\mathbf{S}_1)}_{\mathbf{J}_1} \ddot{q}_1 + \underbrace{\frac{d}{dt} \mathbf{Ad}_{\exp(\{\mathbf{S}_1\}_{q_1})}(\mathbf{S}_1)}_{\frac{d}{dt} \mathbf{J}_1} \dot{q}_1 + \\ & \underbrace{\mathbf{Ad}_{\exp(\{\mathbf{S}_1\}_{q_1}) \exp(\{\mathbf{S}_2\}_{q_2})}(\mathbf{S}_2)}_{\mathbf{J}_2} \ddot{q}_2 + \underbrace{\frac{d}{dt} \mathbf{Ad}_{\exp(\{\mathbf{S}_1\}_{q_1}) \exp(\{\mathbf{S}_2\}_{q_2})}(\mathbf{S}_2)}_{\frac{d}{dt} \mathbf{J}_2} \dot{q}_2 + \dots, \end{aligned} \quad (6.19)$$

which could be written in matrix form as (also equivalent to time differentiation of Eq. (6.10)):

$$\begin{aligned} \dot{\mathbf{V}} = & \begin{bmatrix} \mathbf{J}_1 & \mathbf{J}_2 & \dots & \mathbf{J}_n \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_n \end{bmatrix} + \begin{bmatrix} \frac{d}{dt} \mathbf{J}_1 & \frac{d}{dt} \mathbf{J}_2 & \dots & \frac{d}{dt} \mathbf{J}_n \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} \\ & \dot{\mathbf{V}} = \mathbf{J} \ddot{\mathbf{q}} + \dot{\mathbf{J}} \dot{\mathbf{q}}, \end{aligned} \quad (6.20)$$

where $\frac{d\mathbf{J}_i}{dt} = \dot{\mathbf{J}}_i$ is the time derivative of instantaneous screw coordinates and $\dot{\mathbf{J}}$ is the time derivative of the Jacobian matrix of the kinematic chain. It should be recalled that columns of \mathbf{J} are configuration-dependent and hence an implicit function of time. Hence, it has something to do with the tangential aspect of frame transformation of the screw coordinates.

$$\frac{d\mathbf{J}_i}{dt} = \frac{d}{dt} \mathbf{Ad}_{\exp(\{\mathbf{S}_1\}_{q_1}) \exp(\{\mathbf{S}_2\}_{q_2}) \dots \exp(\{\mathbf{S}_i\}_{q_i})}(\mathbf{S}_i). \quad (6.21)$$

One useful tool in deriving the expression for $\dot{\mathbf{J}}$ is Lie bracket. However, in following, we define it in the form that is relevant to formulations for rigid body kinematics.

Definition 6.2 (Lie Bracket). Given two twists $\mathbf{V}_1 = (\boldsymbol{\omega}_1, \mathbf{v}_1)$ and $\mathbf{V}_2 = (\boldsymbol{\omega}_2, \mathbf{v}_2)$, the **Lie Bracket** $[\mathbf{V}_1, \mathbf{V}_2]$ of \mathbf{V}_1 and \mathbf{V}_2 written either as $[\mathbf{ad}_{\mathbf{V}_1}] \mathbf{V}_2$ in matrix form or $\mathbf{ad}_{\mathbf{V}_1}(\mathbf{V}_2)$ in the form of a mapping, is defined

as

$$\begin{bmatrix} [\boldsymbol{\omega}_1] & \mathbf{0} \\ [\mathbf{v}_1] & [\boldsymbol{\omega}_1] \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_2 \\ \mathbf{v}_2 \end{bmatrix} = [\mathbf{ad}_{V_1}] V_2 = \mathbf{ad}_{V_1}(V_2) \in \mathbb{R}^{6 \times 6} \quad (6.22)$$

where

$$[\mathbf{ad}_V] = \begin{bmatrix} [\boldsymbol{\omega}] & \mathbf{0} \\ [\mathbf{v}] & [\boldsymbol{\omega}] \end{bmatrix} \quad (6.23)$$

The Lie Bracket is also called as *adjoint mapping*, *screw product*, or *spatial cross product*. The result of the Lie Bracket $[V_1, V_2]$ in vector and matrix notations are given by:

$$[V_1, V_2] = \begin{bmatrix} \boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2 \\ \mathbf{v}_1 \times \boldsymbol{\omega}_2 + \boldsymbol{\omega}_1 \times \mathbf{v}_2 \end{bmatrix} \in \mathbb{R}^6 = \begin{bmatrix} [\boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2] & \mathbf{v}_1 \times \boldsymbol{\omega}_2 + \boldsymbol{\omega}_1 \times \mathbf{v}_2 \\ \mathbf{0} & 0 \end{bmatrix}. \quad (6.24)$$

In fact, it can be shown that the i th column of the matrix \mathbf{J} is simply given by the expression $\dot{\mathbf{J}}_i = [V_i, J_i]$ also sometimes written as $V_i \times J_i$ (in the spatial cross product notation from [5]).

Recursive nature

Inspecting Eq. (6.20), the acceleration of any body i in the kinematic chain can be expressed in the following summand form:

$$\dot{V}_i = \sum_{j \leq i} (J_j \ddot{q}_j + [V_j, J_j] \dot{q}_j), \quad (6.25)$$

which again reveals the recursive nature of the acceleration computation, i.e., the acceleration of any body i can be expressed as a sum of the acceleration of previous body $i - 1$ and acceleration across the joint \ddot{q}_i .

$$\dot{V}_i = \dot{V}_{i-1} + J_i \ddot{q}_i + [V_i, J_i] \dot{q}_i \quad (6.26)$$

Substituting Eq. (6.12) in the above equation, and utilizing the bilinearity and anticommutative property of Lie Brackets, one could derive a further simplified recursive equation.

$$\begin{aligned} \dot{V}_i &= \dot{V}_{i-1} + J_i \ddot{q}_i + [V_{i-1}, V_i] \\ \text{or } \dot{V}_i &= \dot{V}_{i-1} + \mathbf{Ad}_{T_i}(\mathbf{S}_i) \ddot{q}_i + \mathbf{ad}_{V_{i-1}}(V_i). \end{aligned} \quad (6.27)$$

System level composition

The spatial acceleration of every body can be summarized in overall spatial acceleration vector $\dot{\mathbf{V}}^{\text{sys}} = [\dot{\mathbf{V}}_1, \dots, \dot{\mathbf{V}}_n]^T \in \mathbb{R}^{6n}$ and can be computed as

$$\dot{\mathbf{V}}^{\text{sys}} = \mathbf{J}^{\text{sys}} \ddot{\mathbf{q}} + \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} \mathbf{J}^{\text{sys}} \dot{\mathbf{q}} = \mathbf{J}^{\text{sys}} \ddot{\mathbf{q}} + \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} \mathbf{V}^{\text{sys}}, \quad (6.28)$$

with $\mathbf{b}^{\text{sys}} = \text{diag}(\mathbf{ad}_{V_1}, \mathbf{ad}_{V_2}, \dots, \mathbf{ad}_{V_n})$ being a diagonal matrix and \mathbf{L}^{sys} being the lower triangular block unit matrix [3] with the following expressions:

$$\mathbf{L}^{\text{sys}} = \begin{bmatrix} \mathbf{I}_{6 \times 6} & \mathbf{0}_{6 \times 6} & \mathbf{0}_{6 \times 6} & \mathbf{0}_{6 \times 6} \\ \mathbf{I}_{6 \times 6} & \mathbf{I}_{6 \times 6} & \mathbf{0}_{6 \times 6} & \mathbf{0}_{6 \times 6} \\ \mathbf{I}_{6 \times 6} & \mathbf{I}_{6 \times 6} & \mathbf{I}_{6 \times 6} & \mathbf{0}_{6 \times 6} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{I}_{6 \times 6} & \mathbf{I}_{6 \times 6} & \mathbf{I}_{6 \times 6} & \mathbf{I}_{6 \times 6} \end{bmatrix}_{6n \times 6n}, \quad (6.29)$$

$$\mathbf{b}^{\text{sys}} = \begin{bmatrix} \mathbf{ad}_{V_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{ad}_{V_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{ad}_{V_3} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{ad}_{V_n} \end{bmatrix}_{6n \times 6n}.$$

6.3 Dynamics of a single rigid body

This section presents the Newton–Euler equations for describing the dynamics of a single rigid body motion in $SE(3)$. In this regard, first preliminary physical concepts necessary to describe the dynamics of a rigid body are briefly introduced. To motivate the advantages of the screw theory and Lie group based approach to robot dynamics, we start with the classical formulation of dynamics of a single rigid body. Then, a twist–wrench based formulation exploiting the Lie group theory is presented.

6.3.1 Physical properties of a rigid body

Rigid body is defined as an object in which the distance between any two given points remains constant in time regardless of external forces exerted on it. A rigid body can be seen as a set of point masses with fixed distances between each other. There are three important physical properties of a rigid body that must be taken into account to describe its dynamics.

- **Mass.** The mass of a physical body is the measure of its resistance to a change in state of its motion (acceleration). If we think of a rigid body as a composition of rigidly connected point masses, then the total mass of the rigid body is given by $m = \sum m_i$, where m_i is the mass of i th point mass.
- **Center of mass (COM).** The center of mass \mathbf{c} of a rigid body in space is the unique point where the weighted relative position of the distributed mass sums to zero. Formally speaking, the center of mass is the location of a point such that $\sum m_i \mathbf{r}_i = 0$, where $\mathbf{r}_i = (x_i, y_i, z_i)$ is the location of a point mass m_i with respect to the COM. It can also be interpreted as the first moment of mass.
- **Moment of Inertia or Rotational Inertia.** The moment of inertia of a physical body is the measure of its resistance to a change in rotational motion (angular acceleration). It can also be interpreted as the second moment of mass and is given by

$$\mathbf{I}_b = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} = \begin{bmatrix} \sum m_i (y_i^2 + z_i^2) & -\sum m_i x_i y_i & -\sum m_i x_i z_i \\ -\sum m_i x_i y_i & \sum m_i (x_i^2 + z_i^2) & -\sum m_i y_i z_i \\ -\sum m_i x_i z_i & -\sum m_i y_i z_i & \sum m_i (x_i^2 + y_i^2) \end{bmatrix}.$$

The summations can be replaced by volume integrals over the body b using the differential volume element dV , with point masses m_i replaced by a mass density function $\rho(x, y, z)$. In either case of these cases, \mathbf{I}_B is *symmetric* and *positive-definite* for any rigid body.

All of these properties are intrinsic to the rigid body itself and do not change under the influence of external forces or time. Overall, one needs a set of 10 real-valued parameters to describe the mass-inertial properties of a rigid body.

$$\Phi = [m, c_x, c_y, c_z, I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{yz}, I_{xz}] \in \mathbb{R}^{10} \quad (6.30)$$

6.3.2 Classical formulation

Consider a single rigid body of mass m consisting of a number of rigidly connected point masses m_i . Assume that this body is moving with a linear velocity \mathbf{v}_b and angular velocity $\boldsymbol{\omega}_b$, and let $\mathbf{p}_i(t)$ denote the time-varying position of m_i , initially located \mathbf{r}_i in the inertial frame $\{b\}$. Then, the velocity

and acceleration of any point on this rigid body can be described as

$$\begin{aligned}\dot{\mathbf{p}}_i &= \mathbf{v}_b + \boldsymbol{\omega}_b \times \mathbf{p}_i \\ \ddot{\mathbf{p}}_i &= \dot{\mathbf{v}}_b + \dot{\boldsymbol{\omega}}_b \times \mathbf{p}_i + \boldsymbol{\omega}_b \times (\mathbf{v}_b + \boldsymbol{\omega}_b \times \mathbf{p}_i),\end{aligned}\quad (6.31)$$

which can be written in the matrix form as

$$\ddot{\mathbf{p}}_i = \dot{\mathbf{v}}_b + [\dot{\boldsymbol{\omega}}_b]\mathbf{r}_i + [\boldsymbol{\omega}_b]\mathbf{v}_b + [\boldsymbol{\omega}_b]^2\mathbf{r}_i. \quad (6.32)$$

From Newton's 2nd law of motion, the force acting on this point is $\mathbf{f}_i = m_i\ddot{\mathbf{p}}_i$ and implies a moment $\mathbf{m}_i = \mathbf{r}_i \times \mathbf{f}_i = [\mathbf{r}_i]\mathbf{f}_i$. The total force acting on this body \mathbf{f}_b is given by

$$\mathbf{f}_b = \sum_i m_i(\dot{\mathbf{v}}_b + [\dot{\boldsymbol{\omega}}_b]\mathbf{r}_i + [\boldsymbol{\omega}_b]\mathbf{v}_b + [\boldsymbol{\omega}_b]^2\mathbf{r}_i), \quad (6.33)$$

which can be simplified if the body frame is assumed to coincide with the COM, i.e., $\sum m_i\mathbf{r}_i = 0$. The simplified expression for the total force is

$$\mathbf{f}_b = m(\dot{\mathbf{v}}_b + [\dot{\boldsymbol{\omega}}_b]\mathbf{v}_b). \quad (6.34)$$

Similarly, the total moment acting on the body \mathbf{m}_b is given by

$$\mathbf{m}_b = \sum_i m_i[\mathbf{r}_i](\dot{\mathbf{v}}_b + [\dot{\boldsymbol{\omega}}_b]\mathbf{r}_i + [\boldsymbol{\omega}_b]\mathbf{v}_b + [\boldsymbol{\omega}_b]^2\mathbf{r}_i), \quad (6.35)$$

which also gets simplified due to the above assumption to

$$\begin{aligned}\mathbf{m}_b &= \left(-\sum_i m_i[\mathbf{r}_i]^2\right)\dot{\boldsymbol{\omega}}_b + [\dot{\boldsymbol{\omega}}_b]\left(-\sum_i m_i[\mathbf{r}_i]^2\right)\boldsymbol{\omega}_b \\ &\text{or } \mathbf{m}_b = \mathbf{I}_b\dot{\boldsymbol{\omega}}_b + [\dot{\boldsymbol{\omega}}_b]\mathbf{I}_b\boldsymbol{\omega}_b.\end{aligned}\quad (6.36)$$

The above equation is also known as Euler's equation for a rotating body. Eq. (6.34) and Eq. (6.36) together constitute the Newton–Euler equations of motion for the single rigid body system and look fairly simple. However, if the same equations were expressed in any other frame, then they would be quite complicated (e.g., revisit Eq. (6.33) and Eq. (6.35)). It is not hard to imagine that this complexity manifests stronger in case of multibody dynamics, where multiple bodies are involved. The forces and moments acting on a specific link are typically expressed in different reference frames, and these must be expressed in terms of a common frame before they can be summed.

6.3.3 Twist-wrench formulation

Let us collect the angular velocity and linear velocity of the body in a body twist vector $\mathbf{V}_b = (\boldsymbol{\omega}_b, \mathbf{v}_b)$ and moment and force vectors in a body wrench vector $\mathbf{W}_b = (\mathbf{m}_b, \mathbf{f}_b)$. Eq. (6.34) and Eq. (6.36) can be collected and written in a combined form as

$$\begin{bmatrix} \mathbf{m}_b \\ \mathbf{f}_b \end{bmatrix} = \begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\mathbf{v}}_b \end{bmatrix} + \begin{bmatrix} [\boldsymbol{\omega}_b] & \mathbf{0} \\ \mathbf{0} & [\boldsymbol{\omega}_b] \end{bmatrix} \begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix}, \quad (6.37)$$

where \mathbf{I} is a 3×3 identity matrix. Exploiting the $[\mathbf{v}]\mathbf{v} = \mathbf{v} \times \mathbf{v} = \mathbf{0}$ and $[\mathbf{v}]^T = -[\mathbf{v}]$ properties of a skew-symmetric matrix, we can rewrite the above equation as:

$$\begin{aligned} \begin{bmatrix} \mathbf{m}_b \\ \mathbf{f}_b \end{bmatrix} &= \begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\mathbf{v}}_b \end{bmatrix} + \begin{bmatrix} [\boldsymbol{\omega}_b] & [\mathbf{v}_b] \\ \mathbf{0} & [\boldsymbol{\omega}_b] \end{bmatrix} \begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix} \\ \underbrace{\begin{bmatrix} \mathbf{m}_b \\ \mathbf{f}_b \end{bmatrix}}_{\mathbf{W}_b} &= \underbrace{\begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix}}_{\mathbf{M}_b} \underbrace{\begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\mathbf{v}}_b \end{bmatrix}}_{\dot{\mathbf{V}}_b} - \underbrace{\begin{bmatrix} [\boldsymbol{\omega}_b] & \mathbf{0} \\ [\mathbf{v}_b] & [\boldsymbol{\omega}_b] \end{bmatrix}^T}_{\text{ad}_{\mathbf{V}_b}^T} \underbrace{\begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix}}_{\mathbf{M}_b} \underbrace{\begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix}}_{\mathbf{V}_b} \end{aligned} \quad (6.38)$$

where each term can be identified as a spatial quantity. It shares a peculiar resemblance with Eq. (6.36) and could be seen as spatial version of this equation. In particular, we discuss the following two terms:

- **Spatial mass-inertia matrix** of a rigid body $\mathbf{M}_b \in \mathbb{R}^{6 \times 6}$ is defined as:

$$\mathbf{M}_b = \begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix} \quad (6.39)$$

and is symmetric and positive definite. Using it, the total kinetic energy of the rigid body can be written as

$$K = \frac{1}{2} \mathbf{V}_b^T \mathbf{M}_b \mathbf{V}_b = \frac{1}{2} \mathbf{v}_b^T m \mathbf{v}_b + \frac{1}{2} \boldsymbol{\omega}_b^T \mathbf{I}_b \boldsymbol{\omega}_b \in \mathbb{R}^+. \quad (6.40)$$

- **Spatial momentum** coscrew $\mathbf{P}_b = (\mathbf{L}_b, \mathbf{p}_b)^T \in \mathfrak{se}^*(3)$ composed of the angular momentum of the body \mathbf{L}_b and its linear momentum \mathbf{p}_b is defined as

$$\mathbf{P}_b = \begin{bmatrix} \mathbf{I}_b & \mathbf{0} \\ \mathbf{0} & m\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix} = \mathbf{M}_b \mathbf{V}_b \in \mathbb{R}^6. \quad (6.41)$$

Using the spatial momentum in body frame, one can compute the wrench acting on the body \mathbf{W}_b as:

$$\mathbf{W}_b = \dot{\mathbf{P}}_b - \left[\mathbf{ad}_{\mathbf{V}_b}^T \right] \mathbf{P}_b \quad (6.42)$$

Thus, the EOM can be written in the following compact form.

$$\mathbf{W}_b = \mathbf{M}_b \dot{\mathbf{V}}_b - \left[\mathbf{ad}_{\mathbf{V}_b}^T \right] \mathbf{M}_b \mathbf{V}_b \quad (6.43)$$

In contrast to two equations Eq. (6.34) and Eq. (6.36) describing the dynamics of the rigid body in the classical formulation, the modern geometric approach allows the description of the dynamics using a single compact equation like Eq. (6.43). However, the bigger advantage of utilizing the Lie group methods in building the equation of motion is the fact that, unlike Eq. (6.36), the dynamics of a rigid body can always be expressed in the form of Eq. (6.43) regardless of the frame in which it is expressed.

Frame invariance

It was shown in Eq. (6.40) that the spatial form can also be used to write a compact expression for the kinetic energy. Since the kinetic energy of the rigid body must be independent of the frame of representation $\{a\}$ or $\{b\}$, one can write

$$\begin{aligned} K &= \frac{1}{2} \mathbf{V}_a^T \mathbf{M}_a \mathbf{V}_a = \frac{1}{2} \mathbf{V}_b^T \mathbf{M}_b \mathbf{V}_b \\ &= \frac{1}{2} \mathbf{V}_a^T \underbrace{\left[\mathbf{Ad}_{T_a} \right]^T \mathbf{M}_b \left[\mathbf{Ad}_{T_a} \right]}_{\mathbf{M}_a} \mathbf{V}_a, \end{aligned} \quad (6.44)$$

where the spatial mass inertia matrix in the frame $\{a\}$ is related to the spatial inertia matrix in frame $\{b\}$ by the relation:

$$\mathbf{M}_a = \left[\mathbf{Ad}_{T_a} \right]^{-T} \mathbf{M}_b \left[\mathbf{Ad}_{T_a} \right]^{-1}. \quad (6.45)$$

Spatial representation

Eq. (6.43) can also be written in the spatial representation, which in fact reveals an interesting property. Using Eq. (6.45), the spatial mass-inertia matrix in the spatial representation \mathbf{M} can be computed from spatial mass-inertia matrix in the body frame \mathbf{M}_b as:

$$\mathbf{M} = \left[\mathbf{Ad}_T \right]^{-T} \mathbf{M}_b \left[\mathbf{Ad}_T \right]^{-1}, \quad (6.46)$$

where \mathbf{T} denotes the homogeneous transformation of the body frame with respect to the base frame. The spatial momentum of the rigid body in the spatial representation is defined as

$$\mathbf{P} = \mathbf{M}\mathbf{V} = [\mathbf{Ad}_T]^{-T} \mathbf{M}_b [\mathbf{Ad}_T]^{-1} \mathbf{V}. \quad (6.47)$$

Using the fact that $\mathbf{V}_b = [\mathbf{Ad}_T] \mathbf{V}$ and Eq. (6.41), one could express the spatial momentum in spatial representation \mathbf{P} in terms of spatial momentum in body-fixed representation \mathbf{P}_b as:

$$\begin{aligned} \mathbf{P} &= [\mathbf{Ad}_T]^{-T} \mathbf{M}_b \mathbf{V}_b \\ &= [\mathbf{Ad}_T]^{-T} \mathbf{P}_b. \end{aligned} \quad (6.48)$$

Differentiating Eq. (6.47) with respect to time, one could arrive at the Newton–Euler equation of the single rigid body dynamics in spatial representation:

$$\mathbf{W} = \dot{\mathbf{P}} = \mathbf{M}\dot{\mathbf{V}} + \dot{\mathbf{M}}\mathbf{V}, \quad (6.49)$$

which is the simplest form possible (unlike Eq. (6.42)) and can be achieved by the spatial representation of the twist, wrench and momentum. The time derivative of spatial mass-inertia matrix $\dot{\mathbf{M}}$ can be computed using

$$\dot{\mathbf{M}} = -[\mathbf{ad}_V^T] \mathbf{M} - \mathbf{M}[\mathbf{ad}_V]. \quad (6.50)$$

Substituting Eq. (6.50) in Eq. (6.49) and using the fact that $[\mathbf{ad}_V] \mathbf{V} = 0$, one could arrive at the NE equations of motion in spatial representation.

$$\begin{aligned} \mathbf{W} &= \mathbf{M}\dot{\mathbf{V}} - [\mathbf{ad}_V^T] \mathbf{M}\mathbf{V} - \mathbf{M}[\mathbf{ad}_V] \mathbf{V} \\ \mathbf{W} &= \mathbf{M}\dot{\mathbf{V}} - [\mathbf{ad}_V^T] \mathbf{M}\mathbf{V} \end{aligned} \quad (6.51)$$

This has exactly the same form as body frame version of EOM, as shown in Eq. (6.43).

6.4 Inverse dynamics

Inverse dynamics is the problem of finding the forces required to produce a given motion in a rigid-body system. For a serial or tree type robot, all joints are assumed to be actuated and hence these forces can be attributed to the motor torques and is useful for analyzing and controlling robots. In

the previous sections, it was shown how a kinematic chain can be topologically described using the concepts from graph theory, then recursive formulations for computing the position, velocity and acceleration kinematics were presented. Then, equations of motion for describing a single rigid body dynamics were presented. In this section, equations of motion for a serial or tree-type system are solved recursively exploiting the concepts presented in previous sections. The inverse dynamics of a general kinematic tree can be obtained in two main steps:

1. Calculate the position, velocity and acceleration state of each body in the kinematic tree. This is called the **forward recursion**.
2. Calculate the wrenches required to produce these accelerations and subsequently the wrenches transmitted across the joints from the wrenches acting on the bodies. This step is called **backward recursion**.

6.4.1 Initialization

Fig. 6.3 shows the free body diagram of a body i which is a part of the kinematic tree. Recall that in a kinematic tree, every body has a unique parent. Revisiting the notation introduced in Section 6.1, $\lambda(i)$ denotes the parent of the body i and its children are given by the array $\mu(i)$. Let us define a body frame $\{B_i\}$ at every body and let ${}^i\mathbf{S}_i$ denote the constant joint screw of joint i represented in the body frame $\{B_i\}$ given by:

$${}^i\mathbf{S}_i = \begin{bmatrix} {}^i\hat{\mathbf{s}}_i \\ {}^i\mathbf{s}_{oi} \times {}^i\hat{\mathbf{s}}_i + h^i\hat{\mathbf{s}}_i \end{bmatrix} \in \mathbb{R}^6, \quad (6.52)$$

where ${}^i\mathbf{s}_{oi}$ is the 3D position vector of a point on the joint axis and ${}^i\hat{\mathbf{s}}_i$ is the unit direction vector of the joint axis, both with respect to the body frame $\{B_i\}$. Further, we denote the reference configuration of the body i with respect to the parent body $\lambda(i)$ in the zero configuration with the homogeneous transformation matrix ${}^{\lambda(i)}\mathbf{B}_i \in SE(3)$. This constitutes all the geometric information that is needed to compute the kinematics of the tree and is readily available from a CAD model of the robot.

Let us define a center of mass (COM) frame of body i with $\{C_i\}$ and express the spatial mass-inertia matrix of the body with respect to this COM frame with \mathbf{M}_{ci} given by:

$$\mathbf{M}_{ci} = \begin{bmatrix} \mathbf{I}_{ci} & \mathbf{0} \\ \mathbf{0} & m_i\mathbf{I} \end{bmatrix}, \quad (6.53)$$

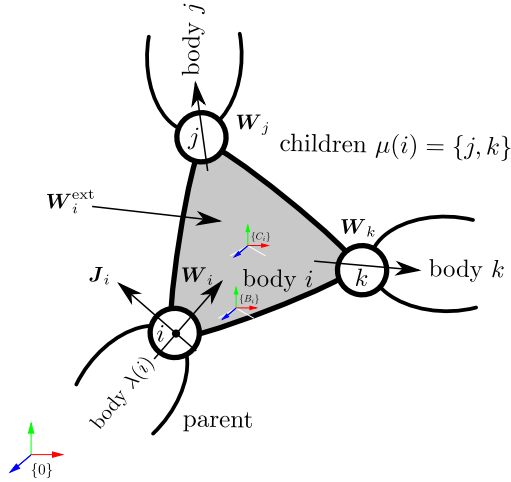


Figure 6.3 Forces acting on a body i [1].

where $m_i \in \mathbb{R}^+$ is the mass of the body i and $\mathbf{I}_{ci} \in \mathbb{R}^{3 \times 3}$ is its inertia matrix expressed in COM frame. Let ${}^{bi}\mathbf{T}_{ci} = ({}^{bi}\mathbf{R}_{ci}, {}^{bi}\mathbf{c}_{ci})$ denote the pose of the COM frame in the body frame for a body i . Using Eq. (6.44), the spatial mass-inertia matrix of the body in the body frame \mathbf{M}_{bi} can be computed as:

$$\begin{aligned} \mathbf{M}_{bi} &= [\mathbf{Ad}_{{}^{bi}\mathbf{T}_{ci}}]^{-T} \mathbf{M}_{ci} [\mathbf{Ad}_{{}^{bi}\mathbf{T}_{ci}}]^{-1} \\ &= \begin{bmatrix} \mathbf{I}_{bi} & m_i [{}^{bi}\mathbf{c}_{ci}] \\ -m_i [{}^{bi}\mathbf{c}_{ci}] & m_i \mathbf{I} \end{bmatrix}, \end{aligned} \quad (6.54)$$

where $\mathbf{I}_{bi} = {}^{bi}\mathbf{R}_{ci} \mathbf{I}_{ci} {}^{bi}\mathbf{R}_{ci}^T - m_i [{}^{bi}\mathbf{c}_{ci}]^2 \in \mathbb{R}^{3 \times 3}$ is the inertia matrix of the body in the body frame (a consequence of parallel axis theorem). This constitutes all the physical information that is needed to compute the dynamics of the kinematic chain and can be extracted from a CAD model of a robot.

6.4.2 Forward recursion

In the forward recursion, the spatial motion state of the bodies are calculated moving forward from the base link to the tip link ($i = 1, \dots, n$). Denote with $\mathbf{V}_i = (\boldsymbol{\omega}_i^T, \mathbf{v}_i^T)^T \in se(3)$ the twist vector of body i , and with \mathbf{J}_i the instantaneous screw coordinate vector of joint i , both in spatial representation.

The first step is to compute the pose of the body in the base frame. This can be done recursively thanks to the body-fixed version of the POE formula.

$$\mathbf{T}_i = \mathbf{T}_{\lambda(i)}^{\lambda(i)} \mathbf{B}_i \exp({}^i \mathbf{S}_i q_i) \quad (6.55)$$

Next, the instantaneous screw coordinate vector \mathbf{J}_i is computed in the base frame.

$$\mathbf{J}_i = \mathbf{Ad}_{\mathbf{T}_i}({}^i \mathbf{S}_i) \quad (6.56)$$

Then, the spatial velocity of body i is computed utilizing the spatial velocity of the previous body and velocity of the current joint.

$$\mathbf{V}_i = \mathbf{V}_{\lambda(i)} + \mathbf{J}_i \dot{q}_i \quad (6.57)$$

Finally, the spatial acceleration of the body is computed utilizing the spatial acceleration of the previous body and the acceleration across the joint, which makes use of instantaneous screw coordinate computed in Eq. (6.56) and the Lie brackets/adjoint mapping.

$$\dot{\mathbf{V}}_i = \dot{\mathbf{V}}_{\lambda(i)} + \mathbf{J}_i \ddot{q}_i + \mathbf{ad}_{\mathbf{V}_{\lambda(i)}} \mathbf{V}_i. \quad (6.58)$$

6.4.3 Backward recursion

Once we have the spatial velocities and accelerations of all the bodies computed moving forward from base to the tip, we can calculate the joint torques or forces by moving backwards from the tip to base link ($i = n, \dots, 1$). Denote with $\mathbf{W}_i = (\mathbf{m}_i^T, \mathbf{f}_i^T)^T \in se^*(3)$ the spatial wrench vector of body i .

The first step in the backward recursion is to compute the spatial mass-inertia matrix in the base frame \mathbf{M}_i from spatial mass-inertia matrix in the body frame \mathbf{M}_{bi} .

$$\mathbf{M}_i = [\mathbf{Ad}_{\mathbf{T}_i}]^{-T} \mathbf{M}_{bi} [\mathbf{Ad}_{\mathbf{T}_i}]^{-1}. \quad (6.59)$$

The wrench acting on any single rigid body i due to its motion ($\mathbf{V}_i, \dot{\mathbf{V}}_i$) is given by Eq. (6.43). However, the total wrench acting on the body i is the sum of the wrench transmitted through the joint, wrench applied to it through its children links and net external wrench acting on that body, resolved in the base frame.

$$\mathbf{W}_i = \sum_{j \in \mu(i)} \mathbf{W}_j + \mathbf{M}_i \dot{\mathbf{V}}_i - \mathbf{ad}_{\mathbf{V}_i}^T \mathbf{M}_i \mathbf{V}_i + \mathbf{W}_i^{\text{ext}} \quad (6.60)$$

The final step is to compute the force/torque needed by the actuator τ_i from the wrench acting on the body \mathbf{W}_i .

$$\tau_i = \mathbf{J}_i^T \mathbf{W}_i \quad (6.61)$$

6.4.4 Computational complexity

The approach presented in this section is also known as the spatial version of inverse dynamics algorithm. It has been argued in [3] that this version of the algorithm has the same $O(n)$ complexity as the inverse dynamics algorithm in body coordinates presented in [5]. However, this version of the algorithm is much more simpler and elegant when compared with the body coordinates version.

6.5 EOM in closed form

In this section, we show how the spatial representation of the recursive Newton–Euler algorithm for inverse dynamics presented in Section 6.4 can be organized into a set of dynamics equations in closed form in generalized coordinates. For body coordinates version of the closed-form inverse dynamics, refer to [4,11,12].

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_{\text{ext}} = \boldsymbol{\tau}. \quad (6.62)$$

Here \mathbf{q} , $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are $(n \times 1)$ vectors of joint position, velocity and acceleration variables of the system, $\mathbf{M}(\mathbf{q})$ is the $(n \times n)$ generalized mass-inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a $(n \times n)$ matrix for Coriolis-centrifugal forces, $\mathbf{g}(\mathbf{q})$ is the $(n \times 1)$ vector of gravity efforts, $\boldsymbol{\tau}_{\text{ext}}$ is the $(n \times 1)$ vector of external forces projected on the tree joints, and $\boldsymbol{\tau}$ is the $(n \times 1)$ vector of force/torque variables.

The first step in RNEA is the recursive computation of twist of each body using Eq. (6.57). Similar to Eq. (6.16), this could be written in closed form as:

$$\mathbf{V}^{\text{sys}} = \mathbf{J}^{\text{sys}} \dot{\mathbf{q}} + \mathbf{V}_{\text{base}}^{\text{sys}} = \mathbf{A}^{\text{sys}} \mathbf{S}^{\text{sys}} \dot{\mathbf{q}} + \mathbf{V}_{\text{base}}^{\text{sys}}, \quad (6.63)$$

where $\mathbf{V}_{\text{base}}^{\text{sys}} = [\mathbf{V}_{\text{base}}, \mathbf{0}, \dots, \mathbf{0}]^T \in \mathbb{R}^{6n}$ is the velocity of the system base.

The second step in RNEA is the recursive computation of spatial acceleration of every body using Eq. (6.58). Similar to Eq. (6.28), this equation

can be written in its closed form as:

$$\dot{\mathbf{V}}^{\text{sys}} = \dot{\mathbf{V}}_{\text{base}}^{\text{sys}} + \mathbf{J}^{\text{sys}} \ddot{\mathbf{q}} + \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} \mathbf{J}^{\text{sys}} \dot{\mathbf{q}} = \dot{\mathbf{V}}_{\text{base}}^{\text{sys}} + \mathbf{J}^{\text{sys}} \ddot{\mathbf{q}} + \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} \mathbf{V}^{\text{sys}}, \quad (6.64)$$

where $\dot{\mathbf{V}}_{\text{base}}^{\text{sys}} = [\dot{\mathbf{V}}_{\text{base}}, \mathbf{0}, \dots, \mathbf{0}]^T \in \mathbb{R}^{6n}$ is the acceleration of the system base.

The third step in RNEA is the recursive computation of wrench acting on each body using Eq. (6.60). This equation can be written in its closed form as:

$$\mathbf{W}^{\text{sys}} = \mathbf{M}^{\text{sys}} \dot{\mathbf{V}}^{\text{sys}} - [\mathbf{b}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \mathbf{V}^{\text{sys}} + \mathbf{W}_{\text{ext}}^{\text{sys}}, \quad (6.65)$$

where $\mathbf{W}_{\text{ext}}^{\text{sys}} = [\mathbf{W}_1^{\text{ext}}, \mathbf{W}_2^{\text{ext}}, \dots, \mathbf{W}_n^{\text{ext}}]^T \in \mathbb{R}^{6n}$ is the vector of external wrenches acting on the system and $\mathbf{M}^{\text{sys}} = \text{diag}(\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_n) \in \mathbb{R}^{6n \times 6n}$ is the spatial mass-inertia matrix of the system.

The final step in RNEA is to compute the actuation forces/torques using Eq. (6.61), which can be written in its closed form as:

$$\boldsymbol{\tau} = [\mathbf{J}^{\text{sys}}]^T \mathbf{W}^{\text{sys}}, \quad (6.66)$$

where $\boldsymbol{\tau}$ is the vector of actuator forces/torques.

After deriving the individual equations in RNEA in closed form, one could start building the closed form Lagrangian equation in generalized coordinates. Substituting Eq. (6.65) into Eq. (6.66), one gets:

$$\boldsymbol{\tau} = [\mathbf{J}^{\text{sys}}]^T \left(\mathbf{M}^{\text{sys}} \dot{\mathbf{V}}^{\text{sys}} - [\mathbf{b}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \mathbf{V}^{\text{sys}} + \mathbf{W}_{\text{ext}}^{\text{sys}} \right), \quad (6.67)$$

in which one can again substitute Eq. (6.64) to arrive at:

$$\begin{aligned} \boldsymbol{\tau} &= [\mathbf{J}^{\text{sys}}]^T \left(\mathbf{M}^{\text{sys}} \left[\dot{\mathbf{V}}_{\text{base}}^{\text{sys}} + \mathbf{J}^{\text{sys}} \ddot{\mathbf{q}} + \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} \mathbf{V}^{\text{sys}} \right] - [\mathbf{b}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \mathbf{V}^{\text{sys}} + \mathbf{W}_{\text{ext}}^{\text{sys}} \right) \\ \text{or } \boldsymbol{\tau} &= [\mathbf{J}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \left[\mathbf{J}^{\text{sys}} \ddot{\mathbf{q}} + \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} \mathbf{V}^{\text{sys}} + \dot{\mathbf{V}}_{\text{base}}^{\text{sys}} \right] - [\mathbf{J}^{\text{sys}}]^T [\mathbf{b}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \mathbf{V}^{\text{sys}} \\ &\quad + [\mathbf{J}^{\text{sys}}]^T \mathbf{W}_{\text{ext}}^{\text{sys}}. \end{aligned}$$

Substituting Eq. (6.63) into the above equation while assuming a system with fixed base, i.e., $\mathbf{V}_{\text{base}}^{\text{sys}} = \mathbf{0}$ and base acceleration as the gravity vector, i.e., $\dot{\mathbf{V}}_{\text{base}}^{\text{sys}} = \dot{\mathbf{V}}_g^{\text{sys}} = [\dot{\mathbf{V}}_g, \mathbf{0}, \dots, \mathbf{0}]^T$, one gets:

$$\boldsymbol{\tau} = \underbrace{[\mathbf{J}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \mathbf{J}^{\text{sys}}}_{\mathbf{M}(q)} \ddot{\mathbf{q}} + \underbrace{[\mathbf{J}^{\text{sys}}]^T \left(\mathbf{M}^{\text{sys}} \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} - [\mathbf{b}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \right) \mathbf{J}^{\text{sys}}}_{\mathbf{C}(q, \dot{q})} \dot{\mathbf{q}} + \underbrace{[\mathbf{J}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \dot{\mathbf{V}}_g^{\text{sys}}}_{\mathbf{g}(q)} + \underbrace{[\mathbf{J}^{\text{sys}}]^T \mathbf{W}_{\text{ext}}^{\text{sys}}}_{\boldsymbol{\tau}_{\text{ext}}}. \quad (6.68)$$

Hence, the expressions for individual terms in Eq. (6.62) are given by:

$$\mathbf{M}(\mathbf{q}) = [\mathbf{J}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \mathbf{J}^{\text{sys}} \quad (6.69)$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = [\mathbf{J}^{\text{sys}}]^T (\mathbf{M}^{\text{sys}} \mathbf{L}^{\text{sys}} \mathbf{b}^{\text{sys}} - [\mathbf{b}^{\text{sys}}]^T \mathbf{M}^{\text{sys}}) \mathbf{J}^{\text{sys}} \quad (6.70)$$

$$\mathbf{g}(\mathbf{q}) = [\mathbf{J}^{\text{sys}}]^T \mathbf{M}^{\text{sys}} \dot{\mathbf{v}}_{\mathbf{g}}^{\text{sys}} \quad (6.71)$$

$$\boldsymbol{\tau}_{\text{ext}} = [\mathbf{J}^{\text{sys}}]^T \mathbf{W}_{\text{ext}}^{\text{sys}}, \quad (6.72)$$

where the system Jacobian can be further factorized and written as $\mathbf{J}^{\text{sys}} = \mathbf{A}^{\text{sys}} \mathbf{S}^{\text{sys}}$, as shown in Eq. (6.17).

6.6 Conclusion

This chapter presents the screw theory and Lie group theory based approaches for the kinematic and dynamic modeling of serial or tree type robotic systems. Starting with a topological description of a kinematic chain using graph theoretic concepts, the recursive relations for computing kinematics are presented. Then, equations of motion for a single rigid body dynamics are presented from both classical and modern viewpoints in order to highlight the advantage of the modern geometric setting. Finally, a fully recursive $O(n)$ algorithm for computing the inverse dynamics of a kinematic tree is presented and further utilized to develop EOM in closed form. This chapter hence provides the theoretical background needed to understand the kinematics and dynamics of more complex series-parallel hybrid robotic systems, which will be discussed in the next chapter.

References

- [1] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
- [2] A. Müller, Screw and Lie group theory in multibody kinematics, *Multibody System Dynamics* 43 (2018) 37–70, <https://doi.org/10.1007/s11044-017-9582-7>.
- [3] A. Müller, Screw and Lie group theory in multibody dynamics, *Multibody System Dynamics* 42 (2) (2018) 219–248, <https://doi.org/10.1007/s11044-017-9583-6>.
- [4] K.M. Lynch, F.C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st edition, Cambridge University Press, New York, NY, USA, 2017.
- [5] R. Featherstone, *Rigid Body Dynamics Algorithm*, 1st edition, Springer, New York, NY, USA, 2008, <https://doi.org/10.1007/978-1-4899-7560-7>.
- [6] R.M. Murray, S.S. Sastry, L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*, 1st edition, CRC Press, Inc., Boca Raton, FL, USA, 1994.
- [7] A. Jain, Graph theoretic foundations of multibody dynamics, *Multibody System Dynamics* 26 (3) (2011) 307–333, <https://doi.org/10.1007/s11044-011-9266-7>.
- [8] A. Mueller, Kinematic topology and constraints of multi-loop linkages, *Robotica* 36 (11) (2018) 1641–1663, <https://doi.org/10.1017/S0263574718000619>.

- [9] R.W. Brockett, Robotic manipulators and the product of exponentials formula, in: *Mathematical Theory of Networks and Systems: Proceedings of the MTNS-83 International Symposium Beer Sheva, Israel, June 20–24, 1983*, Springer, 2005, pp. 120–129.
- [10] J. Denavit, R.S. Hartenberg, A kinematic notation for lower-pair mechanisms based on matrices, *Journal of Applied Mechanics* 22 (2) (2021) 215–221, <https://doi.org/10.1115/1.4011045>, https://asmedigitalcollection.asme.org/appliedmechanics/article-pdf/22/2/215/6748803/215_1.pdf.
- [11] A. Mueller, S. Kumar, Closed-form time derivatives of the equations of motion of rigid body systems, *Multibody System Dynamics* 53 (3) (2021) 257–273.
- [12] F.C. Park, B. Kim, C. Jang, J. Hong, Geometric algorithms for robot dynamics: a tutorial review, *Applied Mechanics Reviews* 70 (1) (2018) 010803, <https://doi.org/10.1115/1.4039078>, https://asmedigitalcollection.asme.org/appliedmechanicsreviews/article-pdf/70/1/010803/6780035/amr_070_01_010802.pdf.

CHAPTER 7

Modular algorithms for kinematics and dynamics of series-parallel hybrid robots

Shivesh Kumar^a, Rohit Kumar^a, Andreas Müller^b, and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

^cWorking Group Robotics, University of Bremen, Bremen, Germany

This chapter presents a modular framework to compute the kinematics and dynamics of arbitrary hybrid robots that are a serial composition of serial or parallel submechanism modules. The approach allows modular composition of the explicit loop closure constraints (resolved either analytically or numerically) and its associated derivatives and transfers them into computation of the analytical forward and inverse dynamics algorithms. The benefit of this approach is that efficient and recursive $O(n)$ dynamics algorithms for tree type systems can be directly used. The results are free of numerical errors resulting from loop closure and free of singularities arising from redundant constraints. This approach forms the basis of the Hybrid Robot Dynamics (HyRoDyn) software framework. The explanation in this chapter is aided with an example of a series-parallel hybrid humanoid robot developed at DFKI-RIC which employs several different closed-loop and parallel mechanism based modules. The content presented here is based on [1–5] and was first presented in the form of a poster [6].

This chapter is organized as follows: Section 7.1 provides theoretical preliminaries for modeling robots with closed loops along with an introduction to the concept of loop closure functions. Section 7.2 introduces the notion of modularity and guidelines for selecting submechanism modules in series-parallel hybrid robotic systems. Section 7.3 presents the modular graph-based topological modeling of series-parallel hybrid robots using the previously identified submechanism modules. Section 7.4 elaborates the analytical and modular approach for kinematic and dynamics modeling of

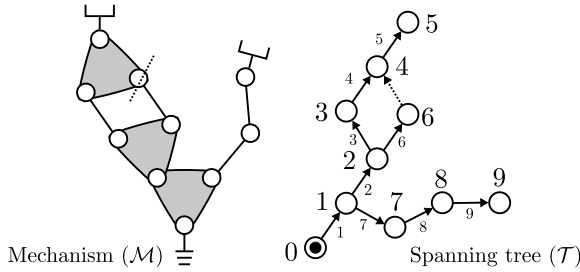


Figure 7.1 A closed-loop robot and its associated spanning tree numbered using regular numbering scheme. Here, links and joints are denoted as vertices and edges respectively (Used with permission of American Society of Mechanical Engineers ASME, from [1,2]; permission conveyed through Copyright Clearance Center, Inc.).

such series-parallel hybrid systems. It further comments on their computational complexity. Section 7.5 draws the conclusions from this chapter.

7.1 Modeling rigid-body systems with closed loops

This section briefly introduces the theory of multi-body dynamics subjected to holonomic and scleronomic constraints. It mostly adopts the notation and terminology introduced by Featherstone in [7]. Consider a rigid body system with N_B bodies, N_J joints, and $N_L = N_J - N_B$ kinematic loops. Assume that a spanning tree is defined and that the joints are enumerated using regular numbering scheme (see Fig. 7.1 for an example). Let n denote the degree of freedom of the selected spanning tree, computed as $n = \sum_{i=1}^{N_B} n_i$, and let n_c denote the number of loop-closure constraints, computed as $n_c = \sum_{k=N_B+1}^{N_J} n_{ck}$ where n_{ck} denotes the number of loop constraints imposed by k th cut joint. Further, let \mathbf{q} indicate the vector of all joints of the spanning tree (of size n) and let $\boldsymbol{\gamma}$ indicate the vector of all independent variables (of size $n - n_c$).

Table 7.1 Loop constraints [7].

Type	position	velocity	acceleration
implicit:	$\phi(\mathbf{q}) = \mathbf{0}$	$\mathbf{K}\dot{\mathbf{q}} = \mathbf{0}$	$\mathbf{K}\ddot{\mathbf{q}} = \mathbf{k}$
explicit:	$\mathbf{q} = \boldsymbol{\gamma}(\boldsymbol{\gamma})$	$\dot{\mathbf{q}} = \mathbf{G}\dot{\boldsymbol{\gamma}}$	$\ddot{\mathbf{q}} = \mathbf{G}\ddot{\boldsymbol{\gamma}} + \mathbf{g}$

7.1.1 Loop constraints

Loop constraints are non-linear constraints on the motion variables of a multi-body system. Loop constraints can be expressed in an implicit and in an explicit way, they are summarized in Table 7.1 at position, velocity, and acceleration levels. Here let $\mathbf{K} = \frac{\partial \phi}{\partial \mathbf{q}}$, $\mathbf{k} = -\dot{\mathbf{K}}\dot{\mathbf{q}}$, $\mathbf{G} = \frac{\partial \gamma}{\partial \dot{\mathbf{y}}}$, and $\mathbf{g} = \dot{\mathbf{G}}\dot{\mathbf{y}}$. If both functions ϕ and γ describe the same constraint,

$$\phi \circ \gamma = \mathbf{0},$$

$$\mathbf{K}\mathbf{G} = \mathbf{0},$$

$$\mathbf{K}\mathbf{g} = \mathbf{k},$$

can be deduced. Algorithms to compute variables in Table 7.1 from the spanning tree are provided in [7] and skipped here for brevity.

Example 7.1. Consider a planar lambda mechanism, a variant of slider crank mechanism.

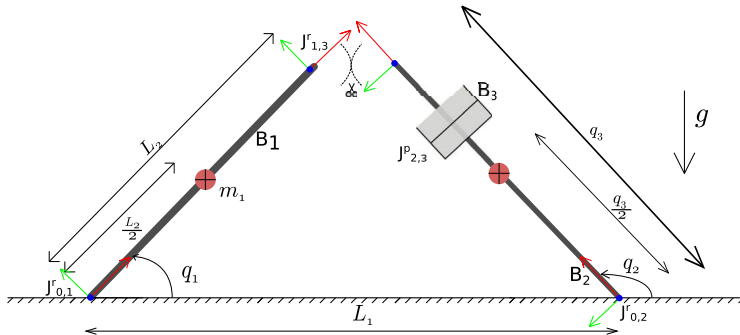


Figure 7.2 Lambda or RRPR mechanism [4].

Referring to Fig. 7.2, the lambda mechanism consist of base of length L_0 , link B_2 as the stator part of the prismatic actuator, and links B_1 and B_3 of lengths L_2 and q_3 respectively. The joints are described as $J_{0,1}^r = 1$ (revolute joint between base link and body B_1), $J_{0,2}^r = 2$ (revolute joint between base link and body B_2), $J_{1,3}^r = 4$ revolute joint between body B_1 and body B_3 , and $J_{2,3}^p = 3$ (revolute joint between body B_1 and prismatic link B_3).

The introduction of cut joint at $J_{1,3}^r = 4$ defines the spanning tree as $\mathcal{T} = \{0, 1, 2, 3\}$.

The topological graph can be depicted in Fig. 7.3, where dashed line represents the cut joint in the graph. For the topological graph in Fig. 7.3, the arrays can be defined as

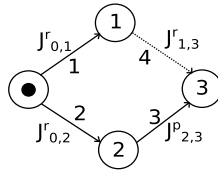


Figure 7.3 Topological graph for lambda mechanism [4].

- Predecessor array $p = \{0, 0, 2\}$
- Successor array $s = \{1, 2, 3\}$
- Parent array $\lambda = \{0, 0, 2\}$
- Child array $\mu = \{\{1, 2\}, \{\}, \{3\}, \{\}\}$
- Support array $\kappa = \{\{1\}, \{2\}, \{2, 3\}\}$

Considering the RRRP mechanism, we have $N_B = 3$, and $N_J = 4$. The number of kinematic loops is $N_J - N_B = 1$. The cut joint as defined in Fig. 7.3 is $k = 4 = J_{1,3}^r$. The screw coordinates of the above model can be written as

Table 7.2 Screw coordinates for different joints [4].

Joint	Pitch h	\mathbf{x}	$\hat{\mathbf{e}}$	$\mathbf{S} = \begin{bmatrix} \hat{\mathbf{e}} \\ \mathbf{x} \times \hat{\mathbf{e}} + h\hat{\mathbf{e}} \end{bmatrix}^T$
Joint 1	0	$\begin{bmatrix} 0, 0, 0 \end{bmatrix}^T$	$\begin{bmatrix} 0, 0, 1 \end{bmatrix}^T$	$\begin{bmatrix} 0, 0, 1, 0, 0, 0 \end{bmatrix}^T$
Joint 2	0	$\begin{bmatrix} L_1, 0, 0 \end{bmatrix}^T$	$\begin{bmatrix} 0, 0, 1 \end{bmatrix}^T$	$\begin{bmatrix} 0, 0, 1, 0, -L_1, 0 \end{bmatrix}^T$
Joint 3	∞	$\begin{bmatrix} 0, 0, 0 \end{bmatrix}^T$	$\begin{bmatrix} 1, 0, 0 \end{bmatrix}^T$	$\begin{bmatrix} 0, 0, 0, 1, 0, 0 \end{bmatrix}^T$

In Table 7.2, \mathbf{x} is the vector from the inertial frame to the body-fixed frame in zero configuration. $\hat{\mathbf{e}}$ is the unit vector along the direction of joint axis. From Fig. 7.2, the revolute joint axis for joints 1 and 2 is in the positive z-direction and the prismatic joint axis for joint 3 is along the positive x-axis.

Position level implicit constraint:

Using screw theory, the position level loop constraint can be defined for the two serial link chains. For the predecessor body of the joint 4, $p(k) = 1$,

$${}^0\mathbf{T}_{p(k)}(\mathbf{q}) = \exp([\mathbf{S}_1]q_1) {}^0\mathbf{T}_{p(k)}(\mathbf{0}), \tag{7.1}$$

where ${}^0\mathbf{T}_{p(k)}(\mathbf{0}) = \mathbf{I}_{4 \times 4}$ is zero pose configuration of the predecessor body.

$${}^0\mathbf{T}_{p(k)}(\mathbf{q}) = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.2)$$

For the successor body of the joint 4, $s(k) = 3$,

$${}^0\mathbf{T}_{s(k)}(\mathbf{q}) = \exp([\mathbf{S}_2]q_2) \exp([\mathbf{S}_3]q_3) {}^0\mathbf{T}_{s(k)}(\mathbf{0}), \quad (7.3)$$

where ${}^0\mathbf{T}_{s(k)}(\mathbf{0})$ is zero pose configuration of the successor body given by

$${}^0\mathbf{T}_{s(k)}(\mathbf{0}) = \begin{bmatrix} 1 & 0 & 0 & L_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (7.4)$$

$${}^0\mathbf{T}_{s(k)}(\mathbf{q}) = \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & L_1 + q_3 \cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & q_3 \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.5)$$

The relative motion of the between the predecessor and successor body is given by $\mathbf{T}_{p(k)}^{-1} \mathbf{T}_{s(k)}$. The loop closure condition can be written as

$$\exp([\mathbf{S}_k]q_k) = ({}^{p(k)}\mathbf{T}_k^{-1}) ({}^0\mathbf{T}_{p(k)}^{-1}) ({}^0\mathbf{T}_{s(k)}) ({}^{s(k)}\mathbf{T}_k) \quad (7.6)$$

$$\exp([\mathbf{S}_k]q_k) = \begin{bmatrix} \Delta \mathbf{R}_k & \Delta \mathbf{p}_k \\ \mathbf{0} & 1 \end{bmatrix} \quad (7.7)$$

$$\exp([\mathbf{S}_k]q_k) = \begin{bmatrix} 1 & 0 & 0 & L_1 - L_2 \cos(q_1) + q_3 \cos(q_2) \\ 0 & 1 & 0 & q_3 \sin(q_2) - L_2 \sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.8)$$

As the cut joint is a revolute joint, the position part of the matrix should be zero.

$$\Delta \mathbf{p}_k = \mathbf{0}. \quad (7.9)$$

This will define our implicit constraints of the model.

$$\boldsymbol{\phi}(\mathbf{q}) = \begin{pmatrix} L_1 - L_2 \cos(q_1) + q_3 \cos(q_2) \\ q_3 \sin(q_2) - L_2 \sin(q_1) \\ 0 \end{pmatrix} = \mathbf{0} \quad (7.10)$$

Velocity level implicit constraint:

The velocity level loop constraints can be written as

$$\mathbf{Ad}_{T_{p(k),k}^{-1}} \mathbf{Ad}_{T_{0,p(k)}^{-1}} (\mathbf{V}_{s(k)}^s - \mathbf{V}_{p(k)}^s) = \mathbf{S}_k \dot{\mathbf{q}}_k \quad (7.11)$$

Multiplying by the orthogonal complement of \mathbf{S}_k , i.e., \mathbf{W}_k^T , as $\mathbf{W}_k^T \mathbf{S}_k = \mathbf{0}$,

$$\mathbf{W}_k^T \mathbf{Ad}_{T_{p(k),k}^{-1}} \mathbf{Ad}_{T_{0,p(k)}^{-1}} (\mathbf{V}_{s(k)}^s - \mathbf{V}_{p(k)}^s) = \mathbf{0} \quad (7.12)$$

Rewriting the above equation,

$$\mathbf{W}_k^T \mathbf{Ad}_{T_{p(k),k}^{-1}} \mathbf{Ad}_{T_{0,p(k)}^{-1}} (\mathbf{J}_{s(k)}^s - \mathbf{J}_{p(k)}^s) \dot{\mathbf{q}} = \mathbf{0}, \quad (7.13)$$

where \mathbf{J}_i^s is the spatial Jacobian matrix of body i .

Therefore, constraint Jacobian matrix can be written as

$$\mathbf{K} = \mathbf{W}_k^T \mathbf{Ad}_{T_{p(k),k}^{-1}} \mathbf{Ad}_{T_{0,p(k)}^{-1}} (\mathbf{J}_{s(k)}^s - \mathbf{J}_{p(k)}^s). \quad (7.14)$$

Computing the above expression,

$$\mathbf{K} = \begin{bmatrix} -L_2 \sin(q_1) & q_3 \sin(q_2) & -\cos(q_2) \\ L_2 \cos(q_1) & -q_3 \cos(q_2) & -\sin(q_2) \end{bmatrix}, \quad (7.15)$$

which is the constraint Jacobian matrix.

The acceleration level implicit constraints can be obtained by differentiating the velocity level constraint equation.

Example 7.2. This example demonstrates the derivation of explicit loop closure constraints for the lambda mechanism considered in the previous example. In explicit form, a subset of position vector need to be chosen. The independent coordinate selected in this case is q_1 ,

$$\boldsymbol{\gamma} = q_1. \quad (7.16)$$

Position level explicit constraints:

For explicit constraints, a function $\boldsymbol{\gamma}$ is defined to satisfy $\boldsymbol{q} = \boldsymbol{\gamma}(\boldsymbol{\gamma})$. From solving Eq. (7.10), q_2 and q_3 are written as a function of q_1 .

$$q_2 = \tan^{-1} \frac{L_2 \sin q_1}{L_2 \cos q_1 - L_1}, \quad (7.17)$$

$$q_3 = \sqrt{L_1^2 + L_2^2 - 2L_1 L_2 \cos q_1}. \quad (7.18)$$

Therefore,

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ \tan^{-1} \frac{L_2 \sin q_1}{L_2 \cos q_1 - L_1} \\ \sqrt{L_1^2 + L_2^2 - 2L_1 L_2 \cos q_1} \end{bmatrix}. \quad (7.19)$$

Velocity level explicit constraint:

Explicit loop closure Jacobian matrix \mathbf{G} is defined as $\mathbf{G} = \frac{\partial \boldsymbol{\gamma}}{\partial \boldsymbol{\gamma}}$. Differentiating the above equations w.r.t. $\boldsymbol{\gamma}$ or q_1 ,

$$\mathbf{G} = \begin{bmatrix} 1 \\ \frac{L_2^2 - L_1 L_2 \cos q_1}{q_3^2} \\ \frac{L_1 L_2 \sin q_1}{q_3} \end{bmatrix}. \quad (7.20)$$

Acceleration level explicit constraint:

Differentiating $\dot{\boldsymbol{q}} = \mathbf{G}\dot{\boldsymbol{\gamma}}$ again,

$$\ddot{\boldsymbol{q}} = \mathbf{G}\ddot{\boldsymbol{\gamma}} + \boldsymbol{g}, \quad (7.21)$$

where

$$\boldsymbol{g} = \dot{\mathbf{G}}\dot{\boldsymbol{\gamma}} = \begin{bmatrix} 0 \\ \frac{L_2 \dot{q}_1 (2L_1 \dot{q}_3 \cos(q_1) - 2L_2 \dot{q}_3 + L_1 \dot{q}_1 q_3 \sin(q_1))}{q_3^3} \\ -\frac{L_1 L_2 \dot{q}_1 (\dot{q}_3 \sin(q_1) - \dot{q}_1 q_3 \cos(q_1))}{q_3^2} \end{bmatrix} \quad (7.22)$$

Example 7.3. This example demonstrates the equivalence of implicit and explicit loop closure constraints for the lambda mechanism. To verify this at velocity level, $\mathbf{K}\mathbf{G} = \mathbf{0}$ must hold.

$$\mathbf{K}\mathbf{G} = \begin{bmatrix} -L_2 \sin q_1 & q_3 \sin q_2 & -\cos q_2 \\ L_2 \cos q_1 & -q_3 \cos q_2 & -\sin q_2 \end{bmatrix} \begin{bmatrix} 1 \\ \frac{L_2^2 - L_1 L_2 \cos q_1}{q_3^2} \\ \frac{L_1 L_2 \sin q_1}{q_3} \end{bmatrix}$$

The first row of the resulting product \mathbf{KG} ,

$$-L_2 \sin q_1 + \frac{q_3 \sin q_2 L_2^2 - L_1 L_2 q_3 \cos q_1 \sin q_2}{q_3^2} - \cos q_2 \frac{L_1 L_2 \sin q_1}{q_3} \quad (7.23)$$

can be simplified as

$$\begin{aligned} &= -L_2 \sin q_1 + \frac{L_2 \sin q_1 L_2^2 - L_1 L_2 q_3 \cos q_1 \sin q_2}{q_3^2} - \frac{(L_2 \cos q_1 - L_1) L_2^2 \sin q_1}{q_3} \\ &= -L_2 \sin q_1 + \frac{L_2 \sin q_1 L_2^2 - 2L_1 L_2^2 \cos q_1 \sin q_1 + L_1^2 L_2 \sin q_1}{q_3^2} \\ &= -L_2 \sin q_1 + L_2 \sin q_1 \frac{L_2^2 - 2L_1 L_2 \cos q_1 + L_1^2}{L_1^2 + L_2^2 - 2L_1 L_2 \cos q_1} = 0 \end{aligned} \quad (7.24)$$

Similarly, for second row of the product \mathbf{KG} can be simplified as

$$\begin{aligned} &L_2 \cos q_1 - q_3 \cos q_2 \frac{L_2^2 - L_1 L_2 \cos q_1}{q_3^2} - \sin q_2 \frac{L_1 L_2 \sin q_1}{q_3} \\ &= L_2 \cos q_1 - (L_2 \cos q_1 - L_1) \frac{L_2^2 - L_1 L_2 \cos q_1}{q_3^2} - \frac{L_2 \sin q_1}{q_3} \frac{L_1 L_2 \sin q_1}{q_3} \\ &= L_2 \cos q_1 - \frac{(L_2^3 \cos q_1 - L_1 L_2^2 \cos^2 q_1 - L_1 L_2^2 + L_1^2 L_2 \cos q_1)}{q_3^2} \\ &\quad - \frac{L_1 L_2^2 - L_1 L_2^2 \cos^2 q_1}{q_3^2} \\ &= L_2 \cos q_1 - L_2 \cos q_1 \frac{(L_2^2 - L_1 L_2 \cos q_1 + L_1^2 - L_1 L_2 \cos q_1)}{q_3^2} = 0 \end{aligned} \quad (7.25)$$

Hence, as $\mathbf{KG} = \mathbf{0}$, the implicit and explicit constraints derived in the above two examples are equivalent.

7.1.2 Equations of motion (EOM)

The equations of motion for a tree topology multi-body system can be written as

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}), \quad (7.26)$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ are $(n \times 1)$ vectors of joint position, velocity, and acceleration variables of the spanning tree, $\mathbf{M}(\mathbf{q})$ is the $(n \times n)$ generalized mass-inertia

matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a $(n \times 1)$ vector for Coriolis–centrifugal and gravity efforts, and $\boldsymbol{\tau}$ is the $(n \times 1)$ vector of force/torque variables. In case of robots with closed loops, the equivalent spanning tree of the robot system is subjected to loop constraint forces

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} + \boldsymbol{\tau}_a + \boldsymbol{\tau}_c, \quad (7.27)$$

where $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}_a$ are the constraint and active forces, respectively produced by the cut joints. If the selected cut joint is passive, $\boldsymbol{\tau}_a = 0$ can be substituted in Eq. (7.27). The constraint force $\boldsymbol{\tau}_c$ is usually unknown, but its value can either be calculated or eliminated from the equation following the Jourdain's principle [8] of virtual power, i.e., $\boldsymbol{\tau}_c \dot{\mathbf{q}} = 0$. Based on the (implicit or explicit) nature of the loop constraints, the equations of motion are developed for the entire system.

7.1.2.1 EOM with implicit loop constraints

The cut joints impose a set of kinematic constraints on the spanning tree, which are briefly introduced in Table 7.1. Assuming that the implicit position level constraints have been successfully differentiated twice, the acceleration level loop constraints can be collected in a single matrix equation of the form

$$\mathbf{K}\ddot{\mathbf{q}} = \mathbf{k}, \quad (7.28)$$

where \mathbf{K} is a $(n_c \times n)$ matrix. If the system is subjected to an implicit loop constraint, then it can be shown that $\boldsymbol{\tau}_c$ takes the form

$$\boldsymbol{\tau}_c = \mathbf{K}^T \boldsymbol{\lambda}, \quad (7.29)$$

where $\boldsymbol{\lambda}$ is the vector of Lagrangian multipliers. Combining Eqs. (7.27), (7.28), and (7.29), the equation of motion of the overall system taking into account implicit loop constraints can be written as

$$\begin{bmatrix} \mathbf{M} & \mathbf{K}^T \\ \mathbf{K} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \mathbf{C} + \boldsymbol{\tau}_a \\ \mathbf{k} \end{bmatrix}. \quad (7.30)$$

This is a system of $(n + n_c)$ equations in $(n + n_c)$ unknowns. The coefficient matrix of dimension $(n + n_c) \times (n + n_c)$ in Eq. (7.30) is symmetric, but not positive definite. If the rank of matrix $r = \text{rank}(\mathbf{K})$ is less than n_c , then the coefficient matrix becomes singular and the system is said

to be over-constrained. Over-constrained systems are actually very common. For example, planar kinematic loops impose redundant constraints on the system – that either need to be removed manually [7] or demand numerical decomposition techniques which deteriorate the computational performance and numerical accuracy of the solution [9].

7.1.2.2 EOM with explicit loop constraints

Using explicit velocity level loop constraint (refer to Table 7.1) and Jourdain's principle of virtual power, one can establish that $\boldsymbol{\tau}_c$ has the following property

$$\mathbf{G}^T \boldsymbol{\tau}_c = 0. \quad (7.31)$$

Similarly, one can write the explicit motion constraints at an acceleration level

$$\ddot{\mathbf{q}} = \mathbf{G}\ddot{\mathbf{y}} + \mathbf{g}. \quad (7.32)$$

Combining Eqs. (7.27), (7.31), and (7.32), the equation of motion taking into account explicit loop constraints can be developed.

$$\begin{bmatrix} \mathbf{M} & -\mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{G} \\ \mathbf{0} & \mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\tau}_c \\ \ddot{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \mathbf{C} + \boldsymbol{\tau}_d \\ -\mathbf{g} \\ \mathbf{0} \end{bmatrix} \quad (7.33)$$

7.1.3 Loop closure functions

It is usually more complex to deal with robots involving closed loops. In contrast to a tree type robot, the mobility of closed-loop system is dependent on r , which can vary with the configuration. Also, the different assembly modes can lead to configuration ambiguities. Thus, it is useful to derive explicit functions for modeling closed-loop systems whenever and wherever possible.

Let us define loop closure functions such that they provide a unique mapping between the independent position variables \mathbf{y} and position variables in the spanning tree \mathbf{q} . The set of independent joint variables \mathbf{y} may or may not be a subset of the position variables of the spanning tree \mathbf{q} . For this assumption to hold true, let us define a set $C \subseteq \mathbb{R}^{n-r}$ of acceptable values of

\mathbf{y} , and assume $\mathbf{y} \in C$. For all $\mathbf{y} \in C$, there exists a function, γ such that

$$\begin{aligned} \mathbf{q} &= \gamma(\mathbf{y}), \\ \dot{\mathbf{q}} &= \mathbf{G}\dot{\mathbf{y}}, \\ \ddot{\mathbf{q}} &= \mathbf{G}\ddot{\mathbf{y}} + \dot{\mathbf{G}}\dot{\mathbf{y}} = \mathbf{G}\ddot{\mathbf{y}} + \mathbf{g}. \end{aligned} \quad (7.34)$$

It must be noted that the above relations are identical to the explicit loop constraint equations noted in Table 7.1 when the loop closure errors are zero. This method is less generic in nature than the ones described before because it is not always possible to find such a mapping analytically. However, the advantages of an analytical solution to the loop closure constraints outweighs the manual effort needed because numerical loop closure errors can not occur. Also, there is no need to introduce a constraint stabilization term unlike when practically dealing with Eq. (7.28). Further, the spanning tree can be selected such that there are no active forces on the cut joints, i.e., $\boldsymbol{\tau}_a = 0$.

7.1.4 Forward and inverse dynamics

The equations of motion presented above could be either solved for independent joint accelerations $\ddot{\mathbf{y}}$ under given actuator force conditions or for the actuator forces \mathbf{u} required to generate given acceleration. The former is called the forward dynamics problem and the latter is called the inverse dynamics problem [7,10]. In the following, the forward and inverse dynamics formula are derived such that a link between the loop closure functions and spanning-tree dynamics is established.

7.1.4.1 Forward dynamics solution

By using Eq. (7.31) and multiplying by \mathbf{G}^T on both sides of Eq. (7.27), the loop constraint forces $\boldsymbol{\tau}_c$ can be eliminated.

$$\mathbf{G}^T \mathbf{M}\ddot{\mathbf{q}} = \mathbf{G}^T (\boldsymbol{\tau} - \mathbf{C}) \quad (7.35)$$

Substituting Eq. (7.32) in Eq. (7.35) and simplifying one can arrive at the solution to the forward dynamics problem:

$$\ddot{\mathbf{y}} = (\mathbf{G}^T \mathbf{M} \mathbf{G})^{-1} (\mathbf{G}^T \boldsymbol{\tau} - \mathbf{G}^T (\mathbf{C} + \mathbf{M} \mathbf{g})). \quad (7.36)$$

7.1.4.2 Inverse dynamics solution

Eq. (7.35) could be rewritten as:

$$\mathbf{G}^T \boldsymbol{\tau} = \mathbf{G}^T (\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}) = \mathbf{G}^T \boldsymbol{\tau}_{ID}, \quad (7.37)$$

where $\boldsymbol{\tau}_{ID}$ is the inverse dynamics output of a spanning tree given by

$$\boldsymbol{\tau}_{ID} = \mathbf{M}(\boldsymbol{\gamma}(\boldsymbol{y}))(\mathbf{G}\ddot{\boldsymbol{y}} + \mathbf{g}) + \mathbf{C}(\boldsymbol{\gamma}(\boldsymbol{y}), \mathbf{G}\dot{\boldsymbol{y}}). \quad (7.38)$$

The solution to Eq. (7.37) is not unique, because \mathbf{G}^T is an $(n-r) \times n$ matrix, which imposes $(n-r)$ constraints on an n dimensional vector of unknowns, leaving r freedoms of choice. In other words, there are ∞^r different values of $\boldsymbol{\tau}$ which will produce the same acceleration. To arrive at a unique solution, the actuated degrees of freedom must be separated from the passive degrees of freedom. This can be done with the help of a matrix \mathbf{G}_u which basically contain the rows of \mathbf{G} corresponding to the actuated degrees of freedom. If the rank of matrix \mathbf{G}_u is equal to $(n-r)$, then the system is properly actuated¹ and a unique solution to the inverse dynamics problem can be found, which is given by:

$$\boldsymbol{\tau}_u = \mathbf{G}_u^{-T} \mathbf{G}^T \boldsymbol{\tau}_{ID}, \quad (7.39)$$

where $\boldsymbol{\tau}_u$ is a vector of actuator forces required to produce the given acceleration $\ddot{\boldsymbol{y}}$.

7.1.5 Comparison of numerical and analytical resolution of loop constraints: case study of a 3D slider crank mechanism

To make a comparison between numerical and analytical resolution of loop closure constraints, we present the case study of a 3D slider-crank mechanism. This mechanism consists of a crank, a connecting rod and a slider. Fig. 7.4a shows the schematic of the 3D slider crank linkage. The crank is driven by a motor with its rotational axis parallel to global X-axis. The slider makes a sliding movement on the ground with its translational axis coinciding with the global X-axis. Those two parts are connected via the connecting rod that itself is connected to the two parts by use of two ZYX ball joints. Since, the system has two spherical joints (SS pair) on the connecting rod, the system has one redundant rotational DOF around the rod's

¹ If $p > \text{rank}(\mathbf{G}_u)$, the system is redundantly actuated and $\boldsymbol{\tau}_u = \mathbf{G}_u^{\dagger T} \mathbf{G}^T \boldsymbol{\tau}_{ID}$.

axis (see Fig. 7.4b). The input motor angle is denoted with φ and the output slider movement is denoted with d . An input movement trajectory, $\varphi = \pi t^2$, is provided to the mechanism for a simulation time of 1 second in 100 time steps. The numerical solution is obtained using the multi-body simulation tool called ADAMS and the analytical solution is implemented in the HyRoDyn software framework. The analytical equations for the loop closure function are skipped here for brevity.

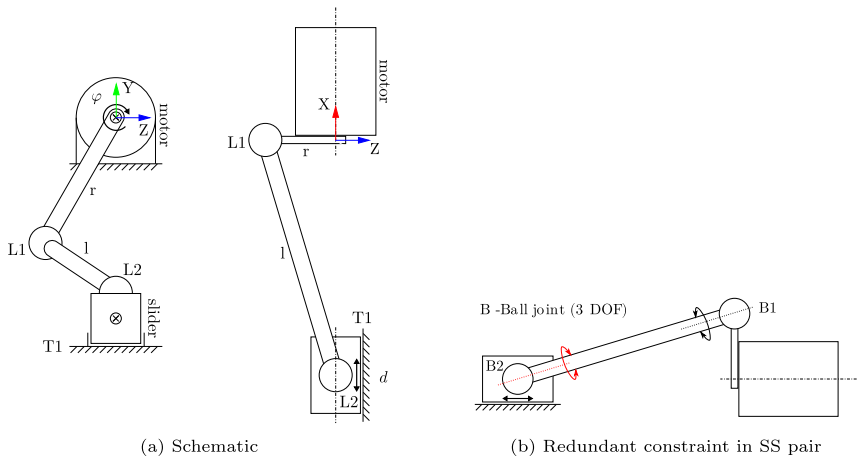


Figure 7.4 3D slider crank mechanism [5].

Fig. 7.5 shows the input and output motion trajectories plotted using HyRoDyn and ADAMS respectively. It can be noticed that the plots produced by ADAMS are subjected to some numerical jitters. Further, the results from the inverse dynamics analysis is plotted (see Fig. 7.6). To study the effect of redundant constraints on the quality of numerical solution, two different cases of this mechanism with same physical dimensions are studied: one with SU pair and the other SS pair on the connecting rod. As one can expect, the mechanism with SU pair should match better with the analytical solver. However, we found that the quality of the numerical solution also depends on whether the inertia term around the rod's redundant axis (highlighted in red in Fig. 7.4b) is defined. Fig. 7.6 (a) shows the best match between the analytical solution and numerical solution obtained from ADAMS for a model with SU pair on the rod and inertia around rod's axis is set to zero. The solution becomes unstable when there is a non-zero inertia defined for the rod around this axis (Fig. 7.6 (b)). Fig. 7.6 (c) and (d) demonstrate the case with SS pair on the rod with and without

zero inertia. Both of these plots are subjected to numerical jitters. Further, it was found that the computational performance of the analytical solver was much better than the numerical solver. Thus, it can be concluded that analytical solutions to loop closure constraints, when available, always outperform their numerical counterparts.

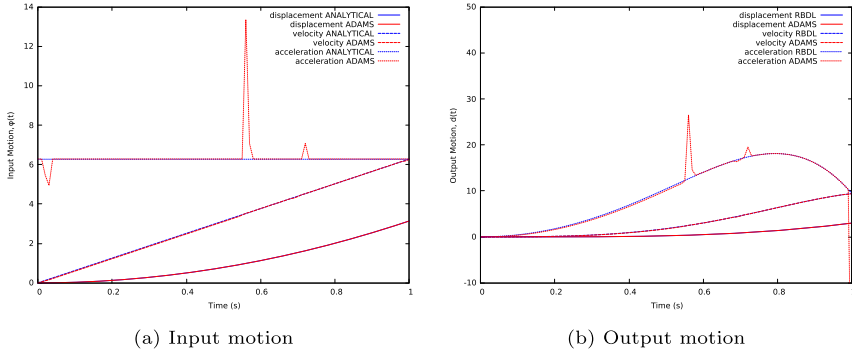


Figure 7.5 Input and output motion of 3D slider crank mechanism in HyRoDyn and ADAMS [5].

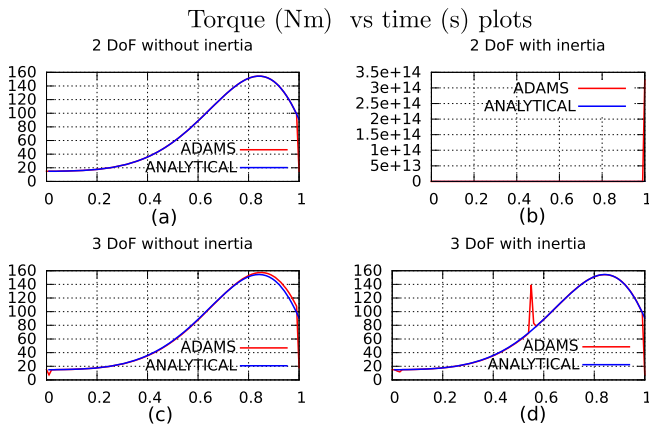


Figure 7.6 Analytical (HyRoDyn) and numerical (ADAMS) inverse dynamics analyzes of 3D slider crank mechanism [5].

Overall, both numerical and analytical approaches may be used in solving loop closure functions. The numerical and analytical formulations have their advantages and disadvantages, which are summarized in Table 7.3. Here, (+) indicates the advantages and (-) indicates the disadvantages. It can be seen that numerical and analytical formulations complement each

other. Hence, this motivates a modular approach to combine the numerical and analytical formulations for resolving loop closure equations.

Table 7.3 Numerical v/s analytical resolution of loop constraints [4].

Numerical Approach	Analytical Approach
Arbitrary mechanism can be solved numerically without any human effort (+)	Arbitrary mechanisms cannot be solved. Expert human knowledge required. (-)
Loop closure errors exists (-)	No loop closure errors (+)
Computationally slow (-)	Computationally fast (+)

7.2 Notion of modularity

Series-parallel hybrid robots are robots that can be seen as a serial composition of serial or parallel submechanisms modules. Table 2 and 3 in [11] present a survey on closed-loop kinematics and parallel submechanism modules, which have been utilized in series-parallel hybrid robot designs in the last decade. Such a design methodology is biologically inspired (see Fig. 7.7), as most joints found in the biological systems are actuated with muscle groups in a parallel architecture. This allows the exploitation of the non-linear transmission from the actuation space to the task space and provide better actuator placement possibilities that can optimize the mass-inertia properties of the limbs. Two observations can be made from this survey:

- Submechanism modules are used as a mechanical generator of a motion subspace (revolute, universal, spherical, free joint, etc.)
- The same type of submechanism with different physical parameters is utilized as a module to serve different purposes (ankle, wrist, torso joints, etc.) in the same robot.

The analysis of closed loop mechanisms is difficult and hence they require special treatments in contrast to tree type systems. Numerical resolution of the loop closure constraints can lead to inaccuracy and poor performance, as described in the previous section. Analytical resolution of loop closure constraints is worth the effort if it can be made reusable as this can help in dealing with a large number of similar loop constraints inside a hybrid robot. This inspires an analytical treatment of loop closure constraints for a submechanism module in a robot assembly, so that the modularity in the hardware design is also reflected in the kinematic and dynamic modeling of the robot.

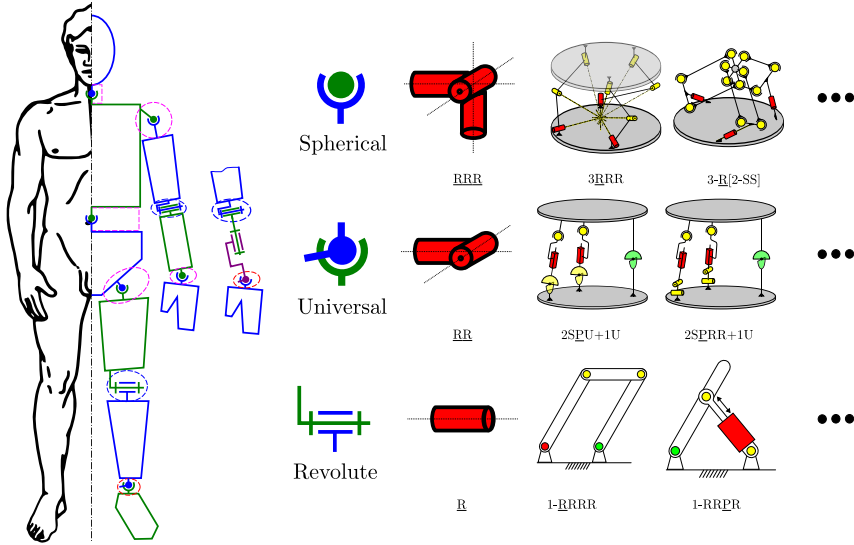


Figure 7.7 Biologically inspired series-parallel hybrid humanoid design [11] (CC BY 4.0).

7.2.1 Definition of submechanism module

A submechanism module (\mathcal{M}_i) is defined as a set of links and joints which can produce any motion from m -dimensional motion subspace of $SE(3)$ while demonstrating properties of a *minimal loop cluster*. A loop cluster is any set of loops with the property that no loop within the cluster is coupled to any other loop outside it; and a minimal loop cluster is the one that cannot be divided into two small loop clusters. In other words, these loop clusters do not share a common edge (joint). A serial combination of links and joints can be seen as a submechanism with no closed loops. This definition helps us in two different ways:

- it reveals the block diagonal structure in the constraint Jacobian matrix (\mathbf{G}) and its derivative ($\dot{\mathbf{G}}$) which can lead to efficiency in kinematics and dynamics computations,
- and, it helps in generation of simpler abstract mechanisms, which do not further contain any closed loops and have physical meaning.

To define a submechanism module, it is crucial to define three subsets of joints from the connectivity graph (\mathcal{G}_i) of a mechanism:

- spanning tree joints ($\mathbf{q}_i \in \mathbb{R}^{n_i}$): all the joints belonging to the spanning tree (\mathcal{T}_i) chosen by regular numbering scheme,

- independent joints ($\mathbf{y}_i \in \mathbb{R}^{m_i}$): a set of independent variables selected such that \mathbf{y}_i defines \mathbf{q}_i uniquely,
- active joints ($\mathbf{u}_i \in \mathbb{R}^{p_i}$): all the joints containing the actuators.

Let us define a selection matrix, $\mathbf{Q}_i \in \mathbb{R}^{p_i \times n_i}$ such that $\mathbf{u}_i = \mathbf{Q}_i \mathbf{q}_i$. Also, in the scope of the current work, it is assumed the submechanism modules are *properly actuated* (i.e., $p_i = m_i$) and *properly constrained* (i.e., $m_i = n_i - n_{ci}$).

7.2.2 Guidelines for selection

While the definition of a submechanism module is intentionally chosen not to be too restrictive, some guidelines can be followed while selecting the submechanism modules:

- It is evident that the choice of independent coordinates for a parallel submechanism module is non-unique. The independent joints in a submechanism module should be chosen such that they represent the operational space or platform coordinates of the submechanism module because it is easier to solve the inverse kinematics of parallel mechanisms analytically in comparison to forward kinematics. For serial submechanism modules, the choice of independent coordinates must be the same as its configuration space coordinates.
- A careful choice of submechanism module can help in exploiting the hierarchy in the robot design through abstractions. The parallel submechanism modules used in the design of series-parallel hybrid robots always encapsulate a known joint type (e.g., revolute, universal, spherical, etc.). Overall, the serial nature of hybrid robot design may simply be following well-studied 6 or 7 DOF anthropomorphic limb designs such as URS, SRS, etc. Hence, analytical inverse kinematics solutions for such limbs can be easily mapped to the series-parallel hybrid robot, therefore enabling the analytical computation of kinematics and dynamics of the complete system.

Example 7.4. The designer may construct the ankle joint of a humanoid robot either using a set of two serially connected actuators (e.g., 2R orthogonal arrangement) for the sake of simplicity or utilize a parallel mechanism (e.g., 2SPRR+1U [12]) for minimizing the lower leg inertia and exploiting the non-linear transmission. Fig. 7.8 shows the two design possibilities and the corresponding definitions of the submechanism modules.

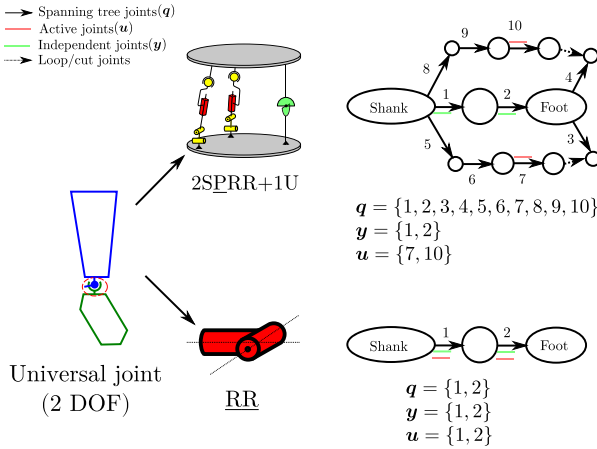


Figure 7.8 Example of submechanism definition (Used with permission of American Society of Mechanical Engineers ASME, from [1,2]; permission conveyed through Copyright Clearance Center, Inc.).

7.3 Topological modeling

A submechanism module is modeled using a two-terminal graph (TTG), which is defined as a graph with two distinguished vertices, called source and sink. The source and sink basically represent a fixed transformations to a submechanism interface point (e.g., physical screws or nut-bolt pair) in the overall assembly of the hybrid robot. In addition to the source and sink links, base and end-effector links on the submechanism are identified on the TTG, which is important for local resolution of loop closure constraints inside the submechanism module. Further, a spanning tree is deduced from this TTG, where all the nodes except for the ones connected using fixed joint are numbered using regular numbering scheme described in the previous chapter. It is additionally ensured that the main branch or trunk of the tree connecting the base and EE link is numbered first and then the additional branches are numbered. We call this extension of regular numbering scheme as *modular graph enumeration* scheme.

The series composition $\mathcal{G} = \mathcal{G}^x \oplus \mathcal{G}^y$ of two TTGs \mathcal{G}^x and \mathcal{G}^y is a TTG created from the disjoint union of graphs \mathcal{G}^x and \mathcal{G}^y by merging the sink of \mathcal{G}^x with the source of \mathcal{G}^y . The source of \mathcal{G}^x becomes the source of \mathcal{G} and the sink of \mathcal{G}^y becomes the sink of \mathcal{G} [13]. A series-parallel hybrid robot can be seen as series composition of various submechanism modules represented by their respective graphs. Let \mathcal{T}^i denote the spanning tree of

a submechanism module and s serial or parallel submechanism modules are joined in series to compose a series-parallel hybrid robot. The spanning tree of this hybrid robot (\mathcal{T}) is given by:

$$\mathcal{T} = \bigoplus_{i=1}^s \mathcal{T}^i. \quad (7.40)$$

The three joint variable sets, namely spanning tree joints, independent joints and active joints set for the hybrid robot can be composed as: $\mathbf{q}^T = (\mathbf{q}_1^T, \dots, \mathbf{q}_s^T)$, $\mathbf{y}^T = (\mathbf{y}_1^T, \dots, \mathbf{y}_s^T)$, $\mathbf{u}^T = (\mathbf{u}_1^T, \dots, \mathbf{u}_s^T)$.

7.3.1 Bottom-up composition

The topological model of a complex series-parallel hybrid robot can be composed from topological models of its submechanism modules as shown in Eq. (7.40). However, it must be noted that the composition operation is non-commutative, i.e., $\mathcal{G}^1 \oplus \mathcal{G}^2 \neq \mathcal{G}^2 \oplus \mathcal{G}^1$. Once, the submechanism graphs have been combined, the fixed joints involved can be removed. Provided the graph descriptions of individual modules, the graph description of the composed mechanism can be easily deduced. Let (p_1, s_1) denote the predecessor array and successor arrays representing \mathcal{G}^1 and (p_2, s_2) denote the predecessor array and successor arrays representing \mathcal{G}^2 . Let ι denote the submechanism interface link on \mathcal{G}^1 , then the predecessor array of the composition p is given by:

$$p = \{p_1, p'_2\} \quad \text{where} \quad \forall i \in \{1, \dots, |p_2|\}, p'_2(i) = \begin{cases} \iota : p_2(i) = 0 \\ p_2(i) + \max(s_1) : p_2(i) \neq 0 \end{cases} \quad (7.41)$$

and the successor array s of the composition is given by:

$$s = \{s_1, s'_2\} \quad \text{where} \quad \forall i \in \{1, \dots, |s_2|\}, s'_2(i) = s_2(i) + \max(s_1). \quad (7.42)$$

From predecessor and successor arrays, other properties such as parent array λ and child array μ can be easily calculated. This way of modeling is preferable when a complex model is being built up from scratch so that each submechanism model can be unit-tested beforehand, or when there are well-defined attachment interfaces on all the modules involved during the composition.

7.3.2 Top-down decomposition

During topological modeling, there might sometimes be a need of decomposing a complex series-parallel hybrid model into models of its submechanism modules. This can also be done easily thanks to the modular graph enumeration scheme. If a topological graph \mathcal{G} decomposes into \mathcal{G}^1 and \mathcal{G}^2 , let ι denote the submechanism interface link on \mathcal{G} , then the successor array splits into s_1 and s'_2 such that $\min(\mu(\iota))$ is the first element of s'_2 . Then, s_2 can be computed as $s_2(i) = s'_2(i) - \max(s_1) \quad \forall i \in \{1, \dots, |s'_2|\}$. The predecessor array p will split into p_1 and p'_2 such that ι is the first element of p'_2 . And the predecessor array p_2 can be computed as:

$$\forall i \in \{1, \dots, |p'_2|\}, p_2(i) = \begin{cases} 0 : p'_2(i) = \iota \\ p'_2(i) - \max(s_1) : p'_2(i) \neq \iota \end{cases} \quad (7.43)$$

This kind of topological decomposition is needed when a complex system model is already at hand and the topological models of the submechanism modules are to be derived.

Example 7.5. Fig. 7.9 shows the composition of a series-parallel hybrid humanoid leg with the help of four submechanism modules. The first module \mathcal{T}^1 is an RR module and consists of Hip1 and Hip2 joints, second module \mathcal{T}^2 is a 1-RRPR module used for Hip3 joint, third module \mathcal{T}^3 is again 1-RRPR module used for Knee joint, and the fourth module \mathcal{T}^4 is 2-SPRR+1U mechanism used for the ankle joint. The respective graphs of the three distinct submechanism modules are provided in Fig. 7.10. \mathcal{T}^2 is attached to \mathcal{T}^1 at link 2, \mathcal{T}^3 is attached to the link 3 of $\mathcal{T}^1 \oplus \mathcal{T}^2$ and \mathcal{T}^4 is attached to the link 4 of $\mathcal{T}^1 \oplus \mathcal{T}^2 \oplus \mathcal{T}^3$. The submechanism interface links are collected in an submechanism interface array $\iota = \{0, 2, 3, 4\}$. The graph description of the overall series-parallel hybrid composition is given by:

$$\begin{aligned} p &= \{0, 1, 2, 2, 4, 3, 3, 7, 6, 9, 10, 10, 6, 13, 14, 6, 16, 17\} \\ s &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\} \\ \lambda &= \{0, 1, 2, 2, 4, 3, 3, 7, 6, 9, 10, 10, 6, 13, 14, 6, 16, 17\} \\ \mu &= \{\{1\}, \{2\}, \{3, 4\}, \{6, 7\}, \{5\}, \{\}, \{9, 13, 16\}, \{8\}, \{\}, \{10\}, \\ &\quad \{11, 12\}, \{\}, \{\}, \{14\}, \{15\}, \{\}, \{17\}, \{18\}, \{\} \end{aligned} \quad (7.44)$$

and the graph enumerated using the modular numbering scheme is shown in Fig. 7.12.

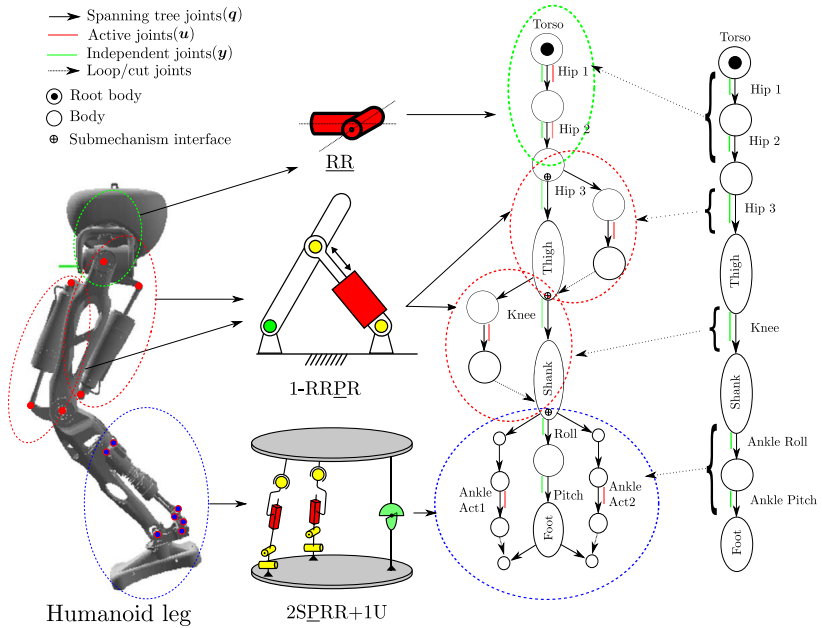


Figure 7.9 Example of series-parallel hybrid composition (Used with permission of American Society of Mechanical Engineers ASME, from [1,2]; permission conveyed through Copyright Clearance Center, Inc.).

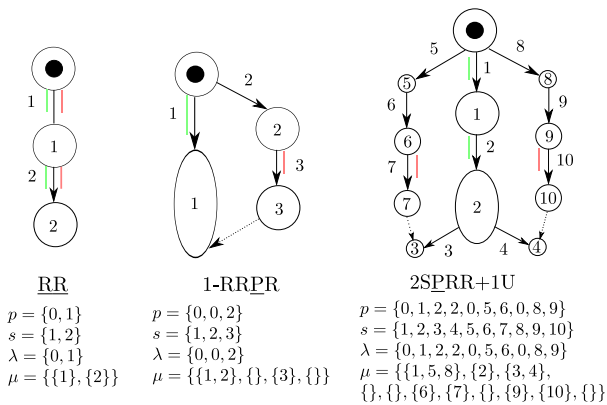


Figure 7.10 Graphs of submechanism modules [5].

7.4 Kinematics and dynamics

7.4.1 Submechanism module

7.4.1.1 Serial submechanisms

For serial submechanism modules, the independent variables in the spanning tree are the same as generalized coordinates of the spanning tree, i.e., $\mathbf{q} = \boldsymbol{\gamma}$, $\dot{\mathbf{q}} = \dot{\boldsymbol{\gamma}}$, and $\ddot{\mathbf{q}} = \ddot{\boldsymbol{\gamma}}$. The loop closure function for a serial submechanism module is given by:

$$\begin{aligned} \mathbf{q} &= \boldsymbol{\gamma}(\boldsymbol{y}) = \boldsymbol{y}, \\ \mathbf{G} &= \mathbf{I}, \\ \mathbf{g} &= \mathbf{0}, \end{aligned} \tag{7.45}$$

where \mathbf{I} is an identity matrix of size $(n \times n)$ and \mathbf{g} is a zero vector of size $(n \times 1)$. Since we assume that all the submechanism modules are properly actuated, the matrix $\mathbf{G}_i = \mathbf{I}$ is also an identity matrix of size $(n \times n)$.

7.4.1.2 Parallel submechanisms with known analytical solutions

Problems related to geometry or kinematics for parallel robots is usually easy to formulate but difficult to solve, because they result in a set of non-linear algebraic equations which need careful analysis and treatment. The three most useful solution techniques to deal with such problems are polynomial continuation, Gröbner bases, and elimination method [14]. To the best knowledge of the author, there is no universally applicable way to solve kinematics problems in case of parallel robots. Hence, the geometer or kinematician may choose any formulation and solution method which works the best for a specific type of parallel robot and arrive at the loop closure function. The loop closure functions for parallel submechanism modules are defined using Eq. (7.34). The foundations for deriving LCFs for 1-RRPR and 2-SPRR+1U mechanisms used in the humanoid leg example can be found in Chapter 3 [7] and Chapter 4 [12] respectively and are skipped here for brevity.

7.4.1.3 Arbitrary parallel submechanisms

There might be cases of parallel submechanisms where the analytical solutions to loop closure constraints are not yet available. In this case, it is possible to extract explicit constraints $(\boldsymbol{\gamma}, \mathbf{G}, \mathbf{g})$ from implicit constraints $(\boldsymbol{\phi}, \mathbf{K}, \mathbf{k})$ numerically. (See Fig. 7.11.)

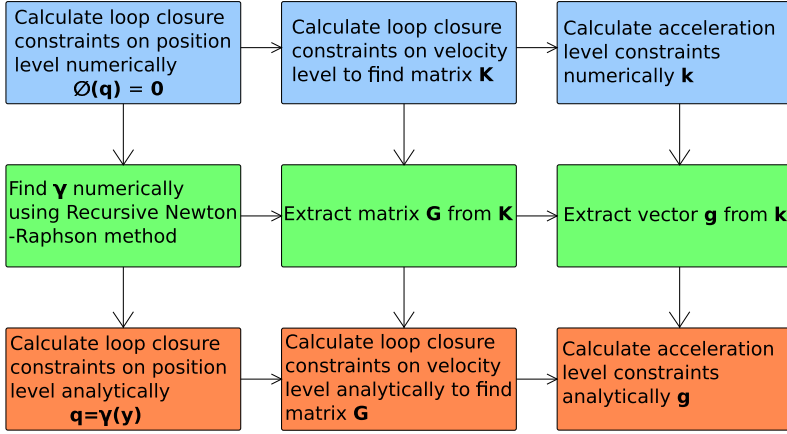


Figure 7.11 Conversion of implicit constraints to explicit constraints [4].

Extraction of explicit Jacobian matrix from implicit Jacobian matrix

Using RBDL function *CalcConstraintsJacobian()*, the constraint Jacobian matrix in implicit form \mathbf{K} is computed. Matrix \mathbf{K} can be rewritten by splitting it into independent and dependent coordinates part. Dependent coordinates is a subset of generalized positions that are expressed as a function of independent coordinates $\mathbf{y} = \mathbf{q}_i \in \mathbb{R}^m$ using function $\boldsymbol{\gamma}$.

$$\mathbf{K}\dot{\mathbf{q}} = \mathbf{0}, \quad (7.46)$$

$$\begin{bmatrix} \mathbf{K}_i & \mathbf{K}_d \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_i \\ \dot{\mathbf{q}}_d \end{bmatrix} = \mathbf{0}, \quad (7.47)$$

where

- \mathbf{K}_i is the independent coordinate part of matrix \mathbf{K} . The size of the matrix is $n^c \times m$ matrix, where n^c is number of constraints due to the cut joint in the closed loop system.
- \mathbf{K}_d is the dependent coordinate part of matrix \mathbf{K} . The size of the matrix is $n^c \times (n - m)$ matrix. Note that it is a square matrix.
- $\dot{\mathbf{q}}_i = \dot{\mathbf{y}}$ is the generalized velocity vector of size (m) in independent coordinates.
- $\dot{\mathbf{q}}_d$ is the generalized velocity vector of size $(n - m)$ in dependent coordinates.

Expanding the previous equation,

$$\mathbf{K}_i \dot{\mathbf{y}} + \mathbf{K}_d \dot{\mathbf{q}}_d = 0, \quad (7.48)$$

$$\dot{\mathbf{q}}_d = -\mathbf{K}_d^{-1} \mathbf{K}_i \dot{\mathbf{y}}. \quad (7.49)$$

Or, in matrix form,

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}_i \\ \dot{\mathbf{q}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_d^{-1} \mathbf{K}_i \end{bmatrix} \dot{\mathbf{y}}. \quad (7.50)$$

Also, $\dot{\mathbf{q}} = \mathbf{G} \dot{\mathbf{y}}$. Therefore, the constraint Jacobian matrix in explicit matrix can be written as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_d^{-1} \mathbf{K}_i \end{bmatrix}. \quad (7.51)$$

To split the matrix \mathbf{K} into independent and dependent parts, selection matrices \mathbf{Q}_i and \mathbf{Q}_d are defined respectively. The independent joints selection matrix \mathbf{Q}_i is of size $(m \times n)$ and dependent joints selection matrix \mathbf{Q}_d is of size $((n - m) \times n)$.

The algorithm to extract \mathbf{G} from \mathbf{K} is given in Algorithm 7.1.

Extraction of explicit bias acceleration from implicit bias acceleration

This section presents the extraction of explicit bias acceleration vector \mathbf{g} from implicit bias acceleration vector \mathbf{k} . For bias accelerations in implicit form \mathbf{k} , the RBDL function *CalcConstrainedSystemVariables()* is used. To compute only the bias acceleration, the function is rewritten as *calc_k()*. The function take the input as model, generalized positions, generalized velocities and constraint set. It return the bias acceleration in implicit form \mathbf{k} .

Differentiating the velocity equation $\mathbf{K} \dot{\mathbf{q}} = \mathbf{0}$,

$$\mathbf{K} \ddot{\mathbf{q}} + \dot{\mathbf{K}} \dot{\mathbf{q}} = \mathbf{0}. \quad (7.52)$$

The bias acceleration in implicit form is given by

$$\mathbf{k} = -\dot{\mathbf{K}} \dot{\mathbf{q}}. \quad (7.53)$$

As in the previous section, the equation can be rewritten in terms of dependent and independent parts.

$$\begin{bmatrix} \mathbf{K}_i & \mathbf{K}_d \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_i \\ \ddot{\mathbf{q}}_d \end{bmatrix} = -\dot{\mathbf{K}}\dot{\mathbf{q}}, \quad (7.54)$$

$$\mathbf{K}_i\ddot{\mathbf{y}} + \mathbf{K}_d\ddot{\mathbf{q}}_d = -\dot{\mathbf{K}}\dot{\mathbf{q}}. \quad (7.55)$$

Multiplying by \mathbf{K}_d^{-1} ,

$$\mathbf{K}_d^{-1}\mathbf{K}_i\ddot{\mathbf{y}} + \ddot{\mathbf{q}}_d = -\mathbf{K}_d^{-1}\dot{\mathbf{K}}\dot{\mathbf{q}}, \quad (7.56)$$

$$\ddot{\mathbf{q}}_d = -\mathbf{K}_d^{-1}\mathbf{k} - \mathbf{K}_d^{-1}\mathbf{K}_i\ddot{\mathbf{y}}. \quad (7.57)$$

Now, as $\ddot{\mathbf{q}} = \mathbf{G}\ddot{\mathbf{y}} + \mathbf{g}$, substituting \mathbf{G} from Eq. (7.51) and comparing with the previous equation,

$$\mathbf{g} = \begin{bmatrix} \mathbf{0} \\ \mathbf{K}_d^{-1}\mathbf{k} \end{bmatrix}. \quad (7.58)$$

The algorithm to extract \mathbf{g} from \mathbf{k} is given in Algorithm 7.2.

Algorithm 7.1 Calculate \mathbf{G} from \mathbf{K} [4].

(in) \mathbf{K}

(out) \mathbf{G}

- 1: $\mathbf{K}_i = \mathbf{K}\mathbf{Q}_i^T$
 - 2: $\mathbf{K}_d = \mathbf{K}\mathbf{Q}_d^T$
 - 3: $\mathbf{G}_d = \mathbf{K}_d^{-1}\mathbf{K}_i$
 - 4: $\mathbf{G} \leftarrow \begin{bmatrix} \mathbf{I} \\ \mathbf{G}_d \end{bmatrix}$
-

Algorithm 7.2 Calculate \mathbf{g} from (\mathbf{K}, \mathbf{k}) [4].

(in) Implicit constraints \mathbf{K}, \mathbf{k}

(out) Explicit bias acceleration \mathbf{g}

- 1: $\mathbf{K}_d = \mathbf{K}\mathbf{Q}_d^T$
 - 2: $\mathbf{g}_d = \mathbf{K}_d^{-1}\mathbf{k}$
 - 3: $\mathbf{g}_i = \mathbf{0}$
 - 4: $\mathbf{g} \leftarrow \begin{bmatrix} \mathbf{g}_i \\ \mathbf{g}_d \end{bmatrix}$
-

7.4.1.4 Equations of motion of a submechanism module

The EOM for the equivalent spanning tree (\mathcal{T}) of a submechanism module subjected to analytical loop closure constraints ($\mathbf{G}^T \boldsymbol{\tau}_{ci} = 0$) is given by

$$\mathbf{G}^T \mathbf{M} \mathbf{G} \ddot{\mathbf{y}} + \mathbf{G}^T (\mathbf{C} + \mathbf{M} \mathbf{g}) = \mathbf{G}^T \boldsymbol{\tau}. \quad (7.59)$$

It may be noted that Eq. (7.59) has the same algebraic form as the equation of motion for the unconstrained system in Eq. (7.26) and hence can be written in the form:

$$\hat{\mathbf{M}} \ddot{\mathbf{y}} + \hat{\mathbf{C}} = \boldsymbol{\tau}_y, \quad (7.60)$$

where $\hat{\mathbf{M}} = \mathbf{G}^T \mathbf{M} \mathbf{G}$ is module's ($m \times m$) mass-inertia matrix which is symmetric and positive-definite and $\hat{\mathbf{C}} = \mathbf{G}^T (\mathbf{C} + \mathbf{M} \mathbf{g})$ is its ($m \times 1$) vector of bias forces and $\boldsymbol{\tau}_y = \mathbf{G}^T \boldsymbol{\tau}$ is the ($m \times 1$) vector of generalized forces for the submechanism module. The actuator forces $\boldsymbol{\tau}_u$ for the submechanism module can be computed using Eq. (7.39), where $\mathbf{G}_u = \mathbf{Q} \mathbf{G}$.

7.4.2 Series-parallel hybrid composition

The loop-closure function for the series-parallel hybrid robot is composed as follows.

$$\begin{aligned} \boldsymbol{\gamma} &= \left[\boldsymbol{\gamma}_1^T \quad \dots \quad \boldsymbol{\gamma}_i^T \quad \dots \quad \boldsymbol{\gamma}_s^T \right]_{(n \times 1)}^T \\ \mathbf{G} &= \begin{bmatrix} \mathbf{G}_1 & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \dots & \mathbf{G}_i & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{G}_s \end{bmatrix}_{(n \times m)} \\ \mathbf{g} &= \left[\mathbf{g}_1^T \quad \dots \quad \mathbf{g}_i^T \quad \dots \quad \mathbf{g}_s^T \right]_{(n \times 1)}^T \end{aligned} \quad (7.61)$$

From Eq. (7.61), three observations about \mathbf{G} can be made: it typically contains many zeros due to branch induced sparsity, it contains various identity matrix blocks corresponding to serial submechanism modules, and it has a block diagonal nature due to modular choice of spanning tree. These properties of the matrix can be used to save some computational costs occurring in sparse matrix multiplications. Fig. 7.12 shows the block-diagonal nature of the loop closure Jacobian for the series-parallel hybrid leg. The actuator Jacobian matrix \mathbf{G}_u also has a block diagonal nature that can be exploited

when computing its inverse, as shown in Eq. (7.62).

$$\begin{aligned}
 \mathbf{G}_u = \mathbf{Q}\mathbf{G} = & \begin{bmatrix} \mathbf{G}_{u1} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{G}_i & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{G}_{uS} \end{bmatrix} \quad (m \times m) \\
 \mathbf{G}_u^{-1} = & \begin{bmatrix} \mathbf{G}_{u1}^{-1} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{G}_{ii}^{-1} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{G}_{uS}^{-1} \end{bmatrix} \quad (m \times m)
 \end{aligned} \tag{7.62}$$

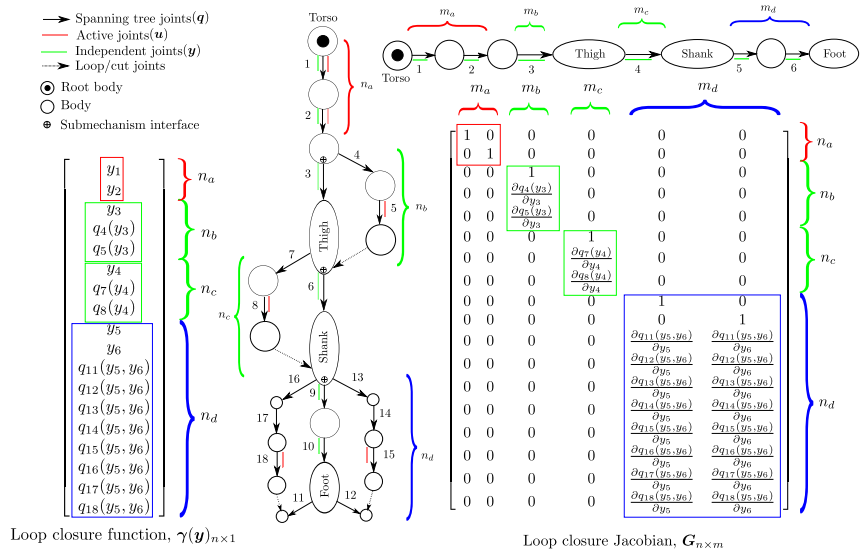


Figure 7.12 Modular composition of loop-closure function (Used with permission of American Society of Mechanical Engineers ASME, from [1,2]; permission conveyed through Copyright Clearance Center, Inc.).

Algorithm 7.3 presents an algorithm to compute the position, velocity and acceleration state of the spanning tree from the loop closure function exploiting these properties. Once, the full system state of the equivalent spanning tree is known, it is straightforward to compute the pose of any

Algorithm 7.3 Spanning tree state (SYSSTATE) [1,2].

(in) Independent joint state $(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}})$
(out) Spanning tree joint state $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$

- 1: **function** SYSSTATE($\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$)
- 2: **for** $\mathcal{M}_i \in (\mathcal{M}_1, \dots, \mathcal{M}_s)$ **do**
- 3: **if** $m_i \neq n_i$ **then** ▷ Check if module is parallel
- 4: $\mathbf{q}_i \leftarrow \boldsymbol{\gamma}_i(\mathbf{y}_i)$
- 5: $\dot{\mathbf{q}}_i \leftarrow \mathbf{G}_i \dot{\mathbf{y}}_i$
- 6: $\ddot{\mathbf{q}}_i \leftarrow \mathbf{G}_i \ddot{\mathbf{y}}_i + \mathbf{g}_i$
- 7: **else**
- 8: $\mathbf{q}_i \leftarrow \mathbf{y}_i, \dot{\mathbf{q}}_i \leftarrow \dot{\mathbf{y}}_i, \ddot{\mathbf{q}}_i \leftarrow \ddot{\mathbf{y}}_i$
- 9: $\mathbf{q} \leftarrow \mathbf{y}_i, \dot{\mathbf{q}} \leftarrow \dot{\mathbf{y}}_i, \ddot{\mathbf{q}} \leftarrow \ddot{\mathbf{y}}_i \forall i \in [1, \dots, s]$
- 10: **return** $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$

point on the spanning tree as well as quantities like point Jacobian, spatial twists and acceleration, etc.

Algorithm 7.4 Inverse Dynamics (IDYN) [1,2].

(in) Independent joint state $(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}})$ and system model (\mathcal{M})
(out) Actuator forces $(\boldsymbol{\tau}_u)$

- 1: **function** IDYN($\mathcal{M}, \mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$)
- 2: $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}] \leftarrow \text{SYSSTATE}(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}})$ ▷ Algorithm 7.3
- 3: $\boldsymbol{\tau} \leftarrow \text{RNEA}(\mathcal{M}, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ ▷ Inv. dyn. of tree, Algorithm 7.5
- 4: **for** $\mathcal{M}_i \in (\mathcal{M}_1, \dots, \mathcal{M}_s)$ **do**
- 5: **if** $m_i \neq n_i$ **then** ▷ Check if module is parallel
- 6: $\boldsymbol{\tau}_{ui} \leftarrow \mathbf{G}_{ui}^{-1} \mathbf{G}_i^T \boldsymbol{\tau}_i$
- 7: **else**
- 8: $\boldsymbol{\tau}_{ui} \leftarrow \boldsymbol{\tau}_i$
- 9: $\boldsymbol{\tau}_u \leftarrow (\boldsymbol{\tau}_{ui} : 1 \leq i \leq s)$
- 10: **return** $\boldsymbol{\tau}_u$

With $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ known by solving the loop constraints determined by Eq. (7.34), only Eqs. (7.26) of the unconstrained system need to be evaluated. This is most efficiently carried out by a recursive $O(n)$ algorithm. Various of such have been proposed in the literature. The actual computational effort depends on the representation of spatial twists and wrenches. The spatial representation [7,10] is deemed as the most efficient representation.

Algorithm 7.5 Modular Recursive Newton Euler Algorithm (RNEA) [1, 2].

(in) Spanning tree joint state $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, vector of external wrenches \mathbf{W}^{ext} and system model (\mathcal{M})

(out) Tree joint forces $(\boldsymbol{\tau})$

```

1: function RNEA( $\mathcal{M}, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ )
2:    $\mathbf{V}_{0,0} = \mathbf{0}$  ▷ Velocity of root link
3:    $\dot{\mathbf{V}}_{0,0} = -\dot{\mathbf{V}}_g$  ▷ Gravity vector
4:   for  $i \in (1, \dots, s)$  do ▷ Inter-modular F.R
5:     for  $k \in (1, \dots, n_i)$  do ▷ Intra-modular F.R
6:        $\mathbf{V}_{i,k} = \mathbf{V}_{i,\lambda_i(k)} + \mathbf{J}_{i,k} \dot{\mathbf{q}}_{i,k}$ 
7:        $\dot{\mathbf{V}}_{i,k} = \dot{\mathbf{V}}_{i,\lambda_i(k)} + \mathbf{J}_{i,k} \ddot{\mathbf{q}}_{i,k} + \mathbf{ad}_{\mathbf{V}_{i,\lambda_i(k)}} \mathbf{V}_{i,k}$ 
8:     for  $i \in (s, \dots, 1)$  do ▷ Inter-modular B.R
9:       for  $k \in (n_i, \dots, 1)$  do ▷ Intra-modular B.R
10:       $\mathbf{W}_{i,k} = \sum_{j \in \mu_i(k)} \mathbf{W}_{i,j} + \mathbf{M}_{i,k} \dot{\mathbf{V}}_{i,k} - \mathbf{ad}_{\mathbf{V}_{i,k}}^T \mathbf{M}_{i,k} \mathbf{V}_{i,k} + \mathbf{W}_{i,k}^{\text{ext}}$ 
11:       $\boldsymbol{\tau}_{i,k} = \mathbf{J}_{i,k}^T \mathbf{W}_{i,k}$ 
12:   return  $\boldsymbol{\tau}$ 

```

For the i th submechanism module, denote with $\mathbf{V}_{i,k} = (\boldsymbol{\omega}^T, \mathbf{v}^T)^T \in se(3)$ the twist vector of body k , and with $\mathbf{J}_{i,k}$ the instantaneous screw coordinate vector of joint k , both in spatial representation. The recursive Newton–Euler algorithm (RNEA) [7,15] consists of a forward recursion ($k = 1, \dots, n_i$), where the spatial state of the system is computed from the generalized coordinates, velocities, and accelerations $(\mathbf{q}_i, \dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_i)$.

$$\begin{aligned} \mathbf{V}_{i,k} &= \mathbf{V}_{i,\lambda_i(k)} + \mathbf{J}_{i,k} \dot{\mathbf{q}}_{i,k}, \\ \dot{\mathbf{V}}_{i,k} &= \dot{\mathbf{V}}_{i,\lambda_i(k)} + \mathbf{J}_{i,k} \ddot{\mathbf{q}}_{i,k} + \mathbf{ad}_{\mathbf{V}_{i,\lambda_i(k)}} \mathbf{V}_{i,k}. \end{aligned} \quad (7.63)$$

The backward recursion ($k = n_i, \dots, 1$) in RNEA computes the generalized forces $\boldsymbol{\tau}_i$ from the spatial state of the system. To this end, denote with $q_{i,k}$ and $\tau_{i,k}$ the k th element of \mathbf{q}_i and $\boldsymbol{\tau}_i$ of submodule i .

$$\begin{aligned} \mathbf{W}_{i,k} &= \sum_{j \in \mu_i(k)} \mathbf{W}_{i,j} + \mathbf{M}_{i,k} \dot{\mathbf{V}}_{i,k} - \mathbf{ad}_{\mathbf{V}_{i,k}}^T \mathbf{M}_{i,k} \mathbf{V}_{i,k} + \mathbf{W}_{i,k}^{\text{app}} \\ \boldsymbol{\tau}_{i,k} &= \mathbf{J}_{i,k}^T \mathbf{W}_{i,k}. \end{aligned} \quad (7.64)$$

Here, $V_{i,k}$, $M_{i,k}$, $W_{i,k}^{\text{app}}$ are the spatial twist, the inertia matrix, and the applied wrench of body k in submodule i , respectively. The 6×6 matrix

$$\mathbf{ad}_V = \begin{pmatrix} \tilde{\omega} & \mathbf{0} \\ \tilde{v} & \tilde{\omega} \end{pmatrix} \quad (7.65)$$

is called the screw product or spatial cross product operator. Here, $\tilde{\omega}$ and \tilde{v} represent the skew symmetric matrices associated with the vectors ω and v , respectively.

An algorithm to solve the inverse dynamics of the series-parallel hybrid composition is provided in Algorithm 7.4 which takes as input the motion described in independent coordinates (y, \dot{y}, \ddot{y}) . First, the full state of the spanning tree is computed (Line 2) with the help of Algorithm 7.3. The next step is to compute the joint forces (τ) for the spanning tree, which is done using a modular form of RNEA presented in Algorithm 7.5. In this algorithm, the forward and backward recursions are carried out first within the module (intra-modular) and then across the modules (inter-modular) to compute the spatial state of the spanning tree (using Eq. (7.63)) and tree joint forces (using Eq. (7.64)). And lastly, the tree joint forces are converted to the actuator forces of the series-parallel hybrid robot using an inter-modular recursion (see Line 6 and Line 8) by exploiting the block-diagonal nature of loop closure Jacobian matrix.

7.4.3 Computational effort

In general, it is more difficult to analyze the computational performance of dynamics algorithms for systems with closed loops, because the closed form solutions to the loop closure constraints might not always exist and the number of floating point operations involved are highly dependent on the geometry of the parallel mechanism. Nevertheless, the presented modular approach leads to some cost savings occurring in matrix-vector multiplication and inversion, which will be discussed here. The tree joint forces (τ) are solved using $O(n)$ RNEA. The projection of these forces to the independent joint space of the robot, i.e., $\tau_y = \mathbf{G}^T \tau$ can be computed with mn multiplications and $m(n-1)$ additions or $2m(n-1)$ floating point operations (FLOPs). Due to block diagonal structure in \mathbf{G} , this matrix-vector multiplication can be done in $2 \sum_{i=1}^s m_i(n_i-1)$ FLOPs. It is trivial to show that $2 \sum_{i=1}^s m_i(n_i-1) \leq 2m(n-1)$, which demonstrates the cost savings involved in this step.² The projection of independent joint forces

² $\sum_{i=1}^s m_i n_i \leq \sum_{i=1}^s m_i \sum_{i=1}^s n_i \forall m_i, n_i \in \mathbb{N}$.

to actuator forces requires the inversion of actuator Jacobian matrix \mathbf{G}_u , which can be done with $O(m^3)$ complexity, and its multiplication with $\boldsymbol{\tau}_y$ vector which requires $2m(m-1)$ FLOPs. Due to the modular formulation, actuator Jacobian matrix \mathbf{G}_u also has a block diagonal structure and instead of a full inversion, its inverse can be computed simply by computing the inverse of its submatrix blocks along the diagonal. Hence, compared to $O((\sum_{i=1}^s m_i)^3)$ complexity involved in this process, the block diagonal nature leads to a reduced inversion complexity of $O(\sum_{i=1}^s m_i^3)$ as $\sum_{i=1}^s m_i^3 \leq (\sum_{i=1}^s m_i)^3 \forall m_i \in \mathbb{N}$. Lastly, the matrix-vector multiplication $\boldsymbol{\tau}_u = \mathbf{G}_u^{-1} \boldsymbol{\tau}_y$ leads to a reduced cost of $2 \sum_{i=1}^s m_i(m_i - 1)$ FLOPs as compared to $2m(m-1)$ FLOPs. These cost savings demonstrate the efficiency of the inverse dynamics algorithm due to the adopted notion of modularity.

It should be noted that different links contribute differently to actuator forces while calculating the inverse dynamics. One way to boost the computational efficiency is to isolate parallel submechanisms which do not contribute much to the inverse dynamics of the full system. A case study on such model simplification has been presented in [16].

7.5 Conclusion

This chapter presents a modular approach for kinematic and dynamic modeling of series-parallel hybrid robots. The key idea behind this approach is to see a complex hybrid mechanism as a serial composition of serial or parallel submechanism module. Once, the submechanisms in the hybrid kinematic chain are identified, the topological model for each submechanism module is derived and used to compose the topological model of the complete hybrid chain. Thanks to the modular graph enumeration during topological modeling, the associated loop closure functions can be easily composed and have a block-diagonal structure. This can be exploited in various kinematics and dynamics algorithms and leads to efficient and user-friendly models. The approach presented here forms the basis for the modular software workbench called HyRoDyn which can efficiently solve arbitrarily complex series-parallel hybrid robots. The framework now sits at the heart of the control architecture of various robots such as Recupera-Reha exoskeleton [17,18], RH5 Humanoid [19,20] and RH5 Manus [21].

References

- [1] S. Kumar, A. Mueller, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, in: Vol. 5A: 43rd Mechanisms

- and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2019, v05AT07A054, <https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2019/59230/V05AT07A054/6453674/v05at07a054-detc2019-97115.pdf>, <https://doi.org/10.1115/DETC2019-97115>.
- [2] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, *Journal of Mechanisms and Robotics* 12 (2) (2020).
 - [3] R. Kumar, S. Kumar, A. Müller, F. Kirchner, Modular and hybrid numerical-analytical approach – a case study on improving computational efficiency for series-parallel hybrid robots, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 3476–3483, <https://doi.org/10.1109/IROS47612.2022.9981474>.
 - [4] R. Kumar, A hybrid numerical and analytical approach towards resolving loop closure constraints in rigid body dynamics, <https://doi.org/10.13140/RG.2.2.14004.99207>, Aug 2021.
 - [5] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
 - [6] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, HyRoDyn: a modular software framework for solving analytical kinematics and dynamics of series-parallel hybrid robots, in: International Conference on Intelligent Robots and Systems, IROS Poster, 2018.
 - [7] R. Featherstone, *Rigid Body Dynamics Algorithm*, Springer, 2008.
 - [8] J.C. Piedboeuf, Kane's equations or Jourdain's principle?, in: Proceedings of 36th Midwest Symposium on Circuits and Systems, vol. 2, 1993, pp. 1471–1474, <https://doi.org/10.1109/MWSCAS.1993.343389>.
 - [9] A. Mueller, Implementation of a geometric constraint regularization for multibody system models, *Archive of Mechanical Engineering* 61 (2) (2014) 365.
 - [10] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*, Springer Verlag, 2011.
 - [11] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Mechatronics* 68 (2020) 102367, <https://doi.org/10.1016/j.mechatronics.2020.102367>, <https://www.sciencedirect.com/science/article/pii/S0957415820300477>.
 - [12] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Mueller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: M. Carricato (Ed.), *Advances in Robot Kinematics*, Springer Verlag GmbH, Cham, 2018.
 - [13] Eppstein, Parallel recognition of series-parallel graphs, *Information and Computation* 98 (1) (1992) 41–55, [https://doi.org/10.1016/0890-5401\(92\)90041-D](https://doi.org/10.1016/0890-5401(92)90041-D), <http://www.sciencedirect.com/science/article/pii/089054019290041D>.
 - [14] J. Nielsen, B. Roth, On the kinematic analysis of robotic mechanisms, *The International Journal of Robotics Research* 18 (12) (1999) 1147–1160, <https://doi.org/10.1177/02783649922067771>.
 - [15] A. Müller, Screw and Lie group theory in multibody dynamics, *Multibody System Dynamics* 42 (2) (2018) 219–248, <https://doi.org/10.1007/s11044-017-9583-6>.
 - [16] S. Kumar, J. Martensen, A. Mueller, F. Kirchner, Model simplification for dynamic control of series-parallel hybrid robots – a representative study on the effects of neglected dynamics shivesh, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 5701–5708, <https://doi.org/10.1109/IROS40897.2019.8967786>.
 - [17] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the recupera

- exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019) 626, <https://doi.org/10.3390/app9040626>.
- [18] S. Kumar, M. Simnofske, B. Bongardt, A. Müller, F. Kirchner, Integrating mimic joints into dynamics algorithms: exemplified by the hybrid Recupera exoskeleton, in: *Proceedings of the Advances in Robotics, AIR '17*, ACM, New York, NY, USA, 2017, pp. 27:1–27:6, <https://doi.org/10.1145/3132446.3134891>.
- [19] J. Esser, S. Kumar, H. Peters, V. Bargsten, J. de Gea Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid robot, in: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021, pp. 400–407, <https://doi.org/10.1109/HUMANOIDS47582.2021.9555770>.
- [20] I. Bergonzani, M. Popescu, S. Kumar, F. Kirchner, Fast dynamic walking with RH5 humanoid robot, in: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023, pp. 1–8, <https://doi.org/10.1109/Humanoids57100.2023.10375193>.
- [21] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 01–07, <https://doi.org/10.1109/ICRA46639.2022.9811843>.

This page intentionally left blank

CHAPTER 8

Forward dynamics with constraint embedding for dynamic simulation

Rohit Kumar^a, Shivesh Kumar^a, Andreas Müller^b, and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

^cWorking Group Robotics, University of Bremen, Bremen, Germany

8.1 Introduction

The forward dynamics involves the computation of accelerations of the multi-body system given the actuated forces

$$\ddot{\mathbf{q}} = \text{FDyn}(\dot{\mathbf{q}}, \mathbf{q}, \boldsymbol{\tau}), \quad (8.1)$$

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})), \quad (8.2)$$

where $\boldsymbol{\tau}$ is the vector of generalized forces, $\mathbf{M}(\mathbf{q})$ is the generalized mass-inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the generalized matrix of Coriolis, centrifugal and gravity efforts, and \mathbf{q} , $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are the generalized positions, velocities and accelerations vector, respectively.

It can be seen that the forward dynamics require the inversion of the mass-inertia matrix. For tree-type systems, it can be solved in mainly two ways [1],

1. Formulate equations of motion for the rigid-body system and solve for acceleration.
2. Propagate through each body such that the accelerations are computed for one joint at a time.

In the first approach, the $n \times n$ joint space mass inertia matrix is calculated for the whole system, where n is the number of spanning-tree joints in the topological graph. The whole set of equations of motion is solved for accelerations as a set of n linear equations by inversion of joint-space mass inertia matrix. These algorithms generally have the complexity as $O(n^3)$. On the other hand, the second method propagates through each body and

computes the accelerations considering joint constraints. These algorithms have the complexity of $O(n)$. The articulated body algorithm is a recursive forward dynamics algorithm for tree type systems that implements second method and was introduced in [2]. The articulated body algorithm (ABA) is efficient when the rigid body system has a large number of bodies. For smaller rigid body systems, the first approach can match or even outperform the ABA algorithm. However, for closed loops in the system, the first approach might suffer due to numerical inversion of large mass inertia matrix and is inefficient.

The idea of embedding the loop closure constraints within the ABA algorithm is introduced in [3] leading to a recursive forward dynamics for constrained systems in minimal coordinates using spatial operator algebra [4]. In the recent years, the algorithms from Featherstone [1] has become increasingly popular for tree type systems among the robotics community and their implementation can be found in many open-source libraries such as RBDL [5], Pinocchio [6], MuJoCo [7], Drake [8], Bullet [9], DART [10], etc. One of the main differences between the formulations from Featherstone and Jain is the numbering scheme. Featherstone's algorithms use a numbering scheme where the numbers increase from base to tip of the kinematic tree, while Jain's algorithms use an opposite numbering scheme where numbering increases from tip to base of the tree. Due to the popularity of Featherstone's notation and numbering scheme, the recursive forward dynamics algorithm for series-parallel hybrid robots is presented in base-to-tip numbering scheme using the constraint embedding approach.

In this chapter, Sections 8.2 and 8.3 will reiterate the concepts used in ABA from [1]. Next, Section 8.4 provides the adaptation of the constraint embedding approach and Section 8.5 shows its application on a reduced version of RH5 Manus humanoid robot. Finally, Section 8.6 shows the results and validation of the adapted constraint embedding approach.

8.2 Articulated body inertia

Articulated body inertia is a major component required in the ABA algorithm. In [1], articulated body inertia is defined as the perceived inertia by a body in the rigid body system. Consider a rigid body B in space with applied force \mathbf{f} and resulting acceleration \mathbf{a} . The equation of motion for the body is:

$$\mathbf{f} = \mathbf{I}\mathbf{a} + \mathbf{p}, \quad (8.3)$$

where \mathbf{I} and \mathbf{p} are the rigid body inertia and bias force respectively. When another body B' is attached to the body B through a joint, the body is perceived to have articulated body inertia. The articulated-body equation of motion can be written as:

$$\mathbf{f} = \mathbf{I}^A \mathbf{a} + \mathbf{p}^A, \quad (8.4)$$

where \mathbf{I}^A and \mathbf{p}^A are the articulated-body inertia and bias force of body B when B' is attached through a connecting joint respectively. B is called the *handle* and the system containing both the bodies is the articulated body. The basic properties of the articulated body are summarized below.

1. The kinematic tree is a *floating kinematic tree*, i.e., articulated body has no connection to the fixed base.
2. Every articulated body has exactly one handle.
3. Every handle has full six degrees of freedom.
4. Every assembly operation consists of connecting one handle to another via a single joint.

8.2.1 Calculation of articulated body inertia

The calculation of articulated body inertia is taken from [1]. Consider two articulated bodies B_1 and B_2 in Fig. 8.1. The applied forces acting on bodies B_1 and B_2 are \mathbf{f}_1 and \mathbf{f}_2 respectively. The resulting accelerations are denoted by \mathbf{a}_1 and \mathbf{a}_2 . The equations of motion are given by

$$\mathbf{f}_1 = \mathbf{I}_1^A \mathbf{a}_1 + \mathbf{p}_1^A, \quad (8.5)$$

$$\mathbf{f}_2 = \mathbf{I}_2^A \mathbf{a}_2 + \mathbf{p}_2^A, \quad (8.6)$$

where \mathbf{I}_1^A and \mathbf{I}_2^A are the articulated-body inertia of bodies B_1 and B_2 respectively. The bias forces are given by \mathbf{p}_1^A and \mathbf{p}_2^A for bodies B_1 and B_2 respectively. When the joint screw coordinate vector \mathbf{X} (in body-fixed representation) connects the bodies, a new unknown force, \mathbf{f}_J arises, transmitting the force from one body to another. The equations describing the relationship between the forces are given by

$$\mathbf{f}_1 = \mathbf{f} - \mathbf{f}_J \quad \text{and} \quad \mathbf{f}_2 = \mathbf{f}_J. \quad (8.7)$$

The unknown force imposes constraints on the system,

$$\mathbf{a}_2 - \mathbf{a}_1 = \mathbf{X}\ddot{\mathbf{q}} + \mathbf{c} \quad \text{and} \quad \boldsymbol{\tau} = \mathbf{X}^T \mathbf{f}_J, \quad (8.8)$$

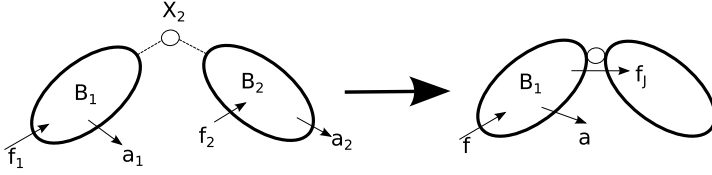


Figure 8.1 Construction of new articulated body, adapted from [1].

where $\mathbf{c} = \dot{\mathbf{X}}\dot{\mathbf{q}}$.

Substituting the values,

$$\begin{aligned}\boldsymbol{\tau} &= \mathbf{X}^T \mathbf{f}_2 \\ &= \mathbf{X}^T (\mathbf{I}_2^A \mathbf{a}_2 + \mathbf{p}_2^A) \\ &= \mathbf{X}^T (\mathbf{I}_2^A (\mathbf{a}_1 + \mathbf{X}\ddot{\mathbf{q}} + \mathbf{c}) + \mathbf{p}_2^A).\end{aligned}$$

So,

$$\ddot{\mathbf{q}} = (\mathbf{X}^T \mathbf{I}_2^A \mathbf{X})^{-1} (\boldsymbol{\tau} - \mathbf{X}^T (\mathbf{I}_2^A (\mathbf{a}_1 + \mathbf{c}) + \mathbf{p}_2^A)). \quad (8.9)$$

Solving for $\ddot{\mathbf{q}}$, the equation of motion relating \mathbf{f} and \mathbf{a}_1 ,

$$\begin{aligned}\mathbf{f} &= \mathbf{f}_1 + \mathbf{f}_2 \\ &= \mathbf{I}_1^A \mathbf{a}_1 + \mathbf{I}_2^A \mathbf{a}_2 + \mathbf{p}_1^A + \mathbf{p}_2^A \\ &= \mathbf{I}_1^A \mathbf{a}_1 + \mathbf{I}_2 (\mathbf{a}_1 + \mathbf{X}\ddot{\mathbf{q}} + \mathbf{c}) + \mathbf{p}_1^A + \mathbf{p}_2^A \\ &= \mathbf{I}_1^A \mathbf{a}_1 + \mathbf{I}_2 (\mathbf{a}_1 + \mathbf{c} + \mathbf{X} (\mathbf{X}^T \mathbf{I}_2^A \mathbf{X})^{-1} (\boldsymbol{\tau} - \mathbf{X}^T (\mathbf{I}_2^A (\mathbf{a}_1 + \mathbf{c}) + \mathbf{p}_2^A))) + \mathbf{p}_1^A + \mathbf{p}_2^A\end{aligned}$$

Comparing it with equation $\mathbf{f} = \mathbf{I}^A \mathbf{a}_1 + \mathbf{p}^A$,

$$\mathbf{I}^A = \mathbf{I}_1^A + \mathbf{I}_2^A - \mathbf{I}_2^A \mathbf{X} (\mathbf{X}^T \mathbf{I}_2^A \mathbf{X})^{-1} \mathbf{X}^T \mathbf{I}_2^A$$

and

$$\mathbf{p}^A = \mathbf{p}_1^A + \mathbf{p}_2^A + \mathbf{I}_2^A \mathbf{c} + \mathbf{I}_2^A \mathbf{X} (\mathbf{X}^T \mathbf{I}_2^A \mathbf{X})^{-1} (\boldsymbol{\tau} - \mathbf{X}^T (\mathbf{I}_2^A \mathbf{c} + \mathbf{p}_2^A)), \quad (8.10)$$

where \mathbf{I}^A and \mathbf{p}^A are the computed articulated body inertia and spatial bias force when B_2 is attached to the *handle* B_1 via a connecting joint.

8.3 Articulated body algorithm

The ABA algorithm calculates forward dynamics by computing the articulated body inertia in the previous section. The algorithm makes three

passes over the tree-type system. In Algorithm 8.1 from [1], the passes are as follows

1. The first pass goes from base to tip of the kinematic tree and is responsible for calculating the velocity and bias terms of each node in the kinematic tree.
2. The second pass runs from tip to base and calculates the articulated body inertia and bias forces.
3. The third pass goes from base to tip and computes the accelerations.

8.4 Recursive forward dynamics using constraint embedding

The constraint embedding formulation for ABA proposed by Jain [3] is reformulated in this part, and it produces a recursive forward dynamics method in minimal coordinates for series parallel hybrid robots [11]. It translates SOA's ideas into conventional Lie group concepts using Featherstone's [1] root-to-tip regular numbering scheme, which is more widely utilized in the robotics community.

8.4.1 Strategy for constraint embedding

The parallel kinematic chain can be considered as a node in the spanning tree [3]. By capturing the submechanism in a new node, this changes the spanning tree into a new spanning tree, as shown in Fig. 8.2. In the figure, \mathcal{G} denotes the new submechanism node, p denotes the parent of \mathcal{G} , and c denotes the child of \mathcal{G} . The explicit loop closure constraints shall be captured in \mathcal{G} .

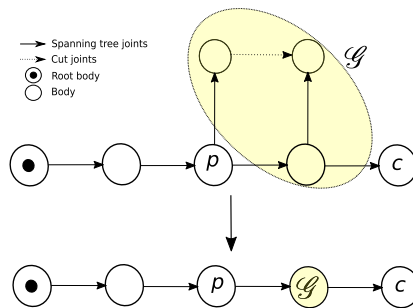


Figure 8.2 Introduction of a submechanism node in the tree, adapted from [3].

Algorithm 8.1 Articulated body algorithm.

```

1:  $\mathbf{v}_0 = \mathbf{0}$ 
2: for  $i = 1$  to  $N_B$  do
3:    $[\mathbf{Ad}_J, \mathbf{X}_i, \mathbf{v}_J, \mathbf{c}_J] = jcalc(jtype(i), \mathbf{q}_i, \dot{\mathbf{q}}_i)$ 
4:    $\mathbf{Ad}_{T_{\lambda(i),i}} = \mathbf{Ad}_J \mathbf{Ad}_T(i)$ 
5:   if  $\lambda(i) \neq 0$  then
6:      $\mathbf{Ad}_{T_{0,i}} = \mathbf{Ad}_{T_{0,\lambda(i)}} \mathbf{Ad}_{T_{\lambda(i),i}}$ 
7:   end if
8:    $\mathbf{v}_i = \mathbf{Ad}_{T_{\lambda(i),i}} \mathbf{v}_{\lambda(i)} + \mathbf{v}_J$ 
9:    $\mathbf{c}_i = \mathbf{c}_J + \mathbf{v}_i \times \mathbf{v}_J$ 
10:   $\mathbf{I}_i^A = \mathbf{I}_i$ 
11:   $\mathbf{p}_i^A = \mathbf{ad}_{\mathbf{v}_i} \mathbf{I}_i \mathbf{v}_i - \mathbf{Ad}_{T_{i,0}}^T \mathbf{f}_i^x$ 
12: end for
13: for  $i = N_B$  to 1 do
14:   $\mathbf{U}_i = \mathbf{I}_i^A \mathbf{X}_i$ 
15:   $\mathbf{D}_i = \mathbf{X}_i^T \mathbf{U}_i$ 
16:   $\mathbf{u}_i = \boldsymbol{\tau}_i - \mathbf{X}_i^T \mathbf{p}_i$ 
17:  if  $\lambda(i) \neq 0$  then
18:     $\mathbf{I}^a = \mathbf{I}_i^A - \mathbf{U}_i \mathbf{D}_i^{-1} \mathbf{U}_i^T$ 
19:     $\mathbf{p}^a = \mathbf{p}_i^A + \mathbf{I}^a \mathbf{c}_i + \mathbf{U}_i \mathbf{D}_i^{-1} \mathbf{u}_i$ 
20:     $\mathbf{I}_{\lambda(i)}^A = \mathbf{I}_{\lambda(i)}^A + \mathbf{Ad}_{T_{\lambda(i),i}}^T \mathbf{I}^a \mathbf{Ad}_{T_{\lambda(i),i}}$ 
21:     $\mathbf{p}_{\lambda(i)}^A = \mathbf{p}_{\lambda(i)}^A + \mathbf{Ad}_{T_{\lambda(i),i}}^T \mathbf{p}^a$ 
22:  end if
23: end for
24:  $\mathbf{a}_0 = -\mathbf{a}_g$ 
25: for  $i = 1$  to  $N_B$  do
26:   $\mathbf{a}' = \mathbf{Ad}_{T_{\lambda(i),i}} \mathbf{a}_{\lambda(i)} + \mathbf{c}_i$ 
27:   $\ddot{\mathbf{q}}_i = \mathbf{D}_i^{-1} (\mathbf{u}_i - \mathbf{U}_i^T \mathbf{a}')$ 
28:   $\mathbf{a}_i = \mathbf{a}' + \mathbf{X}_i \ddot{\mathbf{q}}_i$ 
29: end for

```

The loop closure constraints shall be embedded in the spanning tree when going from $p \rightarrow \mathcal{G} \rightarrow c$. To compute the variables in the algorithm, screw theory and Lie group concepts are used that are well discussed in literature [1,12–14].

The explicit Jacobian matrix, $\mathbf{G}_{\mathcal{G}}$, and explicit bias acceleration vector, $\mathbf{g}_{\mathcal{G}}$, can be computed for a parallel submechanism in the tree, either numerically or analytically. The new submechanism node \mathcal{G} contains par-

allel kinematic chain bodies from the kinematic tree. The reduced Jacobian matrix $\mathbf{J}_{\mathcal{G}}$ is given by

$$\mathbf{J}_{\mathcal{G}} = \mathbf{A}_{\mathcal{G}} \mathbf{X}_{\mathcal{G}} \mathbf{G}_{\mathcal{G}}, \quad (8.11)$$

where

$$\mathbf{A}_{\mathcal{G}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{Ad}_{T_{2,1}} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{Ad}_{T_{3,1}} & \mathbf{Ad}_{T_{3,2}} & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Ad}_{T_{n,1}} & \mathbf{Ad}_{T_{n,2}} & \dots & \mathbf{Ad}_{T_{n,n-1}} & \mathbf{I} \end{bmatrix} \quad (8.12)$$

with \mathbf{Ad}_{T_i} is the adjoint transformation matrix of size (6×6) defined as

$$\mathbf{Ad}_{T_{i,j}} = \begin{bmatrix} \mathbf{R}_{i,j} & \mathbf{0} \\ {}^i\tilde{\mathbf{r}}_{i,j} \mathbf{R}_{i,j} & \mathbf{R}_{i,j} \end{bmatrix}, \quad (8.13)$$

where $\mathbf{R}_{i,j} \in SO(3)$ is the rotation matrix and ${}^i\tilde{\mathbf{r}}_{i,j}$ is the skew symmetric matrix of position vector in frame i .

$$\mathbf{X}_{\mathcal{G}} = \text{diag}({}^1\mathbf{X}_1, {}^2\mathbf{X}_2, \dots, {}^n\mathbf{X}_n), \quad (8.14)$$

where ${}^i\mathbf{X}_i$ is the screw coordinates of the vector, of joint frame i represented in body-fixed frame i .

For the new spanning tree graph, articulated body inertia and bias force for the submechanism node \mathcal{G} need to be computed. The connecting matrices acts as bridge in the ABA algorithm for computing these quantities:

1. Connecting matrix from $p \rightarrow \mathcal{G}$ called as $\mathbf{A}(p, \mathcal{G})$, which maps the nodes in \mathcal{G} to p .
2. Connecting matrix from $\mathcal{G} \rightarrow c$ called as $\mathbf{A}(\mathcal{G}, c)$, which maps the nodes in \mathcal{G} to c .

The connecting matrices are depicted in Fig. 8.3. The connecting matrix $\mathbf{A}(p, \mathcal{G})$ contains the information about joints (blue) connecting node p to \mathcal{G} . Similarly, $\mathbf{A}(\mathcal{G}, c)$ contain the information about the joints (orange) connecting node \mathcal{G} to c .

8.4.2 ABA for forward dynamics of closed loops

As ABA is a well-known recursive algorithm, the main algorithm is skipped here and can be followed in [1,3]. This section only provides the adapted quantities to consider loop closure constraints. Considering n bodies and m

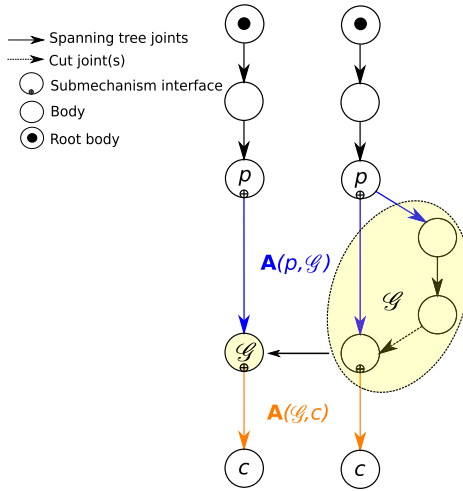


Figure 8.3 Connecting matrices for new submechanism node [11].

independent coordinates in submechanism node \mathcal{G} , $\mathbf{A}(p, \mathcal{G}) \in \mathbb{R}^{6n \times 6}$, and $\mathbf{A}(\mathcal{G}, c) \in \mathbb{R}^{6 \times 6n}$. The recursive formulation when encountering a closed loop in the system can be formulated as

8.4.2.1 First pass

The first pass in the algorithm remains the same and computes velocities and bias terms.

8.4.2.2 Second pass

The second loop is modified when moving from tip to base in the spanning tree. The modifications are listed below.

From child node c to submechanism node \mathcal{G} ($c \rightarrow \mathcal{G}$)

$$\mathbf{I}_{\mathcal{G}}^A = \mathbf{I}_{\mathcal{G}} + \sum_{\forall c \in \mu(\mathcal{G})} \mathbf{A}^T(\mathcal{G}, c) \mathbf{I}_c^A \mathbf{A}(\mathcal{G}, c)$$

$$\mathbf{p}_{\mathcal{G}}^A = \mathbf{p}_{\mathcal{G}} + \sum_{\forall c \in \mu(\mathcal{G})} \mathbf{A}^T(\mathcal{G}, c) \mathbf{p}_c^A$$

$$\mathbf{U}_{\mathcal{G}} = \mathbf{I}_{\mathcal{G}}^A \mathbf{J}_{\mathcal{G}}$$

$$\mathbf{D}_{\mathcal{G}} = \mathbf{J}_{\mathcal{G}}^T \mathbf{U}_{\mathcal{G}}$$

$$\mathbf{u}_{\mathcal{G}} = \boldsymbol{\tau}_{\mathcal{G}Y} - \mathbf{J}_{\mathcal{G}}^T \mathbf{p}_{\mathcal{G}}^A$$

$$\mathbf{c}'_{\mathcal{G}} = \mathbf{A}_{\mathcal{G}} \mathbf{c}_{\mathcal{G}} + \mathbf{A}_{\mathcal{G}} \mathbf{X}_{\mathcal{G}} \mathbf{g}_{\mathcal{G}}$$

$$\begin{aligned}\mathbf{I}_{\mathcal{G}}^a &= \mathbf{I}_{\mathcal{G}} - \mathbf{U}_{\mathcal{G}} \mathbf{D}_{\mathcal{G}}^{-1} \mathbf{U}_{\mathcal{G}}^T \\ \mathbf{p}_{\mathcal{G}}^a &= \mathbf{p}_{\mathcal{G}}^A + \mathbf{I}_{\mathcal{G}}^a \mathbf{c}'_{\mathcal{G}} + \mathbf{U}_{\mathcal{G}} \mathbf{D}_{\mathcal{G}}^{-1} \mathbf{u}_{\mathcal{G}}\end{aligned}$$

From submechanism node \mathcal{G} to parent node p ($\mathcal{G} \rightarrow p$)

$$\begin{aligned}\mathbf{I}_p^A &= \mathbf{A}^T(p, \mathcal{G}) \mathbf{I}_{\mathcal{G}}^a \mathbf{A}(p, \mathcal{G}) + \mathbf{I}_p \\ \mathbf{p}_p^A &= \mathbf{A}^T(p, \mathcal{G}) \mathbf{p}_{\mathcal{G}}^a + \mathbf{p}_p,\end{aligned}$$

where

- $\mathbf{I}_{\mathcal{G}} \in \mathbb{R}^{6n \times 6n}$ contain diagonal terms as spatial mass-inertia matrix of each body in the node \mathcal{G} .
- $\mathbf{I}_{\mathcal{G}}^A \in \mathbb{R}^{6n \times 6n}$ is the articulated body inertia of node \mathcal{G} .
- $\mathbf{p}_{\mathcal{G}} \in \mathbb{R}^{6n}$ is the articulated bias forces of the node \mathcal{G} .
- $\mathbf{J}_{\mathcal{G}} \in \mathbb{R}^{6n \times m}$ is the reduced Jacobian matrix in explicit form for node \mathcal{G} , as in Eq. (8.11).
- $\boldsymbol{\tau}_{\mathcal{G}\gamma} \in \mathbb{R}^m$ are the generalized forces of the node \mathcal{G} in independent coordinates.

Note the extra step that is introduced for the velocity product of the submechanism node, $\mathbf{c}'_{\mathcal{G}} \in \mathbb{R}^{6n}$ by combining it with the explicit constraints.

8.4.2.3 Third pass

The next modifications in third loop are given below when moving from base to tip.

From parent node p to submechanism node \mathcal{G} ($p \rightarrow \mathcal{G}$)

$$\begin{aligned}\mathbf{a}'_{\mathcal{G}} &= \mathbf{A}(p, \mathcal{G}) \mathbf{a}_p + \mathbf{c}'_{\mathcal{G}} \\ \ddot{\mathbf{y}}_{\mathcal{G}} &= \mathbf{D}_{\mathcal{G}}^{-1} (\mathbf{u}_{\mathcal{G}} - \mathbf{U}_{\mathcal{G}}^T \mathbf{a}'_{\mathcal{G}}) \\ \ddot{\mathbf{q}}_{\mathcal{G}} &= \mathbf{G}_{\mathcal{G}} \ddot{\mathbf{y}}_{\mathcal{G}} + \mathbf{g}_{\mathcal{G}} \\ \mathbf{a}_{\mathcal{G}} &= \mathbf{a}'_{\mathcal{G}} + \mathbf{J}_{\mathcal{G}} \ddot{\mathbf{q}}_{\mathcal{G}}\end{aligned}$$

From submechanism node \mathcal{G} to child node c ($\mathcal{G} \rightarrow c$)

$$\begin{aligned}\mathbf{a}'_c &= \mathbf{A}(\mathcal{G}, c) \mathbf{a}_{\mathcal{G}} + \mathbf{c}_c \\ \ddot{\mathbf{q}}_c &= \mathbf{D}_c^{-1} (\mathbf{u}_c - \mathbf{U}_c^T \mathbf{a}'_c),\end{aligned}$$

where

- $\mathbf{a}_{\mathcal{G}} \in \mathbb{R}^{6n}$ is the acceleration of submechanism node \mathcal{G} .
- $\ddot{\mathbf{q}}_{\mathcal{G}} \in \mathbb{R}^n$ is the spanning tree joints acceleration in the submechanism node \mathcal{G} .

In this loop, also note the extra steps that map the explicit constraints to obtain the accelerations of all bodies in the submechanism node.

8.4.3 Mass matrix factorization and inversion

The closed form of forward dynamics uses the mass matrix factorization and inversion, inspired from [3,15–17]. Considering a system with $c \rightarrow \mathcal{G} \rightarrow p$, the properties can be defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_c & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_G \mathbf{A}^T(\mathcal{G}, c) \mathbf{A}_c & \mathbf{A}_G & \mathbf{0} \\ \mathbf{A}_p \mathbf{A}^T(p, \mathcal{G}) \mathbf{A}^T(\mathcal{G}, c) \mathbf{A}_c & \mathbf{A}_p \mathbf{A}^T(p, \mathcal{G}) & \mathbf{A}_p \end{bmatrix} \quad (8.15)$$

Similarly, \mathbf{X} and \mathbf{M} are defined as

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_c & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_G \mathbf{G}_G & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_p \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} \mathbf{I}_c & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_G & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_p \end{bmatrix}. \quad (8.16)$$

The components used in recursive formulations can be written as

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_c & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_G & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_p \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \mathbf{D}_c & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_G & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_p \end{bmatrix}. \quad (8.17)$$

New system quantities, ψ and κ are defined using Eqs. (8.15), (8.16), and (8.17), adapted from [3].

$$\psi = [\mathbf{I} - \mathbf{X}(\mathbf{U}\mathbf{D}^{-1})^T] \mathbf{A} \quad (8.18)$$

$$\kappa = (\mathbf{U}\mathbf{D}^{-1})^T \mathbf{A}_D, \quad (8.19)$$

where \mathbf{A}_D is nilpotent satisfying von-Neuman series $\mathbf{A}_D = (\mathbf{I} - \mathbf{A}^{-1})$. They satisfy the property,

$$\psi^{-1} \mathbf{A} = \mathbf{I} + \mathbf{X} \kappa \mathbf{A}. \quad (8.20)$$

The mass-inertia matrix can be written as

$$\mathbf{M}_\gamma = \mathbf{J}^T \mathbf{M} \mathbf{J}, \quad (8.21)$$

where $\mathbf{J} = \mathbf{AX}$, and \mathbf{M} is defined in Eq. (8.16). Introducing $(\psi\psi^{-1})$ in above equation,

$$\begin{aligned} \mathbf{M}_y &= (\mathbf{AX})^T (\psi\psi^{-1})^T \mathbf{M} \psi \psi^{-1} \mathbf{AX} \\ \mathbf{M}_y &= \mathbf{X}^T (\psi^{-1} \mathbf{A})^T \psi^T \mathbf{M} \psi (\psi^{-1} \mathbf{A}) \mathbf{X} \end{aligned} \quad (8.22)$$

Using Eq. (8.20),

$$\begin{aligned} \mathbf{M}_y &= \mathbf{X}^T [\mathbf{I} + \mathbf{X} \kappa \mathbf{A}]^T \psi^T \mathbf{M} \psi [\mathbf{I} + \mathbf{X} \kappa \mathbf{A}] \mathbf{X} \\ \mathbf{M}_y &= [\mathbf{I} + \kappa \mathbf{AX}]^T \mathbf{X}^T \psi^T \mathbf{M} \psi \mathbf{X} [\mathbf{I} + \kappa \mathbf{AX}] \\ \mathbf{M}_y &= [\mathbf{I} + \kappa \mathbf{AX}]^T \mathbf{D} [\mathbf{I} + \kappa \mathbf{AX}], \end{aligned} \quad (8.23)$$

where $\mathbf{D} = \mathbf{X}^T \psi^T \mathbf{M} \psi \mathbf{X}$. Therefore, inverting Eq. (8.23)

$$\mathbf{M}_y^{-1} = [\mathbf{I} + \kappa \mathbf{AX}]^{-1} \mathbf{D}^{-1} [\mathbf{I} + \kappa \mathbf{AX}]^{-T} \quad (8.24)$$

From standard matrix identity $[\mathbf{I} + \mathbf{AB}]^{-1} = \mathbf{I} - \mathbf{A}[\mathbf{I} + \mathbf{BA}]^{-1} \mathbf{B}$,

$$\begin{aligned} [\mathbf{I} + \kappa \mathbf{AX}]^{-1} &= \mathbf{I} - \kappa \mathbf{A} [\mathbf{I} + \mathbf{X} \kappa \mathbf{A}]^{-1} \mathbf{X} \\ [\mathbf{I} + \kappa \mathbf{AX}]^{-1} &= \mathbf{I} - \kappa \mathbf{A} (\psi^{-1} \mathbf{A})^{-1} \mathbf{X} \\ [\mathbf{I} + \kappa \mathbf{AX}]^{-1} &= \mathbf{I} - \kappa \psi \mathbf{X}. \end{aligned} \quad (8.25)$$

Therefore, using Eq. (8.25) and substituting in Eq. (8.24), mass-matrix inversion can be written using factorization as

$$\mathbf{M}_y^{-1} = [\mathbf{I} - \kappa \psi \mathbf{X}] \mathbf{D}^{-1} [\mathbf{I} - \kappa \psi \mathbf{X}]^T. \quad (8.26)$$

The mass-inertia matrix factorization and inversion is captured in Eq. (8.26).

8.5 Example

A simplistic example of a series-parallel hybrid chain with three submechanisms can be used to illustrate the demonstration. The first submechanism is a serial chain with three revolute joints (a 3R serial chain), the second is a parallel kinematic chain with one loop (a RRPR mechanism), and the third is a serial chain with one revolute joint. In Fig. 8.4, the topological graph of the series-parallel hybrid case is depicted.

From Fig. 8.4, the submechanism node can be created for the parallel kinematic chain of lambda mechanism. The submechanism node consists

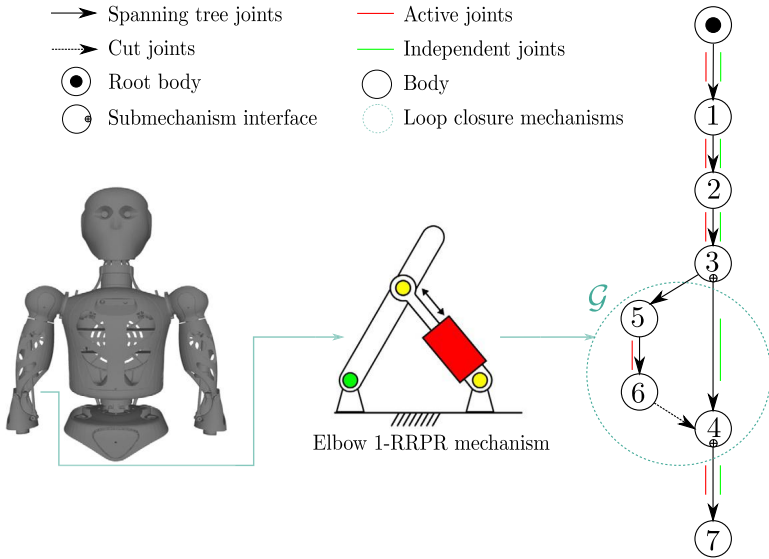


Figure 8.4 Topological graph of 3R-Lambda-R chain.

of three bodies as $\mathcal{G} = \{4, 5, 6\}$. For this submechanism node \mathcal{G} , parent node is $p = 3$, and child node is $c = 7$.

The components can be computed for node \mathcal{G} as,

$$I_{\mathcal{G}} = \begin{bmatrix} I_4 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_5 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_6 \end{bmatrix}, \tag{8.27}$$

where I_i is the spatial inertia matrix of body i .

$$\mathbf{p}_{\mathcal{G}} = \begin{bmatrix} \mathbf{p}_4 \\ \mathbf{p}_5 \\ \mathbf{p}_6 \end{bmatrix}, \tag{8.28}$$

where \mathbf{p}_i is the spatial bias force of body i .

$$A_{\mathcal{G}} = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{Ad}_{T_{5,6}} & I \end{bmatrix}, \tag{8.29}$$

where \mathbf{I} is the Identity matrix.

$$\mathbf{X}_{\mathcal{G}} = \begin{bmatrix} \mathbf{X}_4 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_5 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_6 \end{bmatrix}, \quad (8.30)$$

where \mathbf{X}_i is the screw coordinate vector of joint i in body-fixed representation.

Connecting matrix are very important for mapping of components from submechanism node to either parent node or child node. For this example,

$$\mathbf{A}(p, \mathcal{G}) = \begin{bmatrix} \mathbf{Ad}_{T_{3,4}} & \mathbf{Ad}_{T_{3,5}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Ad}_{T_{5,6}} & \mathbf{I} \end{bmatrix} \quad (8.31)$$

$$\mathbf{A}(\mathcal{G}, c) = \begin{bmatrix} \mathbf{Ad}_{T_{4,7}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \quad (8.32)$$

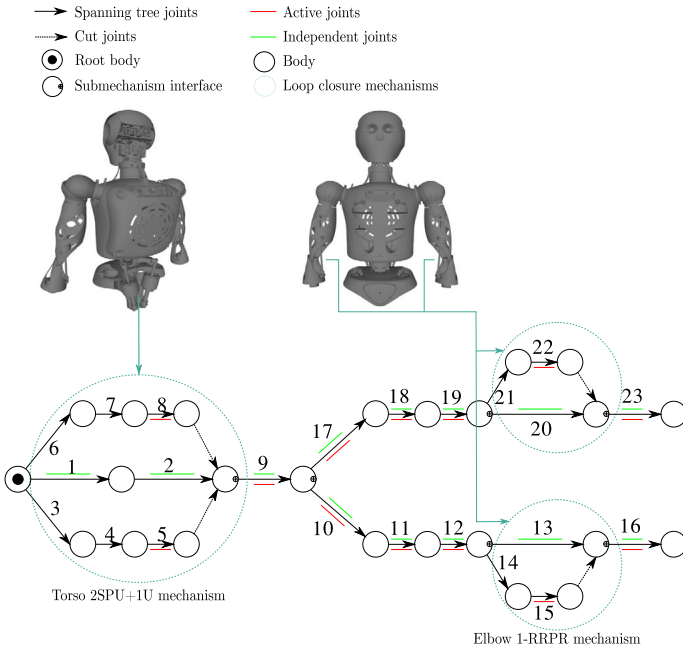
Substituting these matrices, the constraint embedding can be performed for the closed loop system in Fig. 8.4.

8.6 Validation of the constraint embedding approach

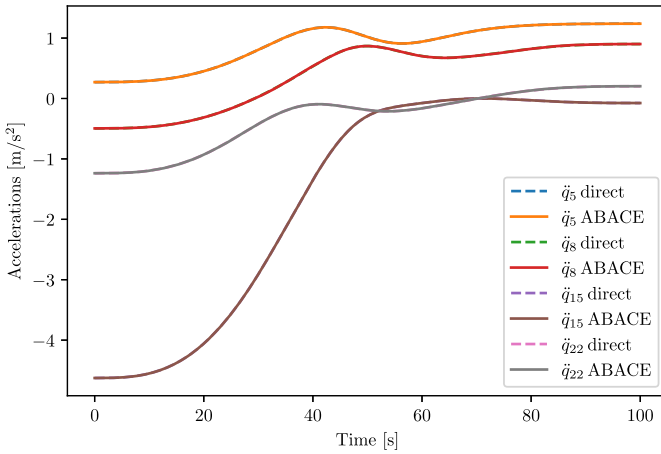
The adapted algorithm for recursive forward dynamics with Lie group formulations is validated on the reduced version of RH5 Manus robot [18].

The validation is done using the direct inversion of joint-space mass inertia matrix to compute the required joint accelerations of the reduced version of RH5 Manus robot [11] in Fig. 8.5(a). The input joint positions and velocities are provided as cycloidal trajectories for the independent joints (green in Fig. 8.5(a)) and the accelerations are computed using both the approaches, i.e., direct inversion of the mass inertia matrix and adapted constraint embedding approach.

Fig. 8.5(b) presents the active joint accelerations (red in Fig. 8.5(a)). The legend *direct* computes the acceleration using direct inversion of mass inertia matrix, and legend *ABACE* computes the acceleration using the constraint embedding approach in ABA. The root mean squared error for all the joints is computed to be zero, thus validating the constraint embedding reformulation.



(a) Reduced version of RH5 Manus robot



(b) Acceleration plots from forward dynamics

Figure 8.5 Validation on a series-parallel hybrid robot example [11].

8.7 Conclusion

By translating SOA concepts into standard Lie group concepts and adapting the recursive algorithm for the base to tip numbering scheme, the chapter reformulates Jain's constraint embedding formulation for ABA. The adapted algorithm is validated with the convention method of direct inversion of mass inertia matrix for a complex series-parallel hybrid robot. By making this method more approachable for developers who are familiar with the base-to-tip numbering scheme and Featherstone's notation, this reformulation can be advantageous to the robotics community.

References

- [1] R. Featherstone, *Rigid Body Dynamics Algorithms*, Springer, 2008, <https://doi.org/10.1007/978-1-4899-7560-7>.
- [2] R. Featherstone, D. Orin, Robot dynamics: equations and algorithms, in: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1, 2000, pp. 826–834, <https://doi.org/10.1109/ROBOT.2000.844153>.
- [3] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*, Springer US, Boston, MA, 2011.
- [4] G. Rodriguez, A. Jain, K. Kreutz-Delgado, A spatial operator algebra for manipulator modeling and control, *The International Journal of Robotics Research* 10 (4) (1991) 371–381, <https://doi.org/10.1177/027836499101000406>.
- [5] M. Felis, RBDL: an efficient rigid-body dynamics library using recursive algorithms, *Autonomous Robots* 41 (2017) 495–511, <https://doi.org/10.1007/s10514-016-9574-0>.
- [6] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiroux, O. Stasse, N. Mansard, The Pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives, in: *IEEE International Symposium on System Integrations (SI)*, 2019.
- [7] E. Todorov, T. Erez, Y. Tassa, MuJoCo: a physics engine for model-based control, in: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.
- [8] R. Tedrake, the Drake Development Team, *Drake: model-based design and verification for robotics*, <https://drake.mit.edu>, 2019.
- [9] E. Coumans, Y. Bai, Pybullet, a python module for physics simulation for games, robotics and machine learning, <http://pybullet.org>, 2016–2021.
- [10] J. Lee, M.X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S.S. Srinivasa, M. Stilman, C.K. Liu, Dart: dynamic animation and robotics toolkit, *Journal of Open Source Software* 3 (22) (2018) 500, <https://doi.org/10.21105/joss.00500>.
- [11] R. Kumar, S. Kumar, A. Müller, F. Kirchner, Modular and hybrid numerical-analytical approach – a case study on improving computational efficiency for series-parallel hybrid robots, in: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 3476–3483, <https://doi.org/10.1109/IROS47612.2022.9981474>.
- [12] K.M. Lynch, F.C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st edition, Cambridge University Press, USA, 2017.

- [13] A. Mueller, Screw and Lie group theory in multibody kinematics, *Multibody System Dynamics* 43 (2018) 37–70, <https://doi.org/10.1007/s11044-017-9582-7>.
- [14] A. Mueller, S. Kumar, Closed-form time derivatives of the equations of motion of rigid body systems, *Multibody System Dynamics* 53 (2021) 257–273, <https://doi.org/10.1007/s11044-021-09796-8>.
- [15] G. Rodriguez, Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics, *IEEE Journal on Robotics and Automation* 3 (6) (1987) 624–639, <https://doi.org/10.1109/JRA.1987.1087147>.
- [16] G. Rodriguez, K. Kreutz-Delgado, Spatial operator factorization and inversion of the manipulator mass matrix, *IEEE Transactions on Robotics and Automation* 8 (1) (1992) 65–76, <https://doi.org/10.1109/70.127240>.
- [17] K. Kreutz-Delgado, A. Jain, G. Rodriguez, Recursive formulation of operational space control, in: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1750–1753, <https://doi.org/10.1109/ROBOT.1991.131874>.
- [18] M. Boukheddimi, S. Kumar, H. Peters, D. Mrona, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8540–8546, <https://doi.org/10.1109/ICRA46639.2022.9811843>.

CHAPTER 9

Whole-body control

Dennis Mronga^a, Shivesh Kumar^a, and Frank Kirchner^{a,b}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bWorking Group Robotics, University of Bremen, Bremen, Germany

9.1 Motivation

In recent years, whole-body control (WBC) [1] has emerged as the standard approach for controlling complex robotic systems with redundant degrees of freedom. The core idea of WBC is to specify robot tasks within the cost function or as constraints of an instantaneous optimization problem, typically a quadratic program, and design a set of feedback controllers around it. In each control cycle, the tasks/constraints are updated, the QP is solved, and the solution is applied to the entire robot actuators (whole-body). This way, complex robot tasks can be composed from simple descriptors (controllers/cost functions), physical limitations can be integrated as constraints, and the redundancy of a robot can be nicely exploited. Considering, e.g., humanoid robots, WBC is required to control the robot's center of mass to maintain balance and integrate it with other primary and secondary tasks like grasping or postural control, while considering the physical limitations of the robot actuators. Fig. 9.1 illustrates the general idea of WBC.

The term *whole-body control* has been established by Luis Sentis in his seminal work on humanoid robot control [2]. In contrast to his approach, which uses a sequence of projections and pseudo-inversions in closed form, modern WBC approaches are typically based on numerical optimization [3–6].

Many software implementations for WBC exist today, Table 9.1 gives an overview on the most popular frameworks. Like most other software for robot motion planning and control, the existing WBC frameworks are designed for serial or tree-type robots. Parallel or series-parallel hybrid mechanisms are not supported. This is for two main reasons: On the one hand, series-parallel hybrid robots are more difficult to model and control than serial or tree-type systems. On the other hand, as also illustrated in Table 9.1, most WBC software frameworks use the Unified Robot Description Format (URDF) to model robot kinematics and dynamics. URDF is

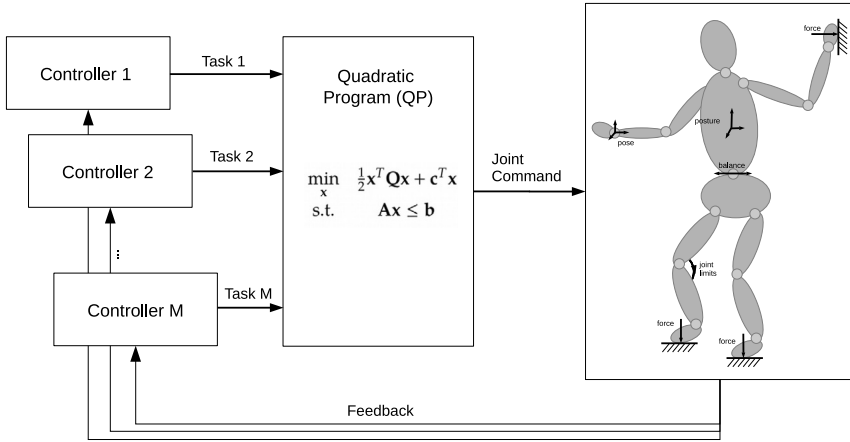


Figure 9.1 General idea of Whole-Body Control.

Table 9.1 Overview of the most popular open-source software frameworks for WBC.

Name	Robot Model	Parser	License	Ref.
Optimization-based framework for Robotic Control Applications (ORCA)	KDL/iDynTree	URDF	CeCILL-C	[8]
Instantaneous Task Specification using Constraints (ITaSC)	KDL	URDF	LGPLv2.1 / BSD	[9]
IHMC Whole-Body Controller	internal	URDF/SDF	Apache 2.0 / GPLv3	[4]
ControlIt!	RBDL	URDF	LGPL	[10]
Task Space Inverse Dynamics (TSID)	Pinochio	URDF	BSD 2	[7]

a well-established tool in the robotics community. However, it does not allow the definition of closed loops in the kinematic model of a robot. As a workaround, the existing WBC implementations use serial/tree-type robot models, which abstract the parallel mechanisms as independent joints. That is, the parallel mechanisms are hidden behind one or multiple rotational or prismatic joints, arranged in series. The forward/inverse kinematics and dynamics of the parallel submechanisms are resolved in a specialized function

after the whole-body controller. This procedure has multiple theoretical and practical disadvantages, which are listed in the following:

1. The physical limits of the actuators within the parallel mechanisms cannot be modeled correctly in existing WBC approaches. As a result, the feasible workspace is over- or underestimated by the whole-body controller, a fact that will be elaborated in detail later.
2. The solution of the WBC problem will be inaccurate as it does not correctly model the dynamic properties of parallel submechanisms, e.g., the mass-inertia distribution. An extensive study on the effect of neglected dynamics in parallel submechanisms has been presented by Kumar et al. [11].
3. Singularities within the parallel submechanisms cannot be resolved/avoided by the whole-body controller.
4. The separate handling of parallel mechanisms leads to custom control software stacks, which are more complicated and more difficult to maintain.

The first point is of particular interest as proper modeling of the physical constraints is important in terms of optimal exploitation of the feasible robot workspace. In WBC, the physical limits are typically modeled as box constraints to the underlying optimization problem. For most parallel mechanisms, the feasible workspace itself is constrained, for example, the maximum position, velocity or torque of an actuator within a parallel mechanism depends on the current configuration. Thus, it is not possible to capture the entire workspace of a robot with closed loops by means of box constraints in independent coordinates.

As an example, consider the parallel ankle mechanism of the RH5 humanoid robot, which is described in [13]. The joint is a 2-dof orientational parallel mechanism of type 2SPRR+1U [14]. It comprises two linear actuators (ball screw drives), which are attached via two passive rotational joints to the foot link on the lower side and via a passive spherical joint to the shank link on the upper side. If both linear drives are moved in the same direction, the ankle performs a pure pitch movement. If they are moved in opposite directions with the same absolute velocity, the ankle mechanism performs a pure roll movement. It is in the nature of this mechanism that the actuation space and the independent joint space are not congruent if the latter is constrained by independent upper and lower joint limits. Thus, the admissible actuator positions of the mechanism cannot be captured by any choice of box constraints in independent coordinates without over- or underestimating the workspace. For the sake of safety and reliability, a con-

servative approximation of the workspace must be assumed as indicated by the green line. In doing so, a part of the admissible workspace is lost.

A similar problem can be observed when designing box constraints for parallel mechanisms on velocity or torque level. As an example, consider the elbow mechanism of the RH5 Manus robot [15]. The elbow has a 1-RRPR structure and is a variant of a slider-crank mechanism with linear actuator [12]. Both, the maximum velocity and torque of the independent elbow joint are position dependent. While the maximum elbow velocity can be obtained around the zero configuration when the arm is stretched, the maximum torque can be retrieved when the elbow is bent by -90 degrees. Obviously, it is not possible to design velocity- or torque-level box constraints in independent coordinates without over- or underestimating the admissible velocity or torque range. Again, for the sake of safety and reliability, a conservative approximation must be selected. Thus, a significant part of admissible velocity or torque range will be lost for most elbow configurations. As a result, the maximum velocity or torque of the elbow joint will be less than what the linear actuator can provide.

In order to overcome the limitations imposed by tree-type WBC approaches, a WBC framework for series-parallel hybrid robots is introduced in this chapter. The approach uses the Hybrid Robot Dynamics (HyRoDyn) library [16] to internally represent the robot model. HyRoDyn is a software workbench for computing the kinematics and dynamics of robots with parallel submechanisms and provides bidirectional mappings between the robot's full joint space, actuation space and independent joint space. Given the kinematic and dynamic quantities provided by HyRoDyn, the optimization problem inherent to WBC can be formulated in the actuation space of a series-parallel hybrid robot. This way, the physical limits of actuators within parallel submechanisms can be properly considered as box constraints. As a result, it is possible to fully exploit the admissible position, velocity and torque range of those actuators. In contrast to that, tree-type WBC approaches underestimate the robot workspace, as they model parallel submechanisms as independent joints and necessarily apply a conservative approximation of the admissible actuator range. The WBC framework introduced in this chapter is generally applicable to any robot with tree-type, parallel or series-parallel hybrid architecture. It provides a clean and theoretically sound solution for WBC of any of these robot types and renders the use of custom functions for computing the kinematics and dynamics of parallel submechanisms unnecessary. The following section describes the WBC framework and the related mathematical basics.

9.2 Whole-body control architecture

As described in earlier chapters, robots with closed loops can be described by three sets of coordinates:

- Spanning tree joints $\mathbf{q} \in \mathbb{R}^n$ describe the entire spanning tree of the robot.
- Independent joints $\mathbf{y} \in \mathbb{R}^m$ (also called generalized coordinates) describe a serial abstraction of the parallel submechanism.
- Actuated joints $\mathbf{u} \in \mathbb{R}^p$ describe all joints that contain an actuator.

Each of the three sets constitutes a different space describing the state of the robot, called (full) joint space, independent joint space, and actuation space, respectively.

Existing WBC approaches model the optimization problem in independent joint space, i.e., the optimization variables are the joint velocities, accelerations or torques in independent coordinates, respectively. The underlying robot model describes a serial or tree-type mechanism. As a simple example, consider a velocity-level WBC for a fixed-base robot and a single task:

$$\begin{aligned} \min_{\dot{\mathbf{y}}} \quad & \|\mathbf{J}\dot{\mathbf{y}} - \mathbf{v}_d\|_2 \\ \text{s.t.} \quad & \dot{\mathbf{y}}_m \leq \dot{\mathbf{y}} \leq \dot{\mathbf{y}}_M \end{aligned} \quad (9.1)$$

Here \mathbf{J} is the robot Jacobian, which maps task space velocities to independent velocity coordinates, $\dot{\mathbf{y}}$ the joint velocities, \mathbf{v}_d the desired spatial velocity, and $\dot{\mathbf{y}}_m, \dot{\mathbf{y}}_M$ the minimum/maximum joint velocities. As (9.1) considers a serial robot, the Jacobian of the spanning tree \mathbf{J} is equal to the Jacobian in independent coordinates \mathbf{J}_y . The solution to the optimization problem is given by the independent joint space velocities, which minimize $\|\mathbf{J}\dot{\mathbf{y}} - \mathbf{v}_d\|_2$. If this WBC approach is applied to a robot with parallel mechanisms, the solution requires a custom mapping to actuation space. Furthermore, the joint velocity limits $\dot{\mathbf{y}}_m, \dot{\mathbf{y}}_M$, modeled as box constraints here, only apply to independent joint space. As described earlier, they might not be able to cover the entire velocity range in actuation space.

In contrast, the WBC approach proposed in this chapter describes the optimization problem in actuation space of a series-parallel hybrid robot. It allows to directly consider actuator limits of parallel mechanisms as box constraints and avoids a custom conversion to actuation space after the whole-body controller.

9.2.1 Velocity-based WBC

On velocity-level, the proposed approach sets up and solves the following optimization problem:

$$\begin{aligned} \min_{\dot{\mathbf{x}}} \quad & \|\mathbf{J}_u \dot{\mathbf{x}} - \mathbf{v}_d\|_2 \\ \text{s.t.} \quad & \mathbf{J}_{u,c}^j \dot{\mathbf{x}} = 0, \quad \forall j \\ & \dot{\mathbf{u}}_m \leq \dot{\mathbf{u}} \leq \dot{\mathbf{u}}_M \end{aligned} \quad (9.2)$$

The optimization variables $\dot{\mathbf{x}} = (\dot{\mathbf{y}}_b \ \dot{\mathbf{u}}) \in \mathbb{R}^{6+p}$ comprise the actuator velocities $\dot{\mathbf{u}}$ of the robot and the 6-dof floating base velocities $\dot{\mathbf{y}}_b$. For the sake of readability, the optimization problem in (9.2) considers only a single task. The term \mathbf{v}_d denotes the desired spatial velocity of that task and $\mathbf{J}_u \in \mathbb{R}^{6 \times (6+p)}$ the corresponding task Jacobian, which describes the mapping from task space to actuation space of the robot. By separating the spanning-tree Jacobian into floating base and remaining robot joints $\mathbf{J} = (\mathbf{J}_b^{6 \times 6} \ \mathbf{J}_r^{6 \times p})$, the actuation space Jacobian can be computed as follows:

$$\mathbf{J}_u = (\mathbf{J}_b \mathbf{G} \quad \mathbf{J}_r \mathbf{G} \mathbf{G}_u^{-1}), \quad (9.3)$$

where $\mathbf{G} \in \mathbb{R}^{n \times m}$ is the loop closure Jacobian, which maps spanning tree to independent coordinates, and $\mathbf{G}_u \in \mathbb{R}^{p \times m}$ is the actuator Jacobian, which relates independent to actuated joints. The computation of the loop closure and actuator Jacobian is skipped here for brevity, but one can refer to standard textbooks [17, 18] regarding the derivation of loop closure constraints. The optimization problem is subject to a set of constraints, including actuator velocity limits $\dot{\mathbf{u}}_m, \dot{\mathbf{u}}_M$, as well as rigid contacts $\mathbf{J}_{u,c}^j \dot{\mathbf{x}} = 0, \forall j$, e.g., feet contact points that must not move. Here, $\mathbf{J}_{u,c}^j \in \mathbb{R}^{6 \times (6+p)}$ is the contact Jacobian of the j -th contact point.

When considering multiple robot tasks, the cost function changes as follows

$$\left\| \sum_i \mathbf{w}_i (\mathbf{J}_u^i \dot{\mathbf{x}} - \mathbf{v}_d^i) \right\|_2. \quad (9.4)$$

The task weights $\mathbf{w}^i \in \mathbb{R}^6$ control the priority of the task i with respect to other tasks within the cost function and may also prioritize particular task variables.

The Jacobians used in (9.2) are expressed in actuation space and computed using the HyRoDyn library. The solution of the optimization problem is the actuator velocities $\dot{\mathbf{u}}$, which minimize all tasks modeled in the cost function under the given constraints. Thus, the approach does not

require a custom mapping to actuation space outside of the whole-body controller. Also, it is able to consider the actuator velocity limits of all parallel submechanisms.

The WBC approach proposed here provides several types of controllers to model different robot tasks. For example, to model a Cartesian positioning task, one can use:

$$\mathbf{v}_d = \mathbf{v}_r + \mathbf{K}_p \begin{pmatrix} \mathbf{p}_r - \mathbf{p} \\ \theta \hat{\boldsymbol{\omega}}_r^a \end{pmatrix}, \quad (9.5)$$

where $\mathbf{v}_r \in \mathbb{R}^6$ is the feed forward spatial velocity, $\mathbf{K}_p \in \mathbb{R}^{6 \times 6}$ a diagonal matrix with the feedback gain constants on the main diagonal, $\mathbf{p}_r, \mathbf{p} \in \mathbb{R}^3$ the reference/actual position of the robot and $\theta \hat{\boldsymbol{\omega}}_r^a \in \mathbb{R}^3$ the difference in orientation between actual and reference pose using a singularity-free representation [19]. Similarly, a joint positioning task can be integrated by using

$$\dot{\mathbf{q}}_d = \dot{\mathbf{q}}_r + \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) \quad (9.6)$$

and setting $\mathbf{v}_d = \dot{\mathbf{q}}_d$, as well as $\mathbf{J}_u = \mathbf{I}$, where \mathbf{I} is the $(6+p) \times (6+p)$ identity matrix.

In (9.2) only velocity limits can be integrated as box constraints in a straightforward way. The integration of joint position limits is not easily possible since it may lead to infeasibility of the optimization problem and discontinuities in the control law [20]. Therefore, avoidance of joint position limits is considered into (9.2) on task level. The avoidance function is implemented as repulsive potential field:

$$\dot{q}_d = k_p \frac{u - u_0}{d} S(d), \quad (9.7)$$

where $d = |u - u_0|$ is the distance to a position limit u_0 (upper or lower, depending on which is closer) of a single actuator and $S(d)$ is a Sigmoid function of this distance, which ensures a smooth control signal when approaching an actuator limit. To integrate joint limit avoidance with other tasks, $\mathbf{v}_d = (\dot{q}_{d,1}, \dots, \dot{q}_{d,p})^T$, $\mathbf{J}_u = \mathbf{I}$, and $\mathbf{w} = (w_1, \dots, w_p)$ is chosen, where:

$$w_i = \begin{cases} (d_0 - d_i)/d_0 & \text{if } d_i < d_0, \\ 0 & \text{else} \end{cases} \quad (9.8)$$

where d_i is the distance to the nearest position limit of the i -th actuator and d_0 the activation distance. This way, the avoidance task is smoothly

activated when approaching an actuator limit, while the motion is unconstrained elsewhere.

9.2.2 Acceleration-based WBC

On acceleration level, the proposed approach sets up and solves the following optimization problem:

$$\begin{aligned}
 \min_{\ddot{\mathbf{x}}, \boldsymbol{\tau}_u, \mathbf{f}_j} \quad & \|\mathbf{J}_u \ddot{\mathbf{x}} + \dot{\mathbf{J}}_u \dot{\mathbf{x}} - \dot{\mathbf{v}}_d\|_2 \\
 \text{s.t.} \quad & \mathbf{H}_u \ddot{\mathbf{x}} + \mathbf{C}_u = \boldsymbol{\tau}_u + \sum_j \mathbf{J}_{cu}^j \mathbf{f}_j \\
 & \mathbf{J}_{u,c}^j \ddot{\mathbf{x}} = -\dot{\mathbf{J}}_{u,c}^j \dot{\mathbf{x}}, \quad \forall j \\
 & \boldsymbol{\tau}_{um} \leq \boldsymbol{\tau}_u \leq \boldsymbol{\tau}_{uM}
 \end{aligned} \tag{9.9}$$

Here, the optimization variables comprise the actuator and floating base accelerations $\ddot{\mathbf{x}} = (\ddot{\mathbf{y}}_b \ \ddot{\mathbf{u}}) \in \mathbb{R}^{6+p}$, the actuation forces/torques $\boldsymbol{\tau}_u \in \mathbb{R}^p$ and the external contact wrenches $\mathbf{f}_j \in \mathbb{R}^6$, $\forall j$. Again, only a single task is considered. The term $\dot{\mathbf{v}}_d$ describes the desired spatial acceleration of that task and \mathbf{J}_u the related task Jacobian. As in the velocity-based WBC approach, the task Jacobians are computed using (9.3). The optimization problem is solved subject to the equations of motion (first row), rigid contact constraints (second row) and actuator force/torque limits, where $\boldsymbol{\tau}_{um}$, $\boldsymbol{\tau}_{uM}$ are the lower and upper force/torque limits of the actuators.

Similar to (9.3), the inertia matrix in actuation space \mathbf{H}_u can be computed by separating the independent joint space inertia matrix \mathbf{H}_y into the submatrices related to the floating base and those related to the remaining robot joints. Thus, the inertia matrix in independent joint space is given by:

$$\mathbf{H}_y = \begin{pmatrix} \mathbf{H}_y^b & \mathbf{H}_y^{br} \\ \mathbf{H}_y^{br} & \mathbf{H}_y^r \end{pmatrix} = \mathbf{G}^T \mathbf{H} \mathbf{G}, \tag{9.10}$$

where \mathbf{G} is again the loop closure Jacobian and \mathbf{H} is the joint space inertia matrix of the entire spanning tree. The submatrix $\mathbf{H}_y^b \in \mathbb{R}^{6 \times 6}$ is the upper left part of \mathbf{H}_y related to the floating base joints, $\mathbf{H}_y^r \in \mathbb{R}^{p \times p}$ is the lower right part of \mathbf{H}_y related to the independent robot joints, and $\mathbf{H}_y^{br} \in \mathbb{R}^{p \times 6}$, $\mathbf{H}_y^{rb} \in \mathbb{R}^{6 \times p}$ are the remaining submatrices. Given this separation, \mathbf{H}_u can be derived as:

$$\mathbf{H}_u = \begin{pmatrix} \mathbf{H}_y^b & \mathbf{H}_y^{br} \mathbf{G}_u^{-1} \\ \mathbf{G}_u^{-T} \mathbf{H}_y^{br} & \mathbf{G}_u^{-T} \mathbf{H}_y^r \mathbf{G}_u^{-1} \end{pmatrix}. \tag{9.11}$$

The bias forces \mathbf{C}_u can be computed in analogy to this, namely by separating the bias forces of the entire spanning tree \mathbf{C} into floating base and actuated joints $\mathbf{C} = (\mathbf{C}_b \ \mathbf{C}_r)^T$. Thus, the bias forces including Coriolis-Centrifugal and gravity effects can be computed as:

$$\mathbf{C}_u = (\mathbf{C}_b \ \mathbf{G}_u^{-T} \mathbf{G}^T (\mathbf{C}_r + \mathbf{H}\mathbf{g} - \mathbf{H}\mathbf{G}\mathbf{G}_u^{-1} \mathbf{g}_u))^T, \quad (9.12)$$

where $\mathbf{g} = \dot{\mathbf{G}}\dot{\mathbf{y}}$ and $\mathbf{g}_u = \mathbf{G}_u\mathbf{g}$.

To model robot tasks, various controllers are provided at the acceleration-level, similar to the velocity-based WBC approach. For example, Cartesian positioning tasks are implemented analogous to (9.5):

$$\dot{\mathbf{v}}_d = \dot{\mathbf{v}}_r + \mathbf{K}_d(\mathbf{v}_r - \mathbf{v}) + \mathbf{K}_p \begin{pmatrix} \mathbf{p}_r - \mathbf{p} \\ \theta \hat{\boldsymbol{\omega}}_r^a \end{pmatrix}, \quad (9.13)$$

where $\dot{\mathbf{v}}_r$, \mathbf{v}_r , \mathbf{p}_r are the reference spatial acceleration, velocity and position, \mathbf{v} , \mathbf{p} the actual spatial velocity and position, \mathbf{K}_p , \mathbf{K}_d the proportional and derivative gain matrices and $\theta \hat{\boldsymbol{\omega}}_r^a \in \mathbb{R}^3$ the difference in orientation between actual and reference pose. Joint positioning and joint limit avoidance are implemented on acceleration level analogously to (9.6) and (9.7).

Although the WBC framework provides further controllers for, e.g., obstacle avoidance or force/torque control, details are skipped here for the sake of brevity. An experimental evaluation of the WBC framework is presented in the following section.

9.3 Experimental results

This section presents experimental results to evaluate the capabilities of the proposed WBC framework for series-parallel hybrid robots. First, its ability to exploit the entire workspace of a series-parallel hybrid robot on position, velocity and torque-level is evaluated. Thereby, the admissible workspace is compared against a similar WBC approach for tree-type robots. Second, the computational performance of the WBC approach is studied for robot models of different complexity. Again, as a benchmark the proposed approach is compared against a tree-type WBC. The evaluation is performed on two different humanoid robots, RH5 [13] and RH5 Manus [15], in simulation and on the real systems.

9.3.1 Application of box constraints in actuation space

As mentioned in previous sections, existing WBC frameworks cannot integrate actuator constraints of parallel mechanisms as they use an abstraction of their mechanical structure in independent joint space. The reachable workspace of the parallel mechanisms cannot be captured accurately by means of box constraints in independent coordinates. Consequently, a conservative approximation of the reachable workspace must be used in order to avoid deadlock situations or even mechanical damage. Such an approximation usually leads to a reduction of the admissible position workspace, or the admissible velocity and force/torque range as explained in Section 9.1. In contrast, the WBC approach proposed in this chapter can exploit the entire position, velocity, and force/torque range of a series-parallel hybrid robot. This is demonstrated in the following sections.

9.3.1.1 Position constraints

First, the approach is evaluated regarding the application of box constraints on position level. Experimental evaluation is performed on the RH5 humanoid, which has a 32 dof series-parallel hybrid structure. For evaluation, the velocity-based WBC described by (9.2) is used and compared to an analogous WBC implementation for tree-type robots. Two tasks are implemented:

1. Squatting: Follow a squatting trajectory by translating the CoM ± 0.2 m vertically to the ground. This task is regulated using the controller in (9.5). The task weights are all set to fixed values $w_i = 1, \forall i$.
2. Joint Limit Avoidance: Avoid actuator position limits using the potential field approach described by (9.7). The task weights are computed dynamically using the approach in (9.8).

Considering the squatting motion, the parallel ankle mechanism is of particular interest. The joints limits of this mechanism in actuation space are $[0.0647, -0.0449]$ in meters, i.e. each of the four actuators can operate within this range. In independent joint space the limits are ± 0.5236 for the pitch and ± 0.7850 for the roll movement.

Fig. 9.2 compares the squatting motion of the tree-type WBC (Fig. 9.2(a)) with the WBC approach for series-parallel hybrid robots (Fig. 9.2(b)). The plots show the trajectories of the CoM (top), the ankle actuators (middle) and the independent ankle pitch joint (bottom). For squatting, both ankle joints perform a pure pitch movement in independent joint space and all four ankle actuators, denoted as L_1, L_2, R_1, R_2 here, move identically. The tree-type WBC approach models the joint position limits

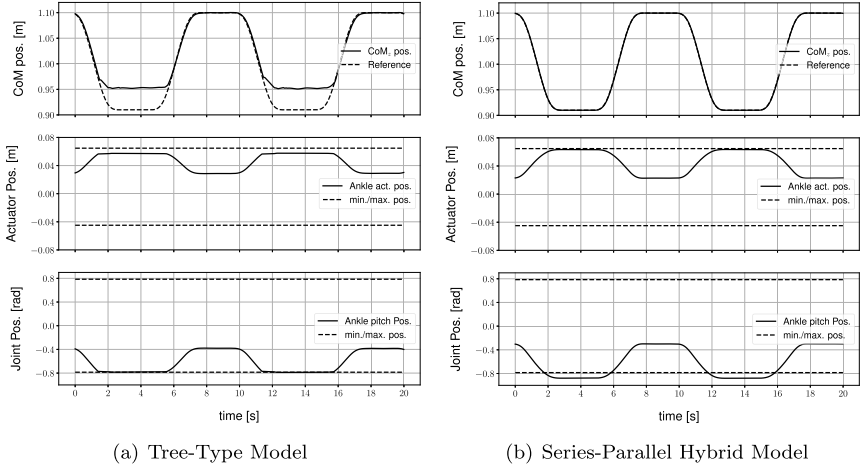


Figure 9.2 Comparison of squatting motion on RH5 using WBC with abstract/tree-type model and series-parallel hybrid model.

in independent joint space. As a result, the actuation space cannot be fully exploited, and the CoM trajectory is impaired by the position constraints. In contrast, the WBC approach for series-parallel hybrid robots models the position limits directly in actuation space. Consequently, the actuation space can be fully exploited, and the CoM trajectory is tracked accurately. While the tree-type WBC approach allows squatting movements in a CoM range of 0.957–1.100 m, the hybrid approach can perform squats in a range of 0.911–1.100 m. In total, the hybrid WBC approach provides a 4.5 cm or approx. 25% larger CoM workspace in this case. Fig. 9.3 shows the resulting motion of the CoM in the xz -plane, as well as screenshots from a video illustrating the results.

9.3.1.2 Velocity constraints

Next, the approach is evaluated regarding the application of box constraints on velocity-level. As experimental platform, the RH5 Manus robot is used, which is a 20-dof upper body humanoid with series-parallel hybrid structure. For evaluation, the velocity-based WBC approach described by (9.2) is used. The robot is supposed to perform highly dynamic boxing motions, which are generated using offline trajectory optimization based on Differential Dynamic Programming (DDP) [22]. Two tasks are implemented in this case, one Cartesian positioning task for each end effector, where both tasks share the torso dof for execution. Considering the boxing movement,

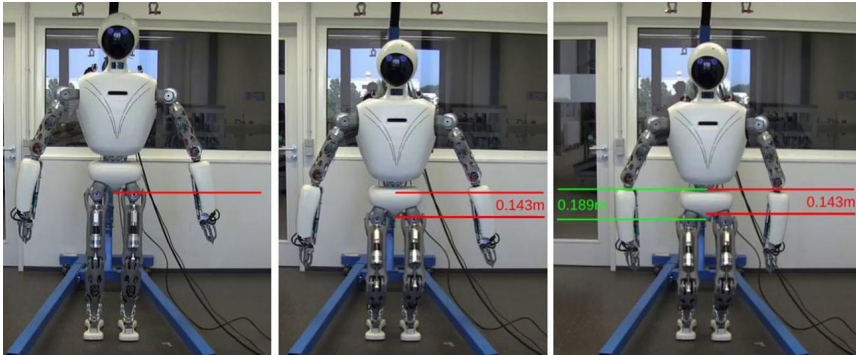


Figure 9.3 Squatting motions on the RH5 humanoid (Screenshots from video [21]). Left: Initial position, mid: Final position using simplified tree model, right: Final position using full hybrid model.

the linear elbow mechanism of RH5 Manus is of particular interest. The velocity limits of the mechanism in independent joint space are $\pm 3.09s^{-1}$ and in actuation space $\pm 0.266m/s$.

As explained in Section 9.1, the maximum elbow velocity in independent joint space is configuration-dependent. Using a tree-type WBC approach would necessarily underestimate the admissible velocity range for most configurations. In contrast, using the proposed WBC approach, the velocity limits can be modeled in actuation space of this closed-loop mechanism.

Fig. 9.4 compares the resulting boxing movements of a WBC approach with tree-type (Fig. 9.4(a)) and with series-parallel hybrid model (Fig. 9.4(b)). The plots show the linear components of the end effector velocity of the left arm (top), the elbow actuator velocity (middle) and the elbow velocity in independent joint space (bottom). When modeling the box constraints in independent joint space using a tree-type WBC approach, the maximum actuator velocities of the elbow mechanism cannot be fully exploited. In contrast, using the hybrid WBC approach, the actuator velocity range can be fully exploited, which leads to higher velocities in independent joint space. As shown in Fig. 9.4(b), the velocities in independent joint space are much higher than $3.09rad/sec$, which is the conservative maximum that must be selected in independent coordinates. However, the end effector velocities are identical in both cases, as the other arm joints compensate for the reduced elbow velocity in the tree-type case. Nevertheless, using the proposed WBC approach the maximum admissible

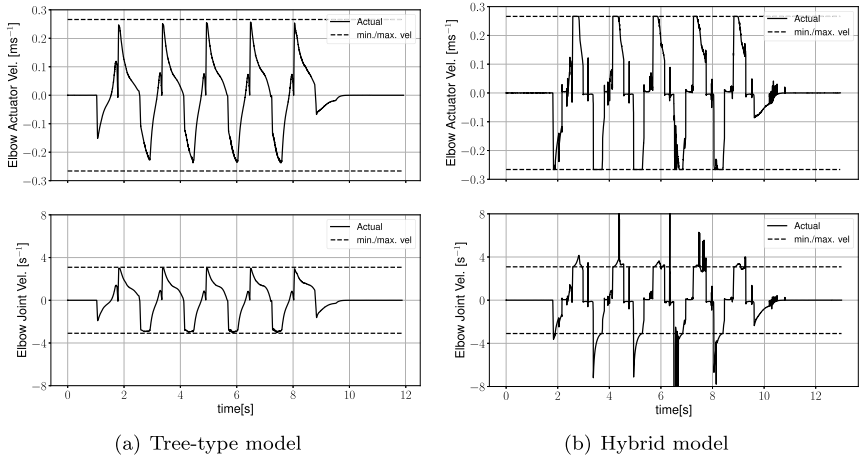


Figure 9.4 Comparison of boxing motions on RH5 Manus using WBC with abstract/tree-type model and series-parallel hybrid model.



Figure 9.5 Executing boxing motions on RH5 Manus, screenshots from video [21].

elbow joint velocity is more than twice as high as in the tree-type case, which allows much more dynamic movements. Fig. 9.5 shows snapshots from a video documenting the boxing experiment.

9.3.1.3 Force/torque constraints

Experimental results on force/torque level are skipped here for the sake of brevity. However, they are expected to produce comparable results to the experiments regarding the integration of velocity-level box constraints. As described in Section 9.1, the maximum torque of the independent elbow joint of RH5 Manus is configuration-dependent, analogous to the maximum velocity. When specifying a box constraint on the maximum torque in independent joint space, a conservative limit as indicated by the dashed line must be assumed. Thus, a tree-type WBC cannot exploit the entire force/torque range. In contrast, the WBC approach for series-parallel hybrid robots can exploit the entire force/torque of the actuators contained in parallel submechanisms. This can be especially advantageous for applica-

tions like humanoid walking or jumping, which require high dynamics and an optimal exploitation of the robot's capabilities.

9.3.2 Computational performance

This chapter evaluates the computational performance of the WBC approach for series-parallel hybrid robots. To examine the effect of increasing model complexity, 16 different robot models, as illustrated in Table 9.2 are compared. For each robot model, model type (serial vs. series-parallel hybrid) and WBC type (velocity-based vs. acceleration-based), the table shows the model size and the resulting QP size. The model size includes the number of spanning tree joints n , number of independent joints m and number of actuated joints p . For serial models, the number of spanning tree joints and independent joint are identical, i.e., $m = n$. For fully actuated robots, the number of actuated joints is identical to the number of independent joints, i.e., $p = m$. In the table, this is true for all robot models without floating base. The QP size is described by the number of decision variables q and the number of constraints c . The decision variables comprise the joint velocities for the velocity-based WBC as in (9.2) and the joint accelerations, actuation torques and external wrenches for the acceleration-based WBC as in (9.9). The number of constraints depends on the dimension of the equations of motion and the rigid contact constraints. The actuator limits $\dot{\mathbf{q}}_m, \dot{\mathbf{q}}_M$ and $\boldsymbol{\tau}_m, \boldsymbol{\tau}_M$ are not included here, as qpOASES implements them as simple decision variable bounds, which are processed much faster than task-level constraints. For the experimental evaluation, similar tasks are defined for each robot model. For the RH5 single leg model, a Cartesian positioning task as in (9.5) and (9.13) is specified. For the RH5 legs and full RH5 model, a Cartesian CoM tracking task is defined, where the foot links are subject to rigid contact constraints. For the RH5 Manus model, a Cartesian positioning task for each gripper is defined is used. As performance indicator the computation time for a complete update cycle is measured, which comprises the time for updating the QP and solving it using a standard QP solver [23]. Experiments are performed in simulation using the Raisim physics simulator [24] and run on a standard laptop with Intel Core i7-8565U CPU (8 x 1.8 GHz).

Fig. 9.6 shows the computation times when applying the WBC tasks described earlier using models of different complexity. For each model, a classical tree-type WBC approach is compared with the analogous series-parallel hybrid WBC approach. It can be observed that the computation

Table 9.2 Complexity of different robots models.

Robot Type	Model Type	WBC Type	Model Size			QP Size	
			n	m	p	q	c
RH5 Manus*	serial	(9.2)	20	20	20	20	0
	hybrid		61	20	20	20	0
	serial	(9.9)	20	20	20	40	20
	hybrid		61	20	20	40	20
RH5 One Leg*	serial	(9.2)	6	6	6	6	0
	hybrid		18	6	6	6	0
	serial	(9.9)	6	6	6	12	6
	hybrid		18	6	6	12	6
RH5 Both Legs**	serial	(9.2)	18	18	12	18	12
	hybrid		42	18	12	18	12
	serial	(9.9)	18	18	12	42	30
	hybrid		42	18	12	42	30
RH5**	serial	(9.2)	38	38	32	38	12
	hybrid		83	38	32	38	12
	serial	(9.9)	38	38	32	82	50
	hybrid		83	38	32	82	50

*Fixed Base, **Floating Base

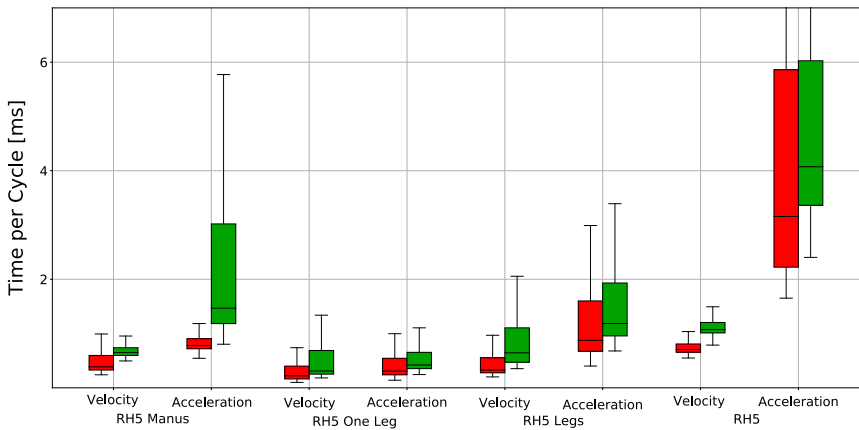


Figure 9.6 Comparison of serial vs. hybrid model regarding the computation time for a complete cycle (model update, scene update, QP solving) for different robot models and scenes.

times for the hybrid WBC are around 1.2–2.5 times larger than for the tree-type WBC on average. There are two reasons for this. On the one hand, the

series-parallel hybrid models have a larger spanning tree than the tree-type models, while the QP size is identical in both cases. On the other hand, the consideration of loop closure constraints for the parallel submechanisms requires additional computational effort. For the most complex models, this effect is significant. However, the greater computational effort is justified given the advantages that come with model fidelity. Also, the approach is able to control the RH5 humanoid robot, which corresponds to the most complex model in Table 9.2, with a cycle of well below 10 ms.

9.4 Discussion and outlook

In this chapter, a novel, computationally efficient approach for WBC of series-parallel hybrid robots has been introduced. The approach uses the HyRoDyn library to specify the optimization problem inherent to WBC in actuation space of a series-parallel hybrid robots. This provides a better exploitation of the admissible workspace, more transparent behavior near singularities and more accurate dynamics than tree-type WBC approaches. The focus of this chapter is on the problem of workspace exploitation. Experimental results were provided on position- and velocity-level using two different humanoid robots. It has been shown that, compared to a tree-type WBC approach, a larger CoM workspace could be obtained when performing squatting movements on the RH5 humanoid. Also, larger joint space velocities could be achieved when performing boxing movements on the RH5 Manus humanoid.

Furthermore, the computational performance has been evaluated for robot models of different complexity. It has been shown that, although the computational effort increases with model fidelity, real-time control with < 10 ms cycle time can be achieved. A further reduction of the computational effort can be achieved by various measures, for example, by using a different solver, reducing the number of decision variables in the QP or by reducing model complexity. The latter can be achieved by using the full model only where it is absolutely required, e.g., when heavy masses are moved inside the parallel mechanisms. In [11], the effect of neglected dynamics in series-parallel hybrid mechanisms is studied.

As future work, further experimental evaluation of the proposed approach is planned. First, the effect of incorrectly modeled robot dynamics in WBC shall be investigated. For dynamic balancing, a greater model fidelity may have a significant effect on the performance of the whole-body controller. Secondly, more complex applications like dynamic walking shall

be tackled. Third, the approach can be transferred into the domain of trajectory optimization in order to produce dynamically consistent motions, which exploit the full capabilities of a series-parallel hybrid robot.

References

- [1] F.L. Moro, L. Sentis, *Whole-Body Control of Humanoid Robots*, Springer, Netherlands, Dordrecht, 2019, pp. 1161–1183, https://doi.org/10.1007/978-94-007-6046-2_51.
- [2] L. Sentis, O. Khatib, A whole-body control framework for humanoids operating in human environments, in: *Proceedings – IEEE International Conference on Robotics and Automation*, May 2006, 2006, pp. 2641–2648, <https://doi.org/10.1109/ROBOT.2006.1642100>.
- [3] A. Del Prete, F. Nori, G. Metta, L. Natale, Prioritized motion-force control of constrained fully-actuated robots: ‘task space inverse dynamics’, *Robotics and Autonomous Systems* 63 (2015) 150–157, <https://doi.org/10.1016/j.robot.2014.08.016>, arXiv:1410.3863.
- [4] T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, J.E. Pratt, Design of a momentum-based control framework and application to the humanoid robot atlas, *International Journal of Humanoid Robotics* 13 (1) (2016) 1650007, <https://doi.org/10.1142/S0219843616500079>, software documentation: https://bitbucket.org/ihmcrobotics/ihmc_ros/wiki/whole-body-controller.
- [5] M. Liu, Y. Tan, V. Padois, Generalized hierarchical control, *Autonomous Robots* 40 (1) (2016) 17–31, <https://doi.org/10.1007/s10514-015-9436-1>.
- [6] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, R. Tedrake, Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot, *Autonomous Robots* 40 (3) (2016) 429–455, <https://doi.org/10.1007/s10514-015-9479-3>.
- [7] Andrea Del Prete, Nicolas Mansard, Oscar E. Ramos, Olivier Stasse, Francesco Nori, Implementing torque control with high-ratio gear boxes and without joint-torque sensors, *International Journal of Humanoid Robotics* 13 (1) (2016), <https://doi.org/10.1142/S0219843615500449>.
- [8] OCRA – Optimization-based Controllers for Robotics Applications, <https://orca-controller.readthedocs.io/en/master>. (Accessed 29 July 2021).
- [9] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, J. De Schutter, iTASC: a tool for multi-sensor integration in robot manipulation, in: *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 426–433, <https://doi.org/10.1109/MFI.2008.4648032>, software documentation: <https://www.orocos.org/itasc.html>.
- [10] ControlIt! – a whole body operational space control middleware, <https://github.com/liangfok/controlit>. (Accessed 29 July 2021).
- [11] S. Kumar, J. Martensen, A. Mueller, F. Kirchner, Model simplification for dynamic control of series-parallel hybrid robots – a representative study on the effects of neglected dynamics, in: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5701–5708, <https://doi.org/10.1109/IROS40897.2019.8967786>.
- [12] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, 2019, <https://nbn-resolving.org/urn:nbn:de:gbv:46-00107793-11>.
- [13] J. Esser, S. Kumar, H. Peters, V. Bargsten, J. de Gea Fernández, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid

- robot, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), IEEE, 2021, pp. 400–407, shifted from 2020 to 2021.
- [14] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: J. Lenarcic, V. Parenti-Castelli (Eds.), *Advances in Robot Kinematics 2018*, Springer International Publishing, Cham, 2019, pp. 431–439.
- [15] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 01–07, <https://doi.org/10.1109/ICRA46639.2022.9811843>.
- [16] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, *Journal of Mechanisms and Robotics* 12 (2) (2020) 021114, <https://doi.org/10.1115/1.4045941>, https://asmedigitalcollection.asme.org/mechanismsrobotics/article-pdf/12/2/021114/6648769/jmr_12_2_021114.pdf.
- [17] R. Featherstone, *Rigid Body Dynamics Algorithms*, Springer US, 2014, <https://doi.org/10.1007/978-1-4899-7560-7>.
- [18] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*, Springer Science & Business Media, 2010, <https://doi.org/10.1007/978-1-4419-7267-5>.
- [19] K. Lynch, F. Park, *Modern Robotics*, Cambridge University Press, 2017, <https://books.google.de/books?id=YUkTnQAACAAJ>.
- [20] F. Flacco, A. De Luca, O. Khatib, Control of redundant robots under hard joint constraints: saturation in the null space, *IEEE Transactions on Robotics* 31 (3) (2015) 637–654, <https://doi.org/10.1109/TRO.2015.2418582>.
- [21] D. Mronga, Whole-body control of series-parallel hybrid robots, <https://www.youtube.com/watch?v=yv6vCUGV6zc>, 2021.
- [22] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, N. Mansard, Crocodyl: an efficient and versatile framework for multi-contact optimal control, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 2536–2542, <https://doi.org/10.1109/ICRA40945.2020.9196673>.
- [23] H. Ferreau, C. Kirches, A. Potschka, H. Bock, M. Diehl, qpOASES: a parametric active-set algorithm for quadratic programming, *Mathematical Programming Computation* 6 (4) (2014) 327–363.
- [24] J. Hwangbo, J. Lee, M. Hutter, Per-contact iteration method for solving contact dynamics, *IEEE Robotics and Automation Letters* 3 (2) (2018) 895–902, www.raism.com.

CHAPTER 10

Whole-body trajectory optimization

Melya Boukheddimi^a, Rohit Kumar^a, Shivesh Kumar^a,
Justin Carpentier^b, and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bINRIA Paris, France

^cWorking Group Robotics, University of Bremen, Bremen, Germany

10.1 Introduction

The recent developments in robotics have seen a large adaptation of closed-loop mechanisms in various robots like exoskeletons [1], multi-legged robots [2], humanoid robots [3,4], etc. These parallel mechanisms provide higher stiffness, high payload capacities, higher precision, etc. Series-parallel hybrid robots can be defined as the combination of serial chain and parallel mechanisms that can bring together the advantages of both topologies. An extensive survey on these is available in [5]. They are often combined to closely mimic the human and animal capabilities that require high stiffness, optimum mass, inertia distribution properties, etc. While there are many advantages, these series-parallel hybrid robots also inherit the kinematic complexities of both serial and parallel architectures.

Currently, the trajectory optimization approaches are very popular and powerful methods for motion planning. They can help in generating various complex movements for multi-body systems [6]. Most of the trajectory optimization problems are usually based on tree type systems as they are easier to model and control. Many of these solvers allow modeling of external kinematic constraints acting on the robot, e.g., humanoid legs in standing or multi-contact scenarios [7–9]. Parallel robots have the advantages of high accuracy and rigidity with a large payload, however, they involve loop closure constraints that are difficult to design and control. The optimization process is usually difficult or time-consuming, which makes it impossible to use in real-time control [10]. Some researchers have tried to control these systems using trajectory optimization approaches, such as [11], where a cascade controller was proposed to control a pneumatic driven parallel

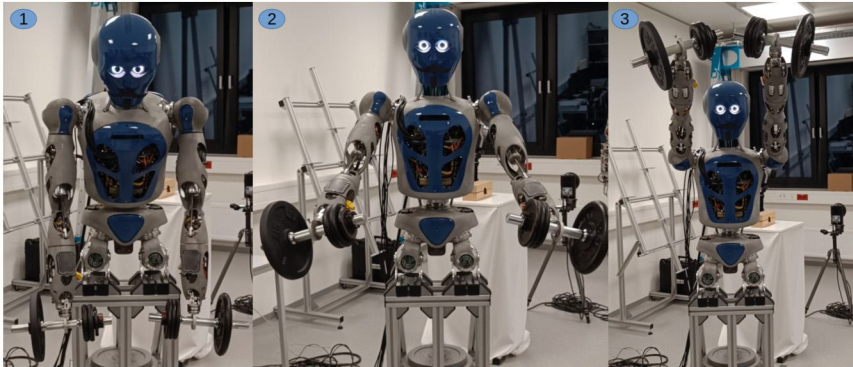


Figure 10.1 Screenshot of the RH5 Manus robot performing a 20 kg lifting motion. The motion is generated using the constrained series-parallel model of the robot in the trajectory optimization process.

robotic platform with optimal parameter tuning. However, the intended tasks are industrial tasks, which means that they are repetitive and kinematic. In [12], the authors proposed an optimal controller based on the firefly algorithm to generate optimal trajectories for a hydraulically parallel robot. In contrast, a complex series-parallel hybrid robot is difficult to solve since it is a combination of many parallel robots. In addition to optimal control formulation, resolving loop closure constraints is a burden on the optimization. When it comes to direct methods, [13] has proposed DIRCON, a direct collocation algorithm which can effectively deal with kinematic constraints in both trajectory generation and stabilization steps with third-order integration accuracy. However, the method was applied to deal only with external kinematic constraints arising from contacts. Various shooting methods have also been proposed in the literature for whole-body trajectory optimization [14] that can deal with kinematic constraints like contacts, but have not been studied for kinematic loops within the robot. Most previous studies using shooting methods considered a serial abstraction of the series-parallel hybrid robot and the results were mapped to the closed-loop mechanisms present in the system [4,15] using an kinematic mapping. This process has the following disadvantages:

- Box constraints used to model the physical limitations of the mechanisms either overestimate or underestimate the effective workspace of the robot (see Section 10.3).

- Parallel mechanisms may be subject to singularities that are not taken into account in the optimization problem while working with serial models.
- The optimization formulation is not accurate since it does not take into account the full dynamics of the closed-loop mechanisms of the system.

To address these issues, we propose a first study case on resolving the loop-closure constraints of the robot within the trajectory optimization process. To this end, we use the open-source software Pinocchio with its proximal formulation of the constrained dynamics. This approach allows us to converge to an optimal solution according to the least squares principle, even in the context of singularities. Among the optimization methods available in the literature, the differential dynamics programming (DDP) approach was used to generate optimal trajectories with respect to the constrained dynamics. We consider the weight lifting task for the RH5 Manus humanoid and demonstrate that by planning the trajectories directly in the actuation space, we can exploit the full capabilities of the robot, which is not possible when working with a serial abstraction of the robot model. Results are shown in simulation as well as experiments.

The chapter is organized as the following: Section 10.2 presents the mathematical preliminaries for whole body trajectory optimization with kinematic constraints. Section 10.3 presents the design of the study on RH5 Manus humanoid platform for weight lifting task. Section 10.4 presents the results and discussion and Section 10.5 discusses and concludes the paper.

10.2 Mathematical background

Series-parallel hybrid robots are subjected to large number of holonomic constraints. These constraints can act *externally* on the system, e.g., multiple contacts with the environment or *internally*, e.g., loop closure constraints for closed-loop mechanisms present within the system. This section presents the mathematical preliminaries of constrained dynamics formulation and the corresponding whole-body trajectory optimization problem.

10.2.1 Constrained multi-body dynamics

The unconstrained multi-body dynamics of the robot is written in the following canonical form:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} + \sum_{k=1}^K \mathbf{J}_k^T(\mathbf{q})\boldsymbol{\phi}_k, \quad (10.1)$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ corresponds to the vector of generalized positions, velocities, and accelerations, $\mathbf{M}(\mathbf{q})$ is the generalized mass-inertia matrix, $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$ is the bias force vector, which includes Coriolis-Centrifugal, and gravity forces and $\boldsymbol{\tau}$ is the vector of generalized torques. K accounts for the additional kinematic constraints on the robot, such as external contact with the environment and internal kinematic loops. At position level, the kinematic constraints could be implicitly written as $\mathbf{f}_c(\mathbf{q}) = 0$. However, since we are solving the dynamics in the acceleration space, it is common to use the second derivatives of this constraint in our problem:

$$\mathbf{J}_k(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}_k(\mathbf{q})\dot{\mathbf{q}} = \mathbf{a}_c^*. \quad (10.2)$$

$\mathbf{J}_k(\mathbf{q})$ is the Jacobian matrix corresponding to the k^{th} application of constraint on the robot. $\boldsymbol{\phi}_k = [\boldsymbol{\lambda}_k \quad \eta_k]^T$ is the vector of the dual external forces ($\boldsymbol{\lambda}_k$) and the torques (η_k) that corresponds to the k^{th} constraints. \mathbf{a}_c^* is the redesired acceleration with corrective terms for constraint satisfaction. Under kinematic constraints (i.e., kinematic loop closure), the dynamics of the robot is subject to the constrained equations of motion presented in (10.1) and (10.2), and this can be written as an optimization problem under equality constraints. The solution of the associated Lagrangian of this constrained dynamics (see [16] for full expansion) is then given by:

$$\begin{bmatrix} \mathbf{0} & \mathbf{J}_k \\ \mathbf{J}_k^T & \mathbf{M} \end{bmatrix} \begin{bmatrix} -\boldsymbol{\phi}_k \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} -\dot{\mathbf{J}}_k(\mathbf{q})\dot{\mathbf{q}} + \mathbf{a}_c^* \\ \boldsymbol{\tau} - \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix}. \quad (10.3)$$

Multiple formulations have been proposed to solve this problem, for example, using Schur's components of the Cholesky-factorized constrained dynamics matrix [14] [17]. We use the recently proposed constrained dynamics algorithm in Pinocchio [16]. It reformulates the problem in a proximal way and provides a sparse way to efficiently handle this problem and its derivatives. We use the Pinocchio C++ Library for our formulations here, since it provides us with an efficient implementation of the unconstrained as well as constrained dynamics along with their analytical derivatives for help in optimization [16] [18][19].

10.2.2 Trajectory optimization formulation

The discrete time optimal control problem can be written as follows:

$$\min_{\mathbf{x}, \mathbf{u}} l_N(\mathbf{x}_N) + \sum_{t=0}^{N-1} l(\mathbf{x}_t, \boldsymbol{\tau}_t) dt \quad (10.4a)$$

$$s.t. \quad \mathbf{x}_0 = \mathbf{f}_0, \quad (10.4b)$$

$$\forall i \in \{0 \dots N-1\}, \quad \mathbf{x}_{i+1} = \mathbf{f}_t(\mathbf{x}_i, \boldsymbol{\tau}_i) \quad (10.4c)$$

- $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$: robot state,
- \mathbf{u} : robot control,
- $\boldsymbol{\tau} \in \mathbb{R}^{n_u}$: actuator effort,
- N : nodes number for the discretized trajectory,
- l_N : terminal cost model, applied on the last node of the trajectory,
- l : running cost model, applied on all remaining nodes,
- \mathbf{f}_0 : the initial state of the problem $(\mathbf{q}_0, \dot{\mathbf{q}}_0)$,
- \mathbf{f}_t : the discretization of the robot dynamics (10.3).

Optimal trajectories are computed using the Box Feasibility DDP (Box-FDDP) solver proposed by the open-source C++ library Crocodyl [14]. The Feasibility DDP [20] enables us to overcome the numerical limitations of the single-shot original DDP formulation [21]. The Box-FDDP [22] gives us the ability to reason about the torque limits of the robot.

10.3 Experimental design

This section presents the experimental design, where we consider the fixed base RH5 Manus [15], a series-parallel hybrid robot with multiple kinematic loops for a weight-lifting task using whole-body trajectory optimization. Firstly, it describes the complete hybrid upper body of the robot with closed-loop mechanisms identifying the theoretical limits of the full robot and its tree abstraction. Next, it presents optimal control formulation for weight lifting with tree and full hybrid robot models.

10.3.1 Closed-loop mechanisms in RH5 Manus robot

The fixed-base model of RH5 Manus robot consists of a total 61 spanning tree joints ($n = 61$). Among these, there are 20 independent joints ($m = 20$) and 20 active joints ($p = 20$). The topological graph of the robot can be seen in Fig. 10.2. All the independent joints are shown as green edges and the actuated joints are shown as red edges. Remaining spanning tree joints are passive in nature. The cut joints for loop closure is denoted by dotted lines. All closed-loop mechanism in the robot is shown in blue boxes.

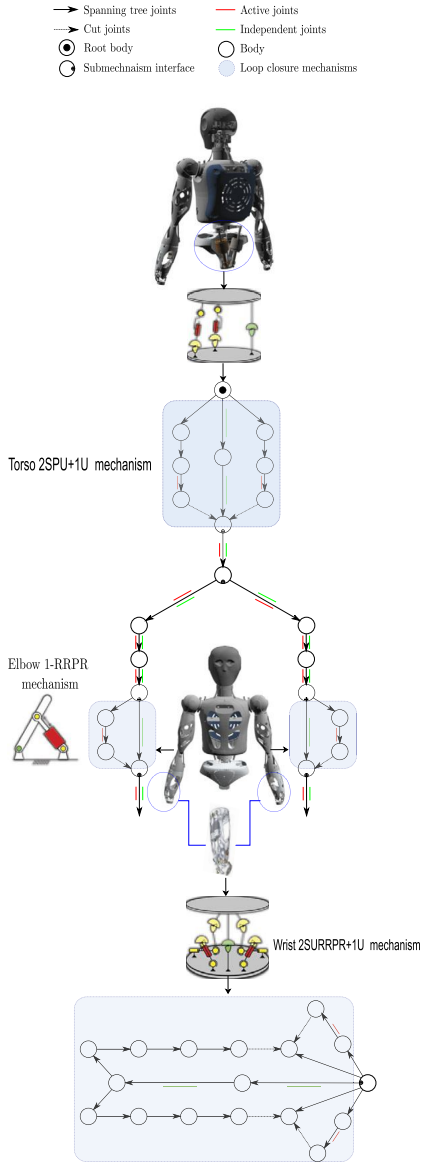


Figure 10.2 Upper body of RH5 Manus robot.

The series-parallel hybrid system in Fig. 10.2 can be represented as a tree type composition of 10 submechanisms. There are 5 serial chain submechanisms and 5 closed-loop submechanism. The first closed-loop

submechanism connected to the root of the graph is a multi closed-loop torso mechanism of type $2SPU+1U$ [23] and is actuated by two prismatic actuators (Joints 5 and 8) located on the left and right of the submechanism, respectively. Pitch and roll movements denotes the independent coordinates (Joints 1 and 2) of the submechanism. Each cut joint is a spherical joint and imposes 3D translation constraint on the submechanism.

The second closed-loop submechanism present in both arms of the robot is a planar closed-loop elbow mechanism of type $RRPR$ and is actuated by a prismatic actuator (joints 15 and 38 in right and left elbows). The elbow rotation is chosen as the independent coordinate (joints 13 and 36 in right and left elbows). In this submechanism, the cut joint is a revolute joint that imposes planar translation constraints.

The third closed-loop submechanism in both arms is complex multi-loop closure wrist mechanism of type $2SU[RRPR]+1U$ [24] and is actuated by two prismatic actuators (joints 28, 31, 51, and 54). Wrist pitch and yaw movements represents the independent coordinates (joints 17, 18, 40, and 41). There are multiple cut joints in the mechanism. One of the cut joints on both sides of the submechanism is a spherical joint that imposes 3D translation constraint. Other cut joint is a revolute joint on both sides that imposes the planar translation constraints.

10.3.2 Box constraints

In an optimization problem, the physical limits of the robot's joints are generally modeled as box constraints [22,25]. However, this is true for serial or tree type systems, but is not necessarily a good argument for closed-loop mechanism [26]. In closed-loop mechanisms, the configuration space can be different from actuation space. When independent joints are considered, the physical limits of the actuators are masked and hence not exploited. There can be various configurations that are feasible in actuation space of the closed-loop mechanism but cannot be exploited while considering box constraints. The configuration analysis in independent joints space and actuation space of the closed-loop mechanisms in RH5 Manus have already been studied [24,27].

10.3.2.1 Torso

As an example, consider the torso submechanism in RH5 Manus robot in Fig. 10.3(a). The full admissible configuration space in the actuation coordinates is shown in Fig. 10.3(b). The box constraints are shown by blue lines in the plot as the actuator position limits. When this mechanism

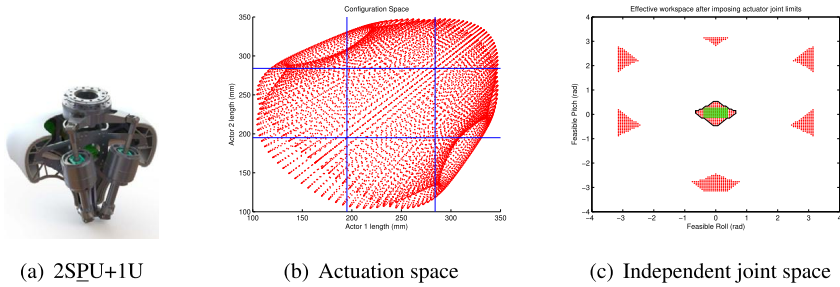


Figure 10.3 Box constraints for RH5 Manus torso [27].

is modeled using independent joints, the configuration space can be seen in Fig. 10.3(c). The black curves represents the effective actuator limits, mapped from actuation space to independent joint space. However, due to box constraints, only green region can be reached, acting as conservative limits. The full capabilities of the robot cannot be exploited.

10.3.2.2 Elbow

The same behavior can be observed on velocity and torque level for the elbow mechanism. The maximum velocity of the linear actuator depends on the joint configuration, shown in Table 10.2. A box constraint defined using the conservative choice of velocity (177 degrees/s) and torque (48 Nm) limits would ensure that the motion is always feasible but will miss out the opportunity to exploit the full potential of the robot capabilities. The robot is capable of providing maximum velocity of 633 degrees/s and maximum torque of 172 Nm in certain configurations in its workspace.

10.3.2.3 Wrist

The analysis of the wrist closed-loop mechanism in the robot gives the same insight as shown in Table 10.2. Being a 2 DOF parallel mechanism, the limits of pitch and yaw movements are configuration dependent which makes it difficult to work with both lower and upper limits of robot capabilities.

10.3.2.4 Overall ROM

The complete range of motion (ROM) for the linear actuators in the closed-loop mechanisms is summarized in Table 10.1. When box constraints for the actuators are mapped to independent joint space, the limits

are configuration-dependent and hence can take any value between maximum lower and maximum upper limits, summarized in Table 10.2.

Table 10.1 ROM of linear actuators used in RH5 Manus.

Actuator	ROM (mm)	Max. force (N)	Max. vel. (mm/s)
Wrist	[113, 178]	1094	200
Elbow	[173.8, 295.42]	2000	266
Torso	[195.0, 282.8]	2716	291

Table 10.2 ROM of the independent joints in RH5 Manus.

Joint	ROM (°)	Max. Torque (Nm)	Max. Velocity (°/s)
Elbow	[-105°, 0°]	48–172	177–633
Wrist Pitch	[-42.5°, 100°]	29–56	364–696
Wrist Yaw	[-32°, 34°]	38–50	386–499
Torso Pitch	[-20°, 30°]	380–493	184–238
Torso Roll	[-25.5°, 25.5°]	285–386	208–400

10.3.3 Tree abstraction of RH5 Manus

As multiple closed-loop mechanisms in a robot can become complex to resolve for errors in a computationally efficient manner, a tree-abstraction of the whole system is generally considered. A tree abstraction of the RH5 Manus robot in Fig. 10.2, is a tree-type system considering only independent coordinated or green edges of the topological graph. The tree abstraction of RH5 Manus robot consist of 20 degrees of freedom ($m = 20$) and Table 10.2 provides the joint limits for the closed-loop mechanisms. Note that a conservative choice would be to consider lower maximum values for velocities and torques, which may underestimate the capabilities of the robot. An ambitious choice would be to consider the upper maximum limits for velocities, which would overestimate the capabilities of the robot.

10.3.4 Optimal control formulation

The optimization problem was designed with the same costs and time horizon for the full hybrid model and the tree-type abstraction model, with 3 different choice of torque and velocity limits, namely: lower, middle, and

upper limits. The middle limits for torque and velocity of the joint are computed as the average of the lower and upper limits. The optimal trajectories for the task movements are formulated with a running cost model and a terminal cost model. The running cost models are defined by the following cost functions:

$$l = \sum_{c=1}^C \alpha_c \Phi_c(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}), \quad (10.5)$$

$\alpha_c \in \mathbb{R}$ is the applied weight to the cost function Φ_c . All the generated motions involves the same type of cost functions.

- *Wrist Target tracking*: The wrist position r_w (right and left) track the final wrist target placement for each desired end configuration.

$$\Phi_1 = \|r_w(t) - r_w^{ref}(t_N)\|_2^2 \alpha_1$$

- *Control regularization*: Minimization of the joint control for dynamically feasible motions.

$$\Phi_2 = \|\boldsymbol{\tau}(t)\|_2^2 \alpha_2$$

- *Posture regularization*: This cost manage the redundancy of the multi-body system.

$$\Phi_3 = \|\mathbf{q}(t) - \mathbf{q}^{ref}(t_N)\|_2^2 \alpha_3$$

Here, t_N refers to the final time of the motion. The terminal cost model includes only the wrist target tracking and the posture regularization cost. For the full model under the loop-closure constraint, $\alpha_2 = 0.6$ is higher than the one used in the tree model $\alpha_2 = 0.01$. In addition to these cost functions, in the full model formulation of the Optimal Control Problem (OCP), the proximal parameter has to be determined as well. In this 50 kg lifting simulation, the proximal parameter $prox = 1e - 5$. All the hyper-parameters in the OCP formulations are determined empirically.

10.4 Results

This section presents the results of trajectory optimization on fixed-base RH5 Manus robot for the weight lifting task. Firstly, the trajectory optimization results are reported when the robot lifts a total of 50 kg weight

(25 kg in each hand). Secondly, the trajectories for lifting a total of 20 kg weight (10 kg in each arm) are tested on the real robot. Lastly, the computational timings are reported for trajectory optimization in both serial abstraction model and full hybrid model of the robot. The simulation and experimental results can also be seen in the accompanying video.

10.4.1 Tree abstraction model vs full hybrid model

For trajectory optimization results, the main objective is to find the configurations that can be exploited in trajectory optimization, while considering the internal loop-closure constraints in the robot. The trajectory optimization formulation is done to lift 50 kg weight (25 kg in each arm). The formulation of OCPs based on the two models are set to reach the same end-effector targets, while lifting the same weight. Same cost models and time horizon were applied in each setting as defined in the previous section. The section is divided into two parts. The first reports the simulation results in tree-type or serial abstraction model of the robot and the second part shows the advantage of using complete hybrid model for trajectory optimization.

10.4.1.1 Tree abstraction model

The trajectory optimization results is shown in Fig. 10.4 for the considered three cases. It can be seen that the tree-type model of the robot is not able to reach the defined target in all the three cases. It performs badly with lower and middle maximum limits of the independent joints of the closed-loop mechanism. The trajectory optimization with the upper maximum limits on the independent joints performs better than the other two cases and is closer to the desired target.

In Fig. 10.5, same can be observed from the torques plot for left elbow and torso pitch joints from trajectory optimization. The dotted lines in the plot refer to the joint torque limits for the three cases. For lower and middle maximum limits, the torques are getting saturated for both left elbow and torso pitch joint during the trajectory. However, with upper maximum limits, elbow torques are well under the limits but torso pitch joint is saturated in the beginning. Considering conservative limits leads to task failure in the trajectory optimization. On the other hand, ambitious choice of the limits may find results in some cases that are not transferable on the robot.

To support the argument on ambitious choice of limits in independent joint configuration space, the results for upper maximum limits are

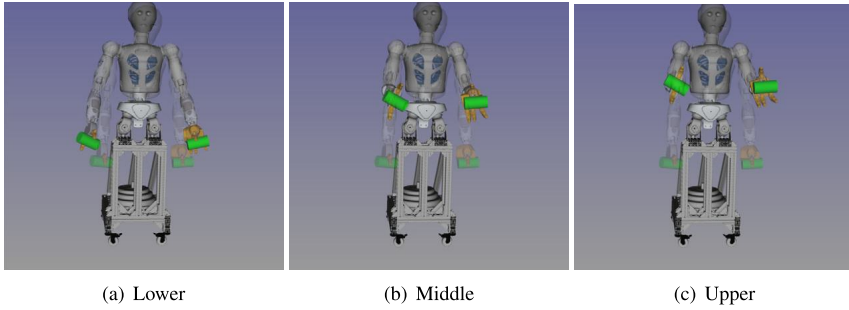


Figure 10.4 Screenshots of the RH5 Manus robot lifting 50 kg weight, with serial abstraction model for lower, middle, and upper limits.

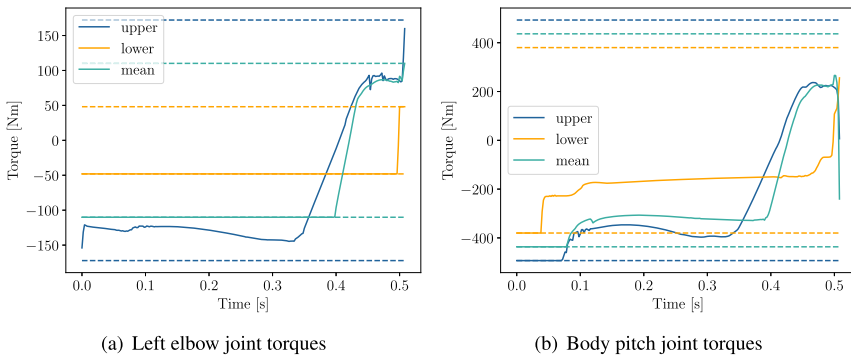


Figure 10.5 Independent joint torques obtained for lifting 50 kg weight using the tree-type model.

observed in actuation space of the closed-loop mechanism. The torque results from trajectory optimization are mapped in actuation space using the inverse statics mapping of the mechanism. The linear actuator forces required to perform the obtained trajectories from optimization are plotted in Fig. 10.7(a). It can be observed that the mapped actuation forces are outside the real limits for both linear actuators in both elbow and torso closed-loop mechanism. Therefore, trajectory optimization while considering the serial abstraction model of the robot failed to perform for the required task.

10.4.1.2 Full hybrid model

In the full hybrid model of the robot, full actuation capabilities of the robot are exploited by planning directly in the actuation space of the robot. OCP formulation remains same for trajectory optimization as discussed before.

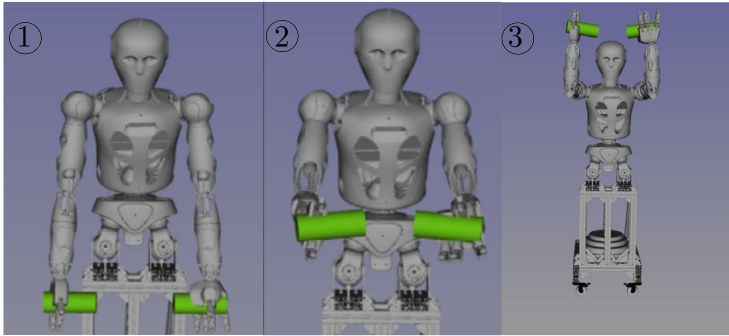


Figure 10.6 Screenshots of the robot lifting 50 kg weight with full hybrid model.

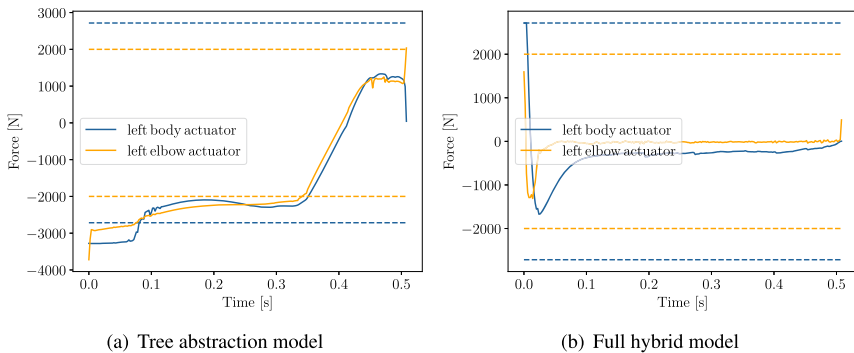


Figure 10.7 Actuation forces for elbow and torso closed-loop mechanisms obtained from trajectory optimization to lift 50 kg weight.

The results are shown in Fig. 10.6, where the robot can be seen to reach the desired target. The obtained motion also mimics how a human would lift the weight.

The torque plots in the actuation space can be seen in Fig. 10.7(b). For both linear actuators in elbow and torso closed-loop mechanisms, the forces are well under the actuator limits. The trajectory optimization is able to find feasible trajectories to lift 50 kg weight in full hybrid model as compared to serial abstraction model. Hence, considering full hybrid model provides an edge over serial abstraction model in trajectory optimization, as more admissible configuration space is used and interesting trajectories are found.

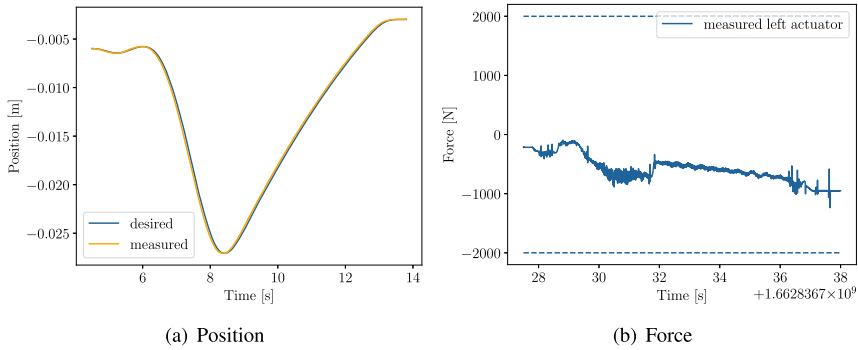


Figure 10.8 Left elbow position and effort for lifting 20 kg weight.

10.4.2 Experimental results

In addition to trajectory optimization results, the trajectories were also performed on the real robot (see Fig. 10.1). The OCP formulation is to lift 20 kg weight (10 kg in each arm), while considering the full hybrid model. Less weight was chosen for hardware safety reasons. To perform the trajectory on the real system, independent joint position trajectories are chosen from the trajectory optimization results and mapped to the actuator space using HyRoDyn [28]. The position and effort plots for the left elbow actuator are shown in Fig. 10.8. The plots highlight the fact that the reference trajectories obtained from trajectory optimization, while considering full hybrid robot, can also be performed on the real robot with good position tracking and effort respecting the limits.

10.4.3 Computational timings

The average CPU time was measured for solving the OCP for 1 iteration 1000 times on a standard laptop with an Intel(R) Core(TM) i7-10750H @ 2.60 GHz processor. These were 31.07 ms and 412.27 ms respectively for serial and hybrid models, which shows a large difference when the closed loops constraints are included in the resolution of an OCP.

10.5 Discussion and conclusion

In this work, we generated a dynamic weight-lifting motion using the design specifications of the RH5 Manus robot model. The RH5 Manus, with its series-parallel hybrid design, provides high stiffness and large pay-

load lifting capabilities. However, in order to exploit these capabilities, it is important to take into account the full robot model, as demonstrated in the paper. For this purpose, we included the loop-closures constraints in the trajectory optimization process. This allowed us to overcome the limitations of the tree-like abstraction model currently used in the state of the art to model and control systems involving PKMs. Through this process, we were able to achieve impressive weight lifting movements in simulation (up to 50 kg) and on the real system (up to 20 kg). The comparison results using the serial model shows its limitations with respect to the full model. This feature was highlighted in the dead lifting motion of 50 kg, where neither of the tree-like abstraction models could successfully achieve an optimal trajectory to reach the target configuration, while lifting the loads while the full model succeeded in this task. This work allows us to demonstrate a well-known theoretical property, that exploiting the kinematics and dynamics constraints in the trajectory optimization process leads to better trajectories for the series-parallel hybrid robots.

The closed loops were modeled using implicit constraints at the acceleration level, which are susceptible of numerical inaccuracies. In order to ensure that these do not affect the physical robot in a negative way, the resulting trajectories were verified with an explicit solution of the constraints using HyRoDyn software [28] before sending them to the robot. Additionally, it was noted that tuning this kind of constrained OCP requires selecting a suitable proximal parameter value. This parameter is variable from task to task for the same robot model. The adjustment of all hyper-parameters in the constrained OCP is more sensitive than a classical OCP and can be time consuming. The computational time of solving a constrained OCP also increases significantly, making online stabilization, including loop-closure, is currently impossible. A bi-level optimization could be a solution to achieve online trajectory optimization while respecting all the capabilities of the robot. This approach should involve different time horizons for each optimization level, separating the resolution of the loop-closures from the minimization of the cost model. To avoid such numerical and computational efficiency issues, explicit formulation of loop-closures could be implemented in the OCP. Furthermore, we would also like to extend the HyRoDyn software to include trajectory optimization in order to resolve the loop-closures constraints in an explicit form and avoid any numerical restrictions.

References

- [1] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the RECUPERA exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019), <https://doi.org/10.3390/app9040626>, <https://www.mdpi.com/2076-3417/9/4/626>.
- [2] D. Kuehn, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, System design and testing of the hominid robot Charlie, *Journal of Field Robotics* 34 (4) (2017) 666–703, <https://doi.org/10.1002/rob.21662>, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21662>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21662>.
- [3] N.A. Radford, P. Strawser, K. Hambuchen, J. Mehling, et al., Valkyrie: NASA's first bipedal humanoid robot, *Journal of Field Robotics* 32 (3) (2015) 397–419, <https://doi.org/10.1002/rob.21560>.
- [4] J. Esser, S. Kumar, H. Peters, V. Bargsten, J.d.G. Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid robot, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), 2021, pp. 400–407, <https://doi.org/10.1109/HUMANOIDS47582.2021.9555770>.
- [5] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Mechatronics* 68 (2020) 102367.
- [6] Y. Tassa, T. Erez, E. Todorov, Synthesis and stabilization of complex behaviors through online trajectory optimization, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 4906–4913.
- [7] K. Mombaur, Using optimization to create self-stable human-like running, *Robotica* 27 (3) (2009) 321–330, <https://doi.org/10.1017/S0263574708004724>.
- [8] I. Mordatch, E. Todorov, Z. Popović, Discovery of complex behaviors through contact-invariant optimization, *ACM Transactions on Graphics* 31 (4) (jul 2012), <https://doi.org/10.1145/2185520.2185539>.
- [9] W. Xi, C.D. Remy, Optimal gaits and motions for legged robots, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 3259–3265, <https://doi.org/10.1109/IROS.2014.6943015>.
- [10] H. Guo, Y. Liu, G. Liu, H. Li, Cascade control of a hydraulically driven 6-DOF parallel robot manipulator based on a sliding mode, *Control Engineering Practice* 16 (9) (2008) 1055–1068, <https://doi.org/10.1016/j.conengprac.2007.11.005>, <https://www.sciencedirect.com/science/article/pii/S0967066107002055>.
- [11] D. Pršić, N. Nedić, V. Stojanović, A nature inspired optimal control of pneumatic-driven parallel robot platform, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 231 (1) (2017) 59–71, <https://doi.org/10.1177/0954406216662367>.
- [12] N. Nedic, V. Stojanovic, V. Djordjevic, Optimal control of hydraulically driven parallel robot platform based on firefly algorithm, *Nonlinear Dynamics* 82 (3) (2015) 1457–1473.
- [13] M. Posa, S. Kuindersma, R. Tedrake, Optimization and stabilization of trajectories for constrained dynamical systems, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 1366–1373.
- [14] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, S. Vijayakumar, N. Mansard, Crocodyl: an efficient and versatile framework for multi-contact optimal control, in: ICRA, 2020.
- [15] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 8540–8546, <https://doi.org/10.1109/ICRA46639.2022.9811843>.

- [16] J. Carpentier, R. Budhiraja, N. Mansard, Proximal and sparse resolution of constrained dynamic equations, in: *Robotics: Science and Systems 2021*, Austin / Virtual, United States, 2021, <https://hal.inria.fr/hal-03271811>.
- [17] R. Featherstone, *Rigid Body Dynamics Algorithms*, Springer, 2014.
- [18] J. Carpentier, F. Valenza, N. Mansard, et al., Pinocchio: fast forward and inverse dynamics for poly-articulated systems, <https://stack-of-tasks.github.io/pinocchio>, 2015–2021.
- [19] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiroux, O. Stasse, N. Mansard, The Pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives, in: *SII*, IEEE, 2019.
- [20] N. Mansard, Feasibility-prone differential dynamic programming is DDP a multiple shooting algorithm?, Online, available: <https://github.com/loco-3d/crocoddyl/tree/master/doc/reports/fddp>, 2020.
- [21] R. Budhiraja, J. Carpentier, C. Mastalli, N. Mansard, Differential dynamic programming for multi-phase rigid contact dynamics, in: *Humanoids*, IEEE, 2018.
- [22] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Sola, N. Mansard, S. Vijayakumar, A direct-indirect hybridization approach to control-limited DDP, arXiv: 2010.00411, 2021.
- [23] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: J. Lenarcic, V. Parenti-Castelli (Eds.), *Advances in Robot Kinematics 2018*, Springer International Publishing, Cham, 2019, pp. 431–439.
- [24] C. Stoeffler, A. del Rio Fernandez, H. Peters, M. Schilling, S. Kumar, Kinematic analysis of a novel humanoid wrist parallel mechanism, in: O. Altuzarra, A. Kecskeméthy (Eds.), *Advances in Robot Kinematics 2022*, Springer International Publishing, Cham, 2022, pp. 348–355.
- [25] Y. Tassa, N. Mansard, E. Todorov, Control-limited differential dynamic programming, in: *ICRA*, IEEE, 2014, pp. 1168–1175.
- [26] D. Mronga, S. Kumar, F. Kirchner, Whole-body control of series-parallel hybrid robots, in: *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 228–234, <https://doi.org/10.1109/ICRA46639.2022.9811616>.
- [27] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
- [28] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, *Journal of Mechanisms and Robotics* 12 (2) (2020).

This page intentionally left blank

PART 4

Case studies on mechatronic system design

This page intentionally left blank

CHAPTER 11

Charlie, a hominidae walking robot

**Daniel Kühn, Alexander Dettmann, Moritz Schilling, Tobias Stark,
and Sankaranarayanan Natarajan**

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH),
Bremen, Germany

11.1 Introduction

The following chapter is divided into three parts. First, a motivation towards the development of a robotic system is given that is able to use different postures. Then, the biological inspiration for the robot is introduced. The chapter closes with the description of one selected application scenario.

11.1.1 Motivation

The idea to design and built tools or machines able to reduce the human workload or to increase the efficiency at constant effort goes back thousands of years. About 70 years ago, stationary industrial robots started working in factories. Nowadays, a wide range of possible areas of application for mobile robotic systems exists, like search and rescue, human assistance, security, or (planetary) exploration. Especially in hostile environments, the use of robots is an excellent alternative to endangering human lives. However, these hostile environments do not offer a homogeneous environment, but instead present the robots with challenges and confront them with a multitude of different environments as well as with increasingly challenging tasks. Generally, legged robots provide the ability to deal with rugged terrain including slopes, overcome obstacles, or deal with fine-grained surfaces, because they are capable of applying forces in noncontinuous ways to the environment and only need small areas of ground for their feet.

In most robots, the body is one rigid unit with no degrees of freedom, mostly to reduce design and control complexity. To gain a higher mobility, it seems necessary to revise the body design, especially since nature shows a different approach. The spine is a central element in vertebrates and is included in basically all movements the vertebrate performs. In robotics, however, the implementation of an artificial spine is ongoing research.

The second key challenge we want to address is the locomotion modes. A robot that can use different types of locomotion has a higher probability of success when it comes to moving in initially unknown terrain. In addition, the closer the robot's outer appearance resembles the human form, the more likely it can handle and move in spaces designed for humans.

As a solution, a hominid system is presented that, inspired by monkeys, is able to perform a stable and robust quadrupedal gait in rough terrain, but also has the ability to stand up and change to a bipedal posture when needed. This posture change expands the robot's field of view and improves its localization and navigation capabilities. In addition, switching to a bipedal stance opens up the possibility of using the front limbs for manipulation tasks, as they are no longer involved in locomotion.

The now extended kinematic complexity and the additionally introduced degrees of freedom must be controllable in some form. This results in an increased sensor density. In order to be able to use the robot in dynamic environments, it needs a fast reaction time to external influences. Brooks stated that "a robot should be able to react to its environment within a human-like time-frame" [2]. Due to latencies and necessary cabling, it does not seem reasonable to control everything from one central unit. In the hominid robot Charlie, a decentralized approach was therefore chosen. A further advantage of local data processing is that within a sub-structure, e.g., a leg, the actuators and sensor system could be combined to a logical unit. This unit can react to external stimuli all by themselves with minor latency.

11.1.2 Biological inspiration

Nature found a way to populate nearly all natural habitats on earth, despite the partially extreme living conditions. If one abstracts the desired application scenario, good examples can be found in nature of what a robotic system should look like in order to perform the desired tasks.

Multi-talented animals like monkeys are able to move with different locomotion, such as bipedal, tripedal and quadrupedal. Of course, there is always one favored form of locomotion, but especially primates realize a high mobility by adapting the locomotion to the current situation. They also possess advanced climbing and manipulation abilities [3] and are known to use tools to gain access to food. It was these kinematic skills that led us to use chimpanzees as the archetype for the design of a multi-talented robot.

Numerous research groups successfully developed biologically inspired robots that mimic the locomotion of their natural counterparts. Still, the

majority of robotic systems focus on one type of locomotion. Multi-legged animals have the intrinsic ability to generate a stable stance. Robotic abstractions mostly use single-point-contact feet (SPCF) (like implemented at LittleDog [4], Starleth [5], or the Scorpion robot [6]), where a small endpoint on each leg is used to interact with the environment. This type of feet has the advantages of being both mechanically and computationally non-complex. As for the Charlie robot, this kind of feet were not an option. Charlie should be able to stand on its rear feet and walk bipedal, so an implementation of a more human-like foot, or a multi-point-contact foot is essential. For a transition from a four-legged to a two-legged posture, multi-point-contact feet are also advantageous, since the center of mass can be more easily kept in the support polygon.

Regarding the change of posture, it becomes essential to increase the robots general mobility. The spine is a central element in the vertebrate's body, used to enhance the overall locomotion among other things. To increase Charlie's mobility, the rigid connection between the front and rear body is replaced by an actuated artificial spinal column.

11.1.3 Application scenarios

Charlie was built as a demonstration platform to test and validate new mobility concepts for future planetary exploration robots. In this application, the ability to traverse inclined and rough terrain, as well as sandy surfaces is mandatory to reach places of high ecological and scientific interest, like craters, valleys, or lava tubes. One target for future exploration mission is the Valles Marineris on Mars. A up to 7 km deep trench of this canyon system is particularly exciting for science. Due to indications of water resources, former volcanic activity and the shading of UV radiation, it fulfills the prerequisite for the existence of extraterrestrial life. However, mountains, gorges and caves form an extremely complex terrain which is difficult to access [7].

The idea is to send a swarm of heterogeneous robots to successfully explore such a demanding environment, i.e., aircrafts for wide-range coarse exploration, rovers for energy-efficient mobility, and walking robots to move around within the rugged rock formations and navigate in caves and crevices. Thus, a walking system like Charlie has high requirements on navigation and control in order to explore such a demanding environment. Besides high locomotion capabilities, every team member has to build up its own map for cost-effective and secure navigation. In addition, it is beneficial to exchange and fuse the generated data as part of the network

intelligence within the swarm to create a decentralized visual and geometric map that mission control and every swarm participant can use. This allows a global multi-robot exploration, taking the specific characteristics of each swarm participant into consideration, allowing an efficient exploration mission. To maximize the cost-effectiveness of each robotic agent, it is important to keep the weight of the robot platform low for high agility and low transport costs. Visual navigation is a very suitable technology that builds on lightweight, passive sensors and due to the large redundancy enables a reliable position estimation. In contrast to radio-based positioning, a permanent connection to other swarm participants is not necessary. However, the position estimation based on a continuous visual odometry using a stereo camera is not sufficient. Especially in areas with low brightness, long exposure times or resource-intensive lighting would be needed. Therefore, a cost-effective navigation needs to be realized by (i) using proprioceptive data (already used to control the robot motion) to record the body position and movement and convert it into coarse position information, and (ii) using visual data from a lightweight active light-emitting camera.

The planetary space exploration application also share many requirements with terrestrial applications such as search and rescue or last mile delivery. Thus, the presented concept for Charlie also as high transfer potential to many other use cases where high mobility and autonomy are required.

11.2 Mechatronic system design

In the following, first the mechanical design of the Charlie robot is described. This includes the leg, foot, and spine design which include parallel mechanisms leading to a series-parallel hybrid mechanism (see [30] for a survey). Then, the electronics are presented, followed by the software design.

11.2.1 Mechanical design

The Charlie hominid system draws heavily from its biological models, the bonobos and the chimpanzees. These animals are well studied and documented in the literature, e.g., [1] or [8]. The design of the robot is strongly based on these biological models. For example, the range of motion of the animals' joints was taken as a basis and an attempt was made to implement this in the robotic system as well, and the standardized lengths of the front and hind legs and the torso can also be seen in the robot.

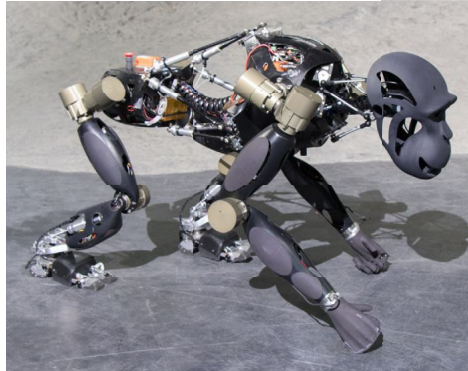


Figure 11.1 The Charlie robot in a quadrupedal pose.

The robot's height from shoulder to ground is 750 mm in a quadrupedal posture and 1300 mm in a bipedal pose, measured from head to ground. Charlie has currently 37 active Degree of Freedoms (DoFs). The actual weight of a fully equipped Charlie robot is about 23 kg. The robot can either be powered externally or internally via rechargeable batteries with a nominal voltage of 44.4 V.

Unlike many robots, Charlie has no rigid connection between hip and shoulder. However, this means that this popular area for accommodating electronics is eliminated. As a result, Charlie's computing and power units are located in the shoulder and hip structures.

11.2.1.1 Limb design

The actual limb design can be seen in Fig. 11.1. In the following, the structure is described more in detail.

Femur

Based on a tubular structure as the ideal torsionally stiff element, a structurally optimized thigh design was developed that can be used as the sole structural component and at the same time allows the integration of the assemblies to be accommodated in the thigh. In addition, the appearance of the load-bearing structure was designed in such a way that no further cladding of the femur is necessary. To design the femur structure, a hollow structure was first created between the knee and hip flanges in such a way that its rough shape resembled the natural model and provided enough space for the components to be integrated. The recesses required for assem-

bly were then provided, and the two assemblies for the toe drive and motor electronics were added inside the structure. Channels for cabling were also integrated.

Using FEM analysis, the structure of the upper leg was optimized in an iterative process with regard to high stiffness and low weight. The result of the simulation is a residual lateral displacement of the foot joint with respect to the robot torso of ± 5.2 mm. A large part of this displacement results from the elastic deformation of the lower leg structure. The optimized thigh structure weighs about 300 g.

Lower leg

Based on the femur optimization, it was decided to perform a similar structural optimization process for the lower leg. The two linear actuators for the drive of the foot joint are considered with corresponding movement space Fig. 11.2a as well as integration space for the differential of the toe drive and the motor and voltage converter board. The force-torque sensor is providing the mechanical interface to the foot. In the lower leg, a uniform stress distribution is realized. The lateral displacement of the foot joint remaining after the lower leg optimization in the simulated loading case is still ± 3 mm. The weight of the new lower leg structure is about 215 g.

Foot

The endpoint of the rear legs are the feet. To minimize the moments of inertia of the legs, the feet have to be as lightweight as possible and yet to be able to bear the body weight. In general, a foot provides different functionalities besides generating and maintaining traction. A reliable grip on all different types of soils and on inclinations is desired, therefore a foot should be able to adapt itself to the given context. In humans and apes, this adaption is realized by allowing the foot to deform up to a certain level and the DoFs introduced by the ankle joint. In literature, the design of human foot is documented in detail and is used, when limited data is available on chimpanzees feet [9], [10].

The human or chimpanzee foot has three DoFs. According to [11], the Range of Motion (RoM) of a human foot is between 10° to -20° eversion/inversion, hereafter referred to as roll motion, and between 20° to -30° dorsiflexion / plantarflexion, hereafter referred to as pitch motion. An abduction/adduction rotation, hereafter referred to as yaw motion, of 10° to -10° can be observed. This movement, however, is gained by the

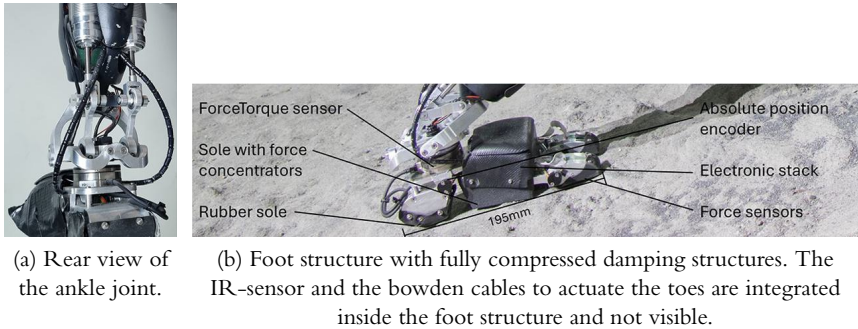


Figure 11.2 Foot design in Charlie.

structure of the leg. For a technical abstraction, mainly the first two DoFs are necessary to drive an artificial foot, whereas the yaw motion can be introduced by the leg DoFs. The RoM of ankle joints in apes is comparable to humans, yet allows a slightly larger RoM [12]. During locomotion, lateral movements of the toes are rather minimal. The main active degree of freedom from the toe is divided in flexion (30° to 40°) and extension (50° to 60°) [9]. However, the toe has an important role to play, since they increase grip or get hold while walking or climbing in unstructured environments. The principle behind the toes is also realized within the Charlie foot.

The technical implementation of the rear foot is shown in Fig. 11.2b. The multi-point contact foot consists of five rigid bodies, including two toes. Active flexion of the toes of up to 75° are realized to increase grip or get hold while walking in inclines, whereas an active extension is not realized. The angle of each connecting point of the rigid bodies can passively change during locomotion and is therefore monitored by absolute angular encoders. The core structure of the mechanical foot is rigid and covered by different materials. The outermost layer is the sole, adding traction and minor damping capabilities to the foot. In addition, it provides mechanical protection to the sensitive inner parts as well. As mentioned, the overall design of the robot is biologically inspired, the same is true for the dimensions of the foot. Each foot is 195 mm long and 50 mm width at the heel and 80 mm at the toes, and has a height of 80 mm. The weight of the foot is 350 g including its sensor processing electronics.

In apes, the anatomy of their front and rear limb's end-effector is quite different. Primates show a quadrupedal walking behavior called knuckle

walking. In this gait, the rear feet are used normally and the sole establishes the ground contact, but the fingertips of the front limbs are flexed and the outer part of the middle segment of their fingers establish a ground contact.

11.2.1.2 Spine

The anatomy of a spinal column strongly depends on the occurring load cases, whereas the primary load case is the type of locomotion. Humans favor bipedal locomotion and for the spine the vertebral bodies are getting smaller from the pelvis to the head. This is because structural load introduced by the weight of the body parts becomes lesser the higher one goes [13]. In chimpanzees, the primary gait is quadrupedal, therefore the appearance of the spine differs in form and size compared to a human spine. In this case, the basic structure of the spine is curved and the size of the vertebral bodies and intervertebral discs throughout the spine are almost identical due to the different load cases.

Living beings with a spine use it extensively. As a human being, one only really become aware of the frequent use of the freedom offered by your back and spine when the movement becomes limited, for example due to back pain. A spine can usually be divided into three sections: the cervical, the thoracic, and the lumbar spine [14]. The thoracic and lumbar section are often merged and called thoracolumbar section. The thoracolumbar spine was identified as the most interesting part for a technical abstraction and implementation in a robotic system.

However, one can not just have a look at the spine itself, but has to look into the overall mechanism. By considering the spinal column as an independent unit, one disregards the underlying complexity of this biomechanical system [14]. The spine itself connected to the skeletal structures nearby via muscles, ligaments, and tendons.

Therefore, a serial design for the actuation of an artificial spine was not chosen, since a parallel kinematic mechanism seems to be corresponding better to the natural counterpart. The often mentioned drawback of parallel kinematics is that they have a more limited workspace compared to serial kinematics. By comparing the spine to a leg or an arm, which have these serial alignments of their degrees of freedom, one can see that the limitation regarding the workspace is also true for a natural spine. The advantage of a parallel alignment is the higher stiffness and it can provide higher torque than a serial kinematics of comparable size and weight.

The design of Charlie's spine follows the principle of a Stewart platform [15]. These kind of platforms can provide high stiffness despite a

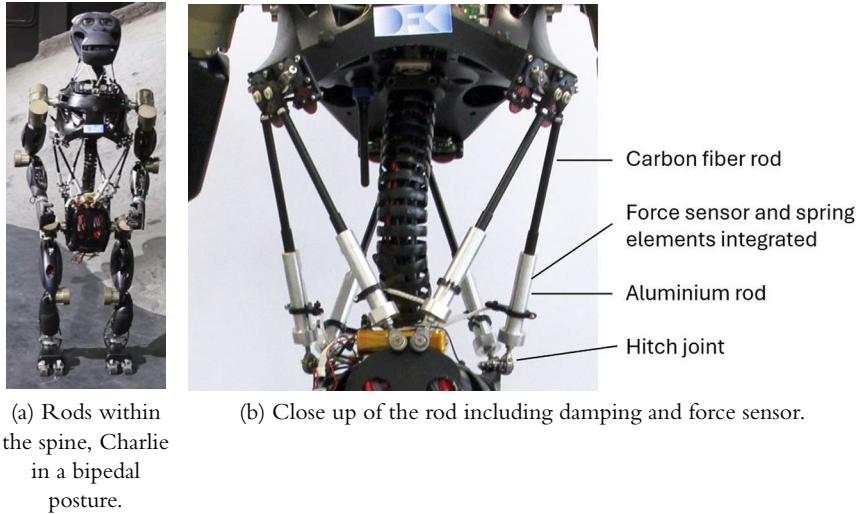


Figure 11.3 Artificial spine in Charlie.

light-weight design; both are excellent properties for mobile robots. The overall weight of the artificial spine is 3.3 kg, including all motors, levers, rods, casing, and electronics. Transmitting data between brain and body is one of the main functions of the nerves embedded in the spine. In Charlie's artificial spinal column, a cable duct is installed as well in the middle of the spine. The length of the rods in Charlie's spine is not changed directly, but by rotational joints, which are connected with levers to the rods. An advantage of this design is that the motors that drive the rods could be placed within the front body. The rods depicted in Fig. 11.3c are designed such that an additional spring element as well as a force sensor are included in the rods. In both systems, the motion is not generated by the spinal column directly, but by muscles or motors on the outside of the system.

11.2.2 Electronics design

Charlie is powered by a single voltage source of 44.4 V, which can be either a Li-Po battery pack or an external source. This voltage source directly powers the central electronics as well as a connected relay board. Only if the boot process of the central electronics board has been successful, the legs, head, and spine subsystems get powered up by switching on the designated relay on the relay board.

On the central electronics (see Section 11.2.2.3) the higher level motion control is executed. From there, several communication links emerge into each of the four legs, the head, the spine and the power management system. The communication links form a tree of interconnected devices reaching to the devices at the end-effectors of the system.

Each joint of the legs is equipped with a stack of printed circuit boards (PCBs): one PCB hosts a Spartan 6 FPGA, one the three half-bridges for actuator commutation, and the last has several components for handling communication and external sensors. The actuator consists of a RoboDrive rotor and stator pair, a Harmonic Drive gear, three Hall sensors, and an optical encoder for accurate measuring of the rotor position and an iC-Haus position sensor on the output shaft to measure the absolute position.

At the end-effectors of the system are one (front legs) or two (rear legs) Multiplexed Data Acquisition Version 2 (*MDAQ2*) boards. They consist of an STM32 microcontroller, a 16 Bit ADC connected to an programmable gain instrumentational amplifier, whose inputs are switchable via analog multiplexers with eight channels each. These boards read out the force-torque sensors at the ankle joints (all legs) or acquire and process the state of the complex feet (for more details on the feet, see Section 11.2.1.1). Of these sensors, the force sensitive sensor array is the most complex one: 7 power rails drive 7 force sensitive resistors forming a 7×7 grid. Measuring all the voltages in the grid while driving one of the power rails at a time enables the on-board firmware to calculate the individual forces across the foot contact surface. With these forces the center of pressure as well as the support polygon of each foot could be obtained.

11.2.2.1 Stereo cameras

Two webcams have been integrated into Charlie's head. The cameras are connected to the Charlie PC in the shoulder area. A transmission of the camera image to the operator PC is realized either via LAN or WiFi.

11.2.2.2 Time of flight camera

A Time of Flight camera is installed in Charlie's mouth to generate a depth image from which a point cloud can be generated. The Camboard pico flexx by pmdtec is integrated, which has a comparably small opening angle of $62^\circ \times 45^\circ$ with a resolution of 224×171 pixel, but it weighs only 8 g and can directly measure distances up to 4 m without cost-intensive post-

processing. In addition, this active, infrared-emitting sensor is especially helpful in very dark environments.

The camera is positioned in a way that in the most upward looking head joint configuration, the upper limit of the field of view is horizontally aligned. This way, Charlie can see far enough in quadrupedal pose, but also can pitch its head downwards in bipedal pose to fully observe the scene. To create a map, the observed depth image is converted into a point cloud, which must then be transformed into the robot base frame via the head and neck kinematics.

11.2.2.3 Central electronics

Due to the use of imaging sensors, it was quickly determined that the computing power in Charlie needed to be increased. Based on weight and dimension, the selection fell on a Com Express CPU module of type 10 with an Intel Atom quad-core processor with 1.91 GHz per core. This CPU module requires a so-called carrier board, which provides various connections to connect existing or newly added peripherals in Charlie. The connections include Ethernet plugs as well as the possibility to connect hard disks, SD cards or USB devices. Connect Tech's COM Express Type 10 Mini Carrier Board was chosen as the carrier board. It is an extremely small carrier board which features latching and locking connectors. The carrier board is therefore ideal for applications with limited space as well as rough terrain and due to its design supports temperatures from -40° Celsius to 85° Celsius. Particularly within the robotic system, heat build-up can occur due to the lack of active fans. In terms of temperature, a high load capacity is therefore desirable.

Charlie's control software allows the existing relays to be switched accordingly to supply the PC with power. If the PC is not needed, it can be deactivated, thus reducing the system's power consumption and contributing to a longer system runtime.

11.2.3 Decentralized control architecture

Charlie's control architecture follows a distributed control approach, as depicted in Fig. 11.4. The high-level control is implemented in ROCK¹ and deployed on the central electronics. There are several ROCK tasks involved. On the one side, there are drivers for the IMU and exteroceptive sensors,

¹ <https://www.rock-robotics.org/>.

i.e., stereo camera system and ToF-camera that are connected via USB. On the other side, ROCK tasks for guidance, navigation and control are used that are further described in Section 11.3. The joint commands and sensor readings are communicated via Ethernet to/from the low-level control board.

The low-level control board (ZynqBrain [16]) manages the communication to all microcontroller- and FPGA-based processing nodes within Charlie's distributed control system. NDLCOM [17] is used as protocol with an LVDS-based daisy-chained communication. The NDLCOM communication includes nodes for all joints as well as sensor nodes at wrists, ankles, spine and feet.

Additional house keeping information from the battery pack and the possibility to power entire extremities are handled through the power management board and the LiPo watcher. The low-level boards act as a relay to the high-level CPU. The communication to the external control station is handled via WiFi.

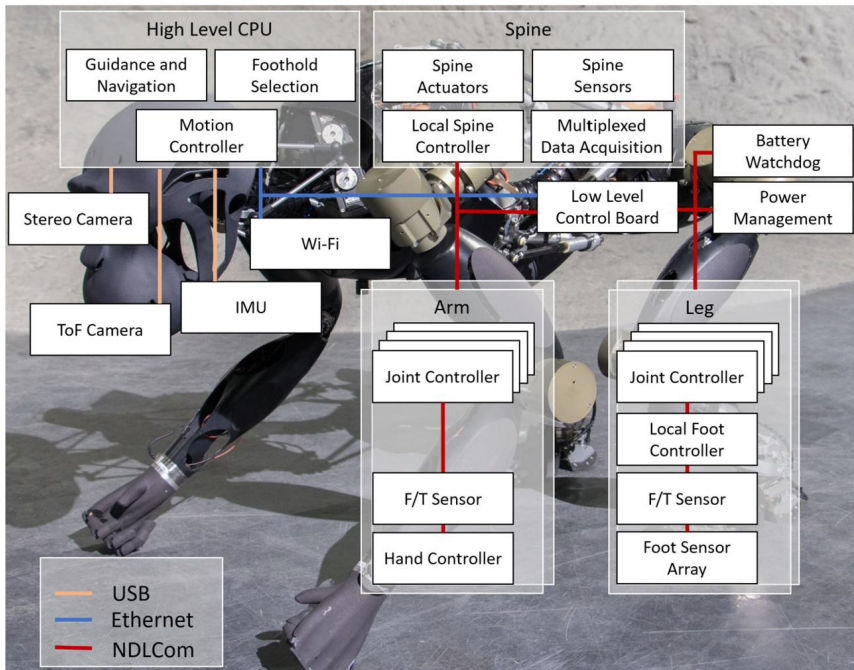


Figure 11.4 Overview over Charlie's sensor and processing nodes.

11.3 Modeling and control

In the following, first the kinematical models of the subsystems of Charlie are introduced. Afterwards the different levels of controlling these subsystems from the individual joints up until the high-level layers like navigation are described in detail.

11.3.1 Kinematic modeling

The kinematic model for Charlie in general consists of a front and a rear body connected by the 6 DoF spine. On each of the bodies there are two legs, which themselves are serial robotic systems. However, the spine joint and the ankle joint are parallel substructures and have to be modeled separately. The ankle joint is implemented using a parallel mechanism of type $2SU[RSPU]+1U$, which is a more complex variant of the $2SPRR+1U$ mechanism discussed in Chapter 4 (also see [31]). Its geometric modeling has been discussed in [18]. The spine joint is a hexapodal structure whose rod lengths are not changed directly by prismatic joints as in case of classical Stewart–Gough platforms of type $6-UPS$, but by revolute joints changing a lever mechanism. It can be modeled as an $6-RUS$ type parallel mechanism and the geometric solutions are available in [19].

11.3.2 Low-level control

In the following the architecture of the low-level control of Charlie is described in detail. From central control, joint commands are periodically transmitted with 100 Hz to all joint nodes of the system and in turn the joint and sensor nodes send their telemetry values like position, speed or force/torque back to the central control.

11.3.2.1 Joint control

The joints are independent units taking the sent commands and controlling their joint torques, angular velocities and angles accordingly. For this purpose a cascaded controller is used, which works at high frequencies of up to 32 kHz. Besides the normal revolute joints in the legs and arms, the ankle and spine joints, which consist of more than one motor, are handled differently and are described in the following.

Ankle joint controller

The ankle joint controller is a special controller because it has to handle the parallel substructure described in Section 11.3.1. The geometric model

of the ankle joint is computed directly within the ankle joint node, so that the high-level control can directly control the pitch and roll angles and velocities of the ankle and not the position of the motors. As a result, no special task in the high-level control is needed, because these to angles behave like normal revolute joints.

Spine controller

Another special controller within Charlie's control architecture is the spine controller. Due to the hexapodal structure, the controller has to calculate the inverse geometric model of the spine (for more details see Section 11.3.1) to get the six angles of the spine motors from the given translation and rotation of the rear body with respect to the front body. Therefore, the desired translations and rotations are indirectly controlled by controlling the six spine motor angles.

11.3.2.2 Local foot controller

Two virtual torsional spring-damper systems at the ankle joints were implemented with the local foot controller to be able to adapt the feet to uneven surfaces without the need of high-level control inputs. One of the two *MDAQ2* boards at the feet (see Fig. 11.2) periodically captures the current force and torque values at the ankle joints. Depending on these torque readings, the ankle joint pitch and roll angles are modified with pitch and roll angle offsets. These offsets are sent to the ankle joint controller directly and hence form a local control loop. To adjust the controller outputs, the spring constant as well as the inertia can be set for each spring individually. The damping constant is derived from these parameters to ensure an always critically damped system behavior. Additionally, the controller outputs can be clamped to a symmetric range.

11.3.2.3 Spine sensor processing

A special feature of Charlie's spine structure is that it can be used as a force-torque sensor between front and rear body. As described in Section 11.2.1.2, each of the rods of the spine includes a 1-DoF force sensor and with the information of the positions of the spine actuators from the spine controller, the sensor board (another *MDAQ2*) could calculate the resulting forces and torques that occur between front and rear body.

11.3.3 Motion control

In the following, the motion control is described that generates joint commands. It is based on a generic modular Central Pattern Generator (CPG) approach that can be utilized for different walking robots. In addition, the two Charlie-specific addons are presented that utilize Charlie's unique spine and foot design to improve the locomotion.

11.3.3.1 Modular central pattern generator

Charlie's motion control is based on a CPG approach that is implemented within a behavior-based control approach, i.e., several behavior modules simultaneously process input data and provide output data to other modules, thus, contributing to the overall emergent robot behavior. The CPG generates a clock, which sets the pace and triggers consecutive leg movements, by activating several trajectory generators for the feet and body motions. The resulting open-loop Cartesian foot trajectories are then superposed by reactive modules to adapt to the surface. Finally, inverse kinematics are used to generate the target values for the position-controlled joints. The result is a generic approach that generates walking trajectories regardless of the exact morphology of the target robot.

Therefore, two basic guidelines are followed. First, the overall control is split into general-applicable behavior modules, which are required for every system, and robot-specific behavior modules, which extend the general architecture and exploit the capability of the target system. Second, behavior modules are grouped to larger behavior modules to implement a hierarchical structure that efficiently reuses functionalities.

Every behavior module provides a specific functionality contributing to the overall robot behavior. On the top layer, in- and outputs of the overall control as well as the main behavior modules are defined, e.g., the *CPG* behavior module to derive gait-dependent step cycles for the limbs and for the body, which supports a crawl gait and a trot. A *posture controller* enables the configuration of different postures. Besides four-legged postures, a two-legged posture and the transition between both is supported. Furthermore, several *limb controllers* for every motion-supporting limb, a *trunc controller*, a *head controller*, as well as several data processing nodes are implemented, which provide module-specific information of the current robot state.

The architecture is a mixture of open-loop trajectory-generating modules and trajectory-adapting reactive modules to adapt to the environment. A balance controller is implemented that continuously monitors the Zero

Moment Point (ZMP) and the support polygon. In order to maintain stability, it creates an offset to the planned trunk motion whenever the ZMP approaches the edge of the Support Polygon (SP). This way, Charlie automatically moves towards an incline to increase its stability. In addition, the Elevation and Depression Reflex (EDR) accounts for irregular terrain by implementing a haptic touchdown. During the down phase of the swing, the ground detection is utilized to influence the target position in two ways. First, in case an early touchdown is detected, the foot will maintain in this crouched position to avoid tipping over. Second, when no touchdown is detected at the end of the down phase, a search motion is initiated that stretches the leg until contact is detected. The open loop trajectories in combination with these reactive behaviors eventually generates a stable waling pattern for unstructured and inclined terrain. Further details are provided in [20].

11.3.3.2 Spine support during locomotion

Charlie is a quadruped robot that features an actuated artificial spine. It can support different kinds of motions, including locomotion, by using its rotational and translational range of motion. Therefore, the *spine support* behavior module computes reference values for the spine to positively influence the position and orientation of the front and rear feet by reducing the required range of motion of the rear legs during walking. The spine is translated along the y-direction if both rear legs are laterally shifted to the same side. The spine is rotated around the z-axis to compensate for a difference in x-direction. The spine is rotated around the x-axis to compensate for a difference in z-direction.

The spine support behavior module modulates the default trajectory by utilizing the special kinematical structure of Charlie. The benefit of this module concerning the range of motion and velocities of the first hip and knee joint was investigated, since they contribute most to the forward motion.

Therefore, a walking pattern of moderate speed ($60 \frac{\text{mm}}{\text{s}}$) was created and compared with and without spine support. The analyzes of the required joint angles and velocities during walking reveals that the demands on movement range and velocity are reduced, when walking is supported by the spine.

The yaw rotation of the spine reduces the distance of desired foot position towards the robot's base frame. Thus, especially in the phase around liftoff, the knee is less stretched which has three advantages. First, more

movement range of the leg remains, which is crucial when reactive behaviors want to modulate the desired foot position, e.g., stretching the leg further when the foot steps into a hole. Second, in this posture, the step length in longitudinal direction can be increased from 390 mm to 420 mm when utilizing the spine until a leg loses ground contact before the actual lift phase starts. In general, the workspace of a limb is increased, highly depending on the position of the end effector, by 6 % to 16 %. Third, less rotational velocities are required with activated spine. Thus, less dynamics are inserted, resulting in greater stability or the ability to move faster. Further details and experimental data can be found in [21]. The active spine support was also tested in inclines from -20° to 20° and it was shown that it reduces the maximum joint velocities in all inclines [22]. The overall power consumption, however, remains nearly constant because the reduced effort for the limbs is compensated with the additional power needed to drive the spine motors.

11.3.3.3 Active foot support during locomotion

The design of Charlie's unique rear feet is a complex system consisting of a sole with many pressure sensors, an active roll-pitch ankle joint, a force-torque sensor, and a microcontroller for local control (see Section 11.3.2.2). Due to this decentralized approach, surface irregularities can directly be compensated with a local spring-damper behavior. Through the automated alignment to the surface, the multi-contact foot design provides increased friction. An experimental evaluation [23] showed that the foot can automatically align to the surface during touchdown to establish as many ground contacts as possible. When the foot is aligned to the surface, a ca. 10% higher friction in longitudinal direction can be expected in contrast to a spherical foot design of similar size.

The motion controller can abstract the ankle to a rotational joint and just has to provide reference positions and desired stiffness for the virtual springs in the ankle joint. They are synchronized to the walking gait. During down-phase, the foot is soft in order to adapt to the ground. During progress of the stance-phase, the achieved offset is kept, but the stiffness is increased so that the corresponding leg can support the forward motion.

A gait-dependent foot pitch modulation lifts the heel before the lift phase starts to allow larger steps. During shift phase, the toes can be lifted to reduce the chance of hitting an obstacle unintendedly with the toes first. In order to maintain the position of the ball, the resulting difference of the ankle position is added before calculating the inverse kinematics of

the leg. An experimental evaluation was conducted to measure the effect of the foot pitch modulation. Therefore, a reference locomotion pattern was used and compared to the same pattern but with activated foot pitch module. Without using the foot pitch behavior, a maximum step length of 390 mm can be achieved. Using a 30° liftoff pitch angle increases the maximum step length to 470 mm, which corresponds to a gain of 20%. During walking, the first hip joint and the knee joint contribute most to the forward motion. The analysis the required joint angles and velocities showed that the demands on movement range and velocity are reduced, when walking is supported by the foot pitch behavior.

Especially in the phase around liftoff (from 75% up to 8% of the gait cycle loop), the use of the foot pitch compensates the stretching of the leg, resulting in a more bent knee. This has the major advantage that additional movement range of the leg is gained, which is crucial when reactive behavior is to modulate the desired foot position, e.g., stretching the leg further when the foot steps into a hole. Regarding the required velocities, the active usage of the foot pitch has an higher influence on the knee joint than on the hip joint. The knee joint velocity during lift phase is almost halved. Thus, Charlie is capable of moving faster with activated foot pitch module.

11.3.4 Guidance and navigation

The motion control allows reliable, blind locomotion over unstructured terrain. On top of this layer, we can find the guidance and navigation layer that estimates the pose in the world frame, generates map and uses planning approaches to guide the robot. In addition, this layer can provide additional information that can improve the locomotion capabilities. The following subsections provide an overview on the utilized tools and methodologies applied on Charlie.

11.3.4.1 *Contact-based state estimation*

By measuring the position of the motors, forward kinematics can be used to directly calculate the pose of the foot points. In addition, the force-torque sensors can determine whether the leg is touching the ground and thus contributing to the robot's locomotion. The generated motions of the legs are averaged and rotated into the world coordinate system using attitude determination via the IMU. The resulting total motion is summed, which determines the pose in the world coordinate system. This dead-reckoning approach is a good pose estimation for a short time, but leads

to an accumulating position error, which can be eliminated by following SLAM procedures.

11.3.4.2 SLAM

To generate a consistent map, at first, the input data is processed using the Point-Cloud-Library (PCL) [24]. Nearby scans are aligned using the Generalized Iterative Closest Point (GICP) algorithm to create relative poses between measurements. This variant performs a point-to-plane matching and is suited for sparse point clouds. PCL is also used to down-sample incoming scans and perform a distance-based outlier rejection. The collected scans and the calculated relative poses are then stored in a graph using the Boost-Graph-Library (BGL). This serves two major purposes: on one side, it provides a common interface to access the stored graph to be used by different front-ends and back-ends. On the other side, BGL also supports a number of graph algorithms like shortest-path and breadth-first-search that can be used in the mapping process directly. Additional relative poses are generated from the robot's odometry. This information is added to the graph in the same way as the GICP result, thus adding to the overall information within the system. The third component in the SLAM solution is the optimization back-end. This component solves for errors that are present within the graph due to contradicting odometry and GICP constraints between all the nodes. As most optimizers require their own graph representation as data input, the graph is stored in BGL and translated to the back-end's format prior to the optimization process. The results are then written back to the poses in the Boost-Graph to avoid dependencies on the used optimizer. In the current setup, the g2o [25] framework is used for the global optimization step.

To test the accuracy, several reference trajectories were traversed with Charlie in the Space Exploration Hall² of the DFKI RIC and the tracking error was measured with a motion tracking system of less than 1 cm accuracy. The measured average tracking error was with 161 mm acceptable for navigation purposes. The experiments revealed that the generated point clouds capture only a small part of the environment due to the camera's relatively narrow field of view. This is also enforced due to the requirement of tilting the camera downwards so that the motion planner can evaluate the floor in front of the robot. Thus, the generated point clouds tend to have

² <https://robotik.dfki-bremen.de/en/research/research-facilities-labs/space-exploration-hall/>.

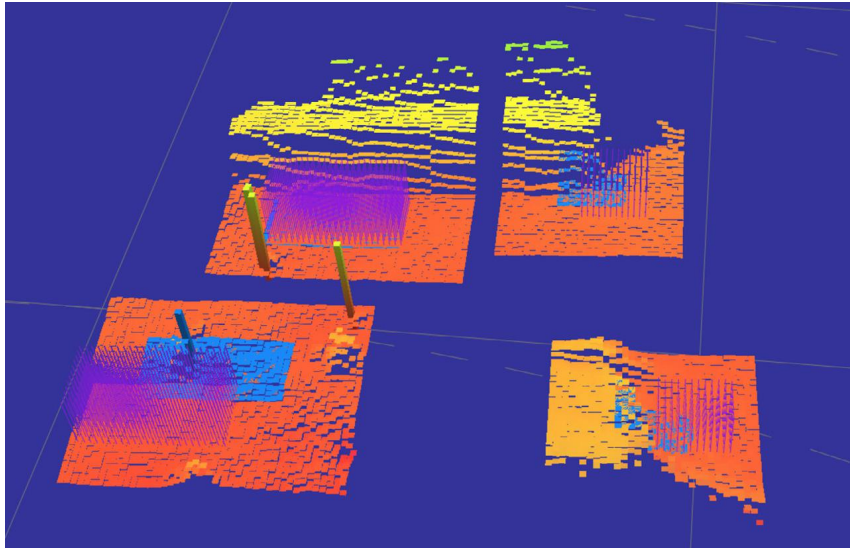
little overlap, resulting in an unreliable position tracking using GICP. As a result, turning motions had to be restricted in rotational speed to remain an overlapping point cloud for the GICP.

11.3.4.3 Path planning

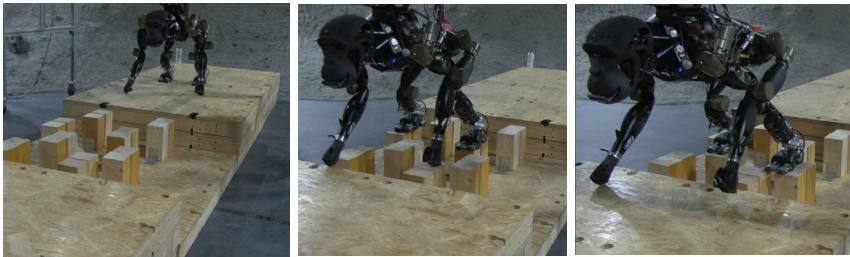
To retrieve a path – or trajectory – that connects the current position with the goal position, the MLS map is transferred into a traversability map. This map uses the mobility characteristics of the robot to represent traversable and non-traversable areas. The actual path planner is based on the Search-Based Planning Library [26]. More specifically, an ARA* planner computes a path in the traversability map by using a grid search and pre-defined robot motion primitives. Thus, the path generated consists of a list of states, each one defined by a position and an orientation. A spline is used to generate a smooth trajectory between them. A trajectory follower is finally computing motion commands, i.e., longitudinal, lateral, and rotational velocities, based on the discrepancy between reference and current pose and sends them to the motion controller.

11.3.4.4 Adaptive footstep planning

The reactive layer of the motion control allows movement over uneven terrain and compensates for instabilities occurring while stepping over small obstacles. Since planning for the legs motion is computationally expensive and takes a long time, an adaptive footstep planning is proposed here, which makes only local adjustments to the target positions of the walking pattern generator. The developed adaptive footstep planning algorithm allows to find an optimal foot contact point on an obstacle or even between different obstacles. Charlie's ToF camera is used to create a Multi Layer Surface (MLS) map to extract the terrain information. A region of interest (ROI) map is created for each leg from the MLS map. The size of the ROI map depends on the foot print size. It must be larger than the respective footprint, but should not be too large to find an unstable posture or to collide with the other legs. In order to stay as close as possible to the target foot step provided by the motion controller, a spiral search pattern is applied to iteratively evaluate potential footholds. Each candidate is evaluated whether the occupied space of a footprint is plain and without inclination. The candidates are rated according to this metric and the best one is chosen. An offset to target to target footstep is calculated and sent to the motion controller (Fig. 11.5a).



(a) Local foot maps with obstacles, original foot contacts (blue), new foot placements (magenta arrows).



(b) Start pose

(c) Stepping field.

(d) Towards target pose

Figure 11.5 Charlie traverses a random stepping field with the adaptive foot-step adaption.

In order to evaluate the adaptive footstep planning algorithm, a random stepping field is created, as shown in Fig. 11.5. The stepping field objects are arranged such that using Charlies walking pattern the robot is not able to cross the stepping field. Here, the footsteps needs to be corrected to successfully cross the stepping field. The walking behavior derived from the motion controller was maintained before and after the stepping field area, but in the stepping field area, the adaptive footstep planner finds an appropriate offset for the motion controller to successfully navigate Charlie over the stepping field without falling (Fig. 11.5). The experiment revealed that a precise map with a consistent and good pose estimation is required.

This demand grows when the exteroceptive sensors are looking to the front and cannot see the surface beneath the robot. Thus, having depth-image proving sensors at every limb would reduce this demand because map with the history of point clouds would not be required.

11.4 Conclusion and outlook

This chapter introduced Charlie, a quadruped that is able to change its posture from a quadrupedal pose to a bipedal pose and vice versa. This capability is mainly enabled through two parallel kinematical structures. Firstly, Charlie features sophisticated ankle for a multi-legged locomotion to demonstrate the advantages of actuated multi-point-contact feet. And secondly, it introduces an actuated artificial spine that replaces the often used rigid connection between the front and rear body to increase the robots mobility. Both unique developments were described in this chapter. It will now conclude with summarizing selected success stories, the lessons learned, and an outlook for future work.

11.4.1 Success stories

Overall, with Charlie unique mechatronic subsystems were created. The robot was developed and used in the iStruct project.³ Particular selling points of the robot are as follows. The development of a complex, actuated artificial spine with 6 degrees of freedom, functionally integrated in a robotic system. Research groups worldwide are trying different approaches to integrate complex artificial spines into robotic systems, but so far, in contrast to iStruct, they have not been shown to work without further external support [27]. With the rear-feet structures, another intelligent subsystem has been developed. With seven active and passive degrees of freedom, the foot can adapt itself to the ground. In order to make this high flexibility controllable, a total of 63 sensors, including a pressure sensor array in the sole, a 6-axis load cell in the ankle joint and an acceleration sensor were used. Local analysis and processing of this sensor information is done in the corresponding subsystems.

In technical systems, the sensors take over the task of the receptors; stimuli coming from outside are transformed in such a way that they can be interpreted. Like the specialized receptors, the sensors also react to a specific stimulus, which is then processed. With a high number of sensors,

³ <https://robotik.dfki-bremen.de/de/forschung/projekte/istruct.html>.

large amounts of data are generated. Transmitting and processing all the data would put a great load on the capacity of the communication network and the central control unit. Local pre-processing, evaluation, and interpretation of the sensor data, which is already running on a large number of electrical components within the modular subsystems, drastically reduces the data volume. In contrast, the global knowledge about the state of the system increases, since an interpretation of the data has already been performed locally.

In 2015, the explorer team of Valles Marineris became a new member. The existing team was missing a robotic platform that can move within the craggy rock formations and penetrate and navigate into caves. Charlie was used to fill the remaining gap in the robotic swarm. In addition, Charlie was used to test a novel visual positioning approach by LMT of the Technical University of Munich and NavVis GmbH. Especially in areas of low brightness, position determination based on continuous visual odometry using a stereo camera, exposure times would be too long or would require continuous and thus resource-intensive illumination. Therefore, Charlie was equipped with a 360° panoramic camera to explore a novel visual positioning and mapping approach. It was shown that single panoramic images and HDR images taken from a standstill with longer exposure times were sufficient to enable localization. Another result of the project was that the visual navigation approach is able to robustly recognize previously visited locations (despite the often poorly distinguishable surrounding texture).

While this approach for visual navigation enables accurate position determination across wide-area environments even in low-light conditions, a complementary approach is needed to determine position change between panoramic views. Especially when traversing rugged terrain, it is critical to precisely place the “legs” of the walking robot on stable surfaces. To make this possible, a proprioceptive approach was researched that uses tactile sensors to detect body position and movement in space and converts this into position information. This is a prerequisite for motion planning and reactive motion control that enables overcoming obstacles. By fusing the tactile data with visually perceived surface structures such as edges and crevices, these two technologies complemented each other to form a very promising approach for reliably overcoming difficult terrain.

The reactive motion control in Charlie was extended by further behavior modules, so that a safe locomotion over even as well as uneven ground and overcoming obstacles with the robot could be demonstrated. In addition, an algorithm for optimal foot placement was developed. This adaptive

footplacement algorithm allows to find an optimal foot contact point for each leg either between or including a wide variety of obstacles using a local map. It is relevant to mention that this is not a purely planning-based walking control of the robot. The reactive walking control is maintained, the planning layer is only allowed to write offsets on the respective walking pattern of the different legs. This is done in real time and extends the mobility of the robot in that it does not have to stop to plan its next steps when the ground is uneven or in front of obstacles. Even if the ground does not behave as expected (for example, due to compliance of an obstacle where contact between the foot and the obstacle has been included in the step cycle), the robot is able to continue its locomotion in a stable manner due to the permanently active reactive control plane.

Charlie's motion controller supports versatile walking gaits, i.e., crawling and trotting in combination with different postures and adaptive behaviors. However, each parameterization of the motion controller generates walking behavior that have their pros and cons on different substrates or applications. A crawl gait, for instance, is comparably slow but stable and should be used on rough terrain, whereas a trot gait is more energy efficient and more suited for flat terrain. Thus, a context-dependent behavior adaptation is needed to select the most suitable behavior for different surfaces and tasks to use the potential of the flexible locomotor system and maintain maximum stability and efficiency. In [20], an experience-based approach was developed and tested on Charlie to address this challenge. It detects the current context in terms of desired velocity and surface properties and uses past experience, i.e., evaluated behaviors in similar contexts, to generate a behavior that is supposed to deliver the application-specific best results. The approach is learning through the application and evaluation of new behaviors, increasing the knowledge base and thus improving the autonomous behavior adaptation.

Traversing uneven and inclined terrain always raises a potential threat to endanger a robot's stability. In general, Charlie's robustness due to integrated sensors, actuators, and processing units, usually prevents the system from the undesired event of tipping over. However, having the ability to self-righten itself is another crucial ability to proceed with the current mission. Therefore, [28] developed and tested self-righting behaviors for Charlie. Hand-crafted, optimized and evolutionary learned behaviors have all been investigated and evaluated. An informative video of such a self-righting behavior can be found in [29].

11.4.2 Design limitations or lessons learned

Charlie showed that its design is well suited for exploration tasks in unstructured terrain. Its sensor disposition and distributed processing power allows stable quadrupedal walking motions with proper adaptations to terrain irregularities. However, its compliance is limited to contacts that are sensed through the available force-torque and foot sensors. A compliance on joint-level could further improve the adaptability to the environment and thus increase the stability. Advancing to a torque-based control would also allow to apply state-of-the-art dynamic control approaches, including whole-body control and model predictive control. This would also stabilize the posture change from four to two legs and could enable bipedal walking. The latter would also require a redesign of the configuration of the hip joints. The current design produces quite wide lateral stance that is not optimal for bipedal walking.

11.4.3 Future work

In future projects, further work on dynamic control based on HyRo-Dyn [32] is planned to improve Charlie's robustness and mobility in harsh environments. In addition, Charlie is one robotic system in the DLR-funded project NoStrandAMust (Grant Number 50RA2122). In the project, data from different robots on different soils will be recorded in order to learn soil interaction models with them. These will be integrated into a simulation environment and then used for autonomous adaptation of path planning or locomotion to increase the safety and efficiency of the robots. Thus, Charlie's sensing and actuation capabilities will be further exploited to eventually emerge to robust exploration system that will serve as a blueprint for future space exploration systems.

Acknowledgment

The presented work was carried out in the project iStruct and VIPE funded by the German Space Agency (DLR, Grant numbers: 50RA1013, 50RA1014, and 50NA1516) with federal funds from the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament.

References

- [1] G. Bourne, *The Chimpanzee. Anatomy, Behaviour, and Diseases of Chimpanzees*, Karger Basel, New York, 1969.
- [2] R.A. Brooks, How to build complete creatures rather than isolated cognitive simulators, *Architectures for Intelligence* (1991) 225–239.

- [3] S. Thorpe, R.H. Crompton, Orangutan positional behavior and the nature of arboreal locomotion in hominoidea, *American Journal of Physical Anthropology* 131 (2006) 384–401.
- [4] M.P. Murphy, A. Saunders, C. Moreira, A.A. Rizzi, M. Raibert, The littledog robot, *The International Journal of Robotics Research* 30 (2) (2011) 145–149, <https://doi.org/10.1177/0278364910387457>, <http://ijr.sagepub.com/content/30/2/145.full.pdf+html>, <http://ijr.sagepub.com/content/30/2/145.abstract>.
- [5] M. Hutter, C. Gehring, M. Bloesch, M.A. Hoepffinger, C.D. Remy, R. Siegwart, Starleth: a compliant quadrupedal robot for fast, efficient, and versatile locomotion, in: *Climbing and Walking Robots (CLAWAR)*, 2012.
- [6] D. Spenneberg, F. Kirchner, Omnidirectional walking in an eight legged robot, in: *International Symposium on Robotics and Automation (ISRA)*, 2000.
- [7] A.S. McEwen, M.C. Malin, M.H. Carr, W.K. Hartmann, Voluminous volcanism on early Mars revealed in valles marineris, *Nature* 397 (6720) (1999) 584–586.
- [8] R.H. Crompton, Inertial properties of primates: new techniques for laboratory and field studies of locomotion, *American Journal of Physical Anthropology* 99 (1996) 547–570.
- [9] I.A. Kapanđji, *Funktionelle Anatomie der Gelenke – Schematisierte und kommentierte Zeichnungen zur menschlichen Biomechanik Band 1 – untere Extremität*, 3rd edition, Hippokrates Verlag, Stuttgart, 1999.
- [10] H. Elftman, J. Manter, The evolution of the human foot, with especial reference to the joints, *Journal of Anatomy* 70 (Pt 1) (1935) 56.
- [11] T.L. Allinger, J.R. Engsborg, A method to determine the range of motion of the ankle joint complex, in vivo, *Journal of Biomechanics* 26 (1) (1993) 69–76, [https://doi.org/10.1016/0021-9290\(93\)90614-K](https://doi.org/10.1016/0021-9290(93)90614-K), <http://www.sciencedirect.com/science/article/pii/S002192909390614K>.
- [12] J.M. DeSilva, Functional morphology of the ankle and the likelihood of climbing in early hominins, *Proceedings of the National Academy of Sciences* 106 (16) (2009) 6567–6572, <https://doi.org/10.1073/pnas.0900270106>, <http://www.pnas.org/content/106/16/6567.full.pdf+html>, <http://www.pnas.org/content/106/16/6567.abstract>.
- [13] A. Aiello, L. Aiello, C. Dean, *An Introduction to Human Evolutionary Anatomy*, Elsevier Academic Press, London, 2002, <http://discovery.ucl.ac.uk/56949/>.
- [14] I.A. Kapanđji, *Funktionelle Anatomie der Gelenke – Schematisierte und kommentierte Zeichnungen zur menschlichen Biomechanik Band III: Wirbelsäule*, 3rd edition, Hippokrates Verlag, Stuttgart, 1999.
- [15] D. Stewart, A platform with six degrees of freedom, *Proceedings of the Institution of Mechanical Engineers* 180 (1) (1965) 371–386.
- [16] H. Woehrle, M. Tabie, S.K. Kim, F. Kirchner, E.A. Kirchner, A hybrid FPGA-based system for EEG- and EMG-based online movement prediction, *Sensors* 17 (7) (2017) 1552.
- [17] M. Zenzes, P. Kampmann, M. Schilling, T. Stark, Simple protocol for heterogeneous embedded communication networks, in: *Proceedings of the Embedded World Exhibition and Conference*, 2016.
- [18] C. Stoeffler, A. del Rio Fernandez, H. Peters, M. Schilling, S. Kumar, Kinematic analysis of a novel humanoid wrist parallel mechanism, in: O. Altuzarra, A. Kecskeméthy (Eds.), *Advances in Robot Kinematics 2022*, Springer International Publishing, Cham, 2022, pp. 348–355.
- [19] A. Dutta, D.H. Salunkhe, S. Kumar, A.D. Udai, S.V. Shah, Sensorless full body active compliance in a 6 DOF parallel manipulator, *Robotics and Computer-Integrated Manufacturing* 59 (2019) 278–290, <https://doi.org/10.1016/j.rcim.2019.04.010>, <https://www.sciencedirect.com/science/article/pii/S0736584518303296>.

- [20] A. Dettmann, Experience-based behavior adaptation of kinematically-complex robots, Ph.D. thesis, Universität Bremen, 2021.
- [21] D. Kuehn, A. Dettmann, F. Kirchner, Design and evaluation of an active artificial spine in a hominid robot, *International Journal of Mechanical Engineering and Robotics Research* 9 (3) (2020) 387–394.
- [22] D. Kuehn, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, System design and field testing of the hominid robot Charlie, *Journal of Field Robotics* 34 (2017) 666–703.
- [23] A. Dettmann, D. Kühn, F. Kirchner, Control of active multi-point-contact feet for quadrupedal locomotion, *International Journal of Mechanical Engineering and Robotics Research* 9 (4) (2020) 481–488.
- [24] R.B. Rusu, S. Cousins, 3D is here: point cloud library (PCL), in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 1–4.
- [25] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, g2o: a general framework for graph optimization, in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 3607–3613.
- [26] M. Likhachev, Search-based planning with motion primitives, 2010.
- [27] J. Or, Humanoids grow a spine: the effect of lateral spinal motion on the mechanical energy efficiency, *IEEE Robotics & Automation Magazine* 20 (2) (2013) 71–81, <https://doi.org/10.1109/MRA.2012.2185990>.
- [28] S. Teiwes, Development of a self-righting behavior on the four-legged walking robot Charlie, Master's thesis, University of Bremen, 2020.
- [29] S. Teiwes, A self-righting behavior of the walking robot Charlie, https://youtu.be/ECR_AXm8g4Y, 2020.
- [30] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Mechatronics* 68 (2020) 102367, <https://doi.org/10.1016/j.mechatronics.2020.102367>.
- [31] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: J. Lenarcic, V. Parenti-Castelli (Eds.), *Advances in Robot Kinematics 2018*, Springer International Publishing, Cham, 2019, pp. 431–439.
- [32] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, *Journal of Mechanisms and Robotics* 12 (2) (2020) 021114, <https://doi.org/10.1115/1.4045941>.

This page intentionally left blank

CHAPTER 12

Multi-legged robot Mantis

Wiebke Brinkmann^a, Alexander Dettmann^a, Leon C. Danter^a,
Tobias Stark^a, Christopher Schulz^a, Holger Sprengel^a, Marc Manz^b,
and Sebastian Bartsch^b

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bAirbus Defense and Space (ADS), Bremen, Germany

12.1 Introduction

Mantis is an extraordinary robotic system in which nature has served as inspiration. The biological model was a praying mantis, which now exists in robotic form in the dimension of 1.96 m x 1.84 m x 0.32 m and 109 kg. How it came to this development and from which components Mantis is composed, as well as in which application areas it can be used and was used is described in the following sections.

12.1.1 Problem description

Development of robotic systems that cover the possibility to be capable of walking in unstructured terrain and being able to provide dual arm manipulation capability are a big challenge. The need for extended manipulation capabilities in future space missions is mentioned within several space agency guidelines, such as NASA's Vision for Space Exploration. In order to work in cooperation with humans in prospective space missions, it is advantageous to provide dual arm manipulation [1]. The envisaged planetary mission scenarios pose several functional demands on hardware and software. The system needs to cope with uneven terrain and slopes of up to 35°, even on loose surfaces. Components have to be sealed to protect from dust and other substances as to avoid abrasion and short circuits. Sensors are required to sense the integrity of the surface underfoot, the system state, the environment and the objects to be manipulated. To achieve a certain level of mission reliability, space systems are often designed redundantly to increase reliability when a subsystem or component fails. A commonly used concept for redundant systems is the implementation of redundant avionics, motor windings, or even complete propulsion systems, but concepts such

as replacing failed components or subsystems could also be a viable solution. A hyperredundant system with multiple degrees of freedom could also be used in a reconfigured locomotion pattern to continue operations. This solution can be observed in nature in various creatures that adapt locomotion patterns after an injury to an extremity in order to move forward as efficiently as possible with this limitation. Redundancies are required on software and hardware levels to respond to changes of environment and unforeseen failures of sub-components such as single legs or sensors [2]. To provide redundancy, aid stability and render the design more efficient, the arms can also be used as legs. The system is aimed to be capable of stable movement while simultaneously manipulating objects with its arms. To walk in a statically stable pattern at least four legs are required. This allows one leg to be lifted and set to a new position, while the others support the body. The described requirements are the outcome of thorough mission analyses. The goal in Mantis development was to use knowledge and experience from previous projects in combination with existing technologies and improve those in an effort to create a new design with maximum efficiency [1] [3].

Mantis is a walking robot with six extremities (four legs and two arms) developed in the project LIMES.¹ The main challenge was the development of a robot prototype, which combines manipulation and locomotion abilities. Including the goal to provide the system with mobility in a variety of terrain as well as an intelligent control framework capable of choosing the most suitable form of locomotion depending on perceived surface conditions by generating and optimizing different forms of locomotion [1].

12.1.2 Biological inspiration

For an efficient locomotion design it is apparent to reduce the number of extremities to a minimum. In order to accomplish the combined tasks of stable walking and concurrent dual arm manipulation it was determined that a minimum of six extremities are necessary. The first step was to position the six extremities around the body. The inspiration for this implementation were biological systems such as ants, spiders, crabs, and praying mantis as well as existing robotic systems, such as CENTAUR [4], Spidernaut [5], or ATHLETE [6] from NASA. Especially the static

¹ Learning Intelligent Motions for kinematically complex legged robots for Exploration in Space – <https://robotik.dfki-bremen.de/en/research/projects/limes.html>.

stability in manipulation posture was a crucial element. The praying mantis (Fig. 12.1A) was chosen as the most suitable biological inspiration, as it also uses its arms for manipulation and locomotion. The derived configuration (Fig. 12.1B) minimizes the complexity needed to straighten up the body for manipulation tasks due to a decoupled joint constellation. The main characteristic of the mantis concept is its high stability, which is achieved by a low center of mass (COM) with respect to the length and width [1].

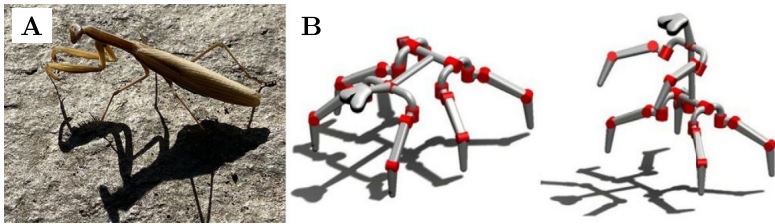


Figure 12.1 The praying mantis (A) serves as biological inspiration for the derived concept (B). (Credit: A - Katja Danter, B - [1].)

A low CM is one of the main factors which enables insects to scale steep slopes [7]. Possibly this is the result of diverging optimization goals in the evolutionary development of mammals and insects. Insects attain great stability through the number of legs (always six), which allows stable walking and running patterns as well as an unusually low CM. Even mammals, with a high CM with respect to their body length try to reduce the height of their CM when dealing with steep slopes, but the main strategy is to shift the center of mass [8]. This is a primary reason for choosing an insect body layout to realize a stable walking machine. For the intended use this is one important feature, the other is a high static stability in manipulation postures and low torque required to straighten up the body. To achieve this, it needs to force the location of its center of mass near to the leg rockers where the two leg pairs are mounted on. To this end, most of the components are located in a kind of abdomen to form a counter weight to the arms and the upper body. If the body is in manipulation posture, the center of mass still remains near to the center of the support polygon and thereby the system's static stability persists. The two rockers are actuated by linear drives. If the robot is in the manipulation posture these linear drives enable the robot to vary its manipulation height by extending or contracting them simultaneously. If they move in the opposite direction, the body tilts to the left or right [1].

12.1.3 Application scenarios

PRO-ACT,² a project funded by the European Commission Horizon 2020 Space Strategic Research Cluster,³ has been one of the most recent projects involving Mantis as robotic system. Its strengths of stability and flexibility in combination with its advanced locomotion and manipulation abilities made Mantis a valuable robotic agent for the project's mission scenario, aimed at cooperative manipulation tasks in a multi-robot team. In the scope of the project a new five-legged transport mode was added to enable higher locomotion velocities during manipulation tasks. In summary, this enabled Mantis to operate in three different modes [9]:

1. In locomotion mode, the robot walks on all six extremities. This is advantageous in difficult terrain and contributes to the excellent all-terrain capabilities of Mantis (Fig. 12.2A).
2. In manipulation mode, Mantis is in an upright position and uses the four rear legs for locomotion and the two arms for manipulation. This posture provided Mantis dual-arm manipulation capabilities while being firmly grounded (Fig. 12.2B).
3. In transport mode, the four legs and one arm are on the ground while the other arm is in the air to grab and hold an object. In this five-legged mode, Mantis can transport a payload with one arm and has a more stable and dynamic walking ability than in the four-legged manipulation mode (Fig. 12.2C).



Figure 12.2 Mantis in its six-legged locomotion mode (A), four-legged manipulation posture (B), and five-legged transport mode (C). (Credit: Annemarie Popp, DFKI.)

² Planetary RObots deployed for Assembly and Construction Tasks – <https://www.h2020-pro-act.eu>.

³ <https://ec.europa.eu/programmes/horizon2020>.

12.2 Mechatronic system design

The development of the Mantis mechanical structure, the selection of the electrical component, and the software design are described in more detail below. The existing parallel partial mechanisms on the main body and the four ankles are particularly emphasized.

12.2.1 Mechanical design

The first requirement that was established was the degrees of freedom (DOF) necessary for the manipulation and locomotion. To allow remote operators an intuitive interaction with the environment, it makes sense to provide an anthropomorph arrangement of the arms or at least similar capabilities [10] [11]. For this purpose a seven DOF manipulator design was chosen. Furthermore, it was apparent from previous projects that dual arm manipulation requires an additional DOF in order to increase the workspace of the dual arm configuration. A further advantageous DOF was defined by the ability to rotate the spine in order to allow the upper body to turn whilst the torso remains static. Each leg has six DOFs, four of which are used to position the foot on the ground, the remaining two adapting to the surface to increase traction [1].

Mantis is designed following the main idea to create a robot which is able to walk statically stable while manipulating with two arms. Although statically stable walking is possible on two extremities, walking on four extremities while maintaining ground contact with at least three of them provides greater stability, which is particularly useful when additional forces arise during manipulation. A system like Mantis is well-suited for applications requiring high manipulation payloads during locomotion, e.g., clearing disaster sites or setting up infrastructure in a planetary exploration setting. In the original design, Mantis possesses six extremities for locomotion, each having six DOF. In addition, Mantis is able to erect its body and free the two foremost extremities to use them as arms, both featuring three-fingered grippers for dual arm manipulation and a bracket to walk on it. The main electronic compartment is located in the rearmost body segment, the abdomen. It provides a counterweight for the upper body and thus shifts the center of mass towards the frame articulation. This feature facilitates switching between the locomotion and manipulation postures. In the former, the actuated frame articulation allows the shifting of the center of mass along the robot's longitudinal axis if both linear actuators extend or retract simultaneously, while in the latter two types of movement are possible: simultaneous movement of the actuators in the same direction leads

to an alteration of the robots height. Opposing movement allows the robot to lean to the left or right. The sensor head is actuated by three joints and contains a stereo camera system, an inertia measurement unit, and a lidar sensor. The first joint is used to compensate the torso's pitch, simplifying control algorithms and allowing intuitive teleoperation of the system. Altogether, Mantis has 61 active DOF used to control the movements of its body. The system with dimensions of 1.96 m x 1.84 m x 0.32 m (six-legged walking pose) and an overall weight of 109 kg is able to carry 40 kg of payload.

Due to the use of Mantis in a space-related project, where several robot systems had to work cooperatively, changes and improvements to the design of Mantis became necessary [12] [13] [14] [15] [9]. In order to cover the needs of the planned demonstration mission scenarios, several mechanical adaptations on Mantis are performed:

- Redesigned and integrated grippers on both forearms to support the desired dual-usage for locomotion and manipulation without manual intervention.
- Modifications of the ankle joints of the legs for increased robustness.
- Reinforced hip linkages of the legs to support higher loads.
- Integration of additional hardware to build up on advanced and standardized outcomes from previous European space robotic projects (so-called operational grants (OGs) from the project PERASPERA Horizon 2020) to integrate Mantis into a heterogeneous multi-robot team.

The original grippers of Mantis needed to be manually switched from manipulation to walking mode. Furthermore, their payload was too low for the tasks required in the aforementioned project. To cater to the new requirements, remove the need for manual intervention and to increase their payload capability, the forearms of Mantis were redesigned in 2020. The new grippers require no mechanism to switch between locomotion and manipulation modes because they are fully integrated in the forearm structure. Additionally, their clamping force was increased significantly. They can grip and hold objects of up to 20 kg. The structure consists of riveted aluminum sheets covered by a 3D-printed enclosure and the gripper consists of a fixed and a moving clamp, actuated by a BLDC motor that drives a ball screw.

The contact surface for walking is an ellipsoidal section to accommodate varying postures. It is built from rubber brackets that can be easily replaced in case of wear or damage. A 6-axis FT-sensor is integrated in the forearm

structure for measuring ground contact and manipulation forces. An additional force sensor between the moving clamp and linear actuator measures the clamping force and reed contacts are used as end-stops to initialize the gripper.

At the present time, Mantis features six extremities for locomotion: four legs and two forearms with integrated grippers. The remaining chapter will refer to the current design of Mantis.

12.2.2 Parallel kinematics

Mantis features parallel kinematic mechanisms in several locations:

- Transition area between head and torso (main body);
- The hip joint for lifting the leg;
- The ankle joint.

12.2.2.1 Main body

As mentioned above, a parallel kinematic is located in the transition area between head and torso, also called main body. The main body of Mantis is constructed in such a way that the end part (torso) is connected to a carbon tube to which the parallel kinematics are attached. More precisely, the parallel kinematics consists of the main components center part (carbon tube), a central joint, two electric linear cylinders and two bearings for the linear cylinder Fig. 12.3.

The legs can be moved independently of each other via linear cylinders. The legs are swivelled around the central middle joint. The swivel angle is $\pm 22^\circ$ and is executed via linear cylinders. The angular position of the legs is monitored by two rotary potentiometers, which are integrated in the central joint. The linear cylinder does not contain its own displacement transducer. A force sensor is located in the cylinder to control the push and pull forces. The sensor system (displacement transducer; force sensor) is used to control/activate the walking movement. The robot can straighten up and move away via the kinematics. The two front legs can be swivelled back and forth via the parallel kinematics. The feed in the cylinder is generated by a spindle drive (purchased part). In addition to the spindle, the drive consists of an electric motor and gearbox, flange-mounted electronics and an integrated force sensor. The force torque sensor records the push and pull forces from the rear rod end (electric cylinder). A holding brake is not necessary, since the spindle is designed to be self-locking. If the robot is upright and the power supply fails, it will remain in the respective position. The design of the individual parts is such that the linear cylinder is

protected against moisture and dust. The sensitive components are encapsulated. The linear cylinder is a self-locking, threaded spindle with integrated load cell and encoder. The push or pull force can withstand 710 N permanently, 1530 N briefly, and 2700 N statically. The feed rate is 5.2 mm s^{-1} , the power is 40 W, and the rated voltage is 36 V.

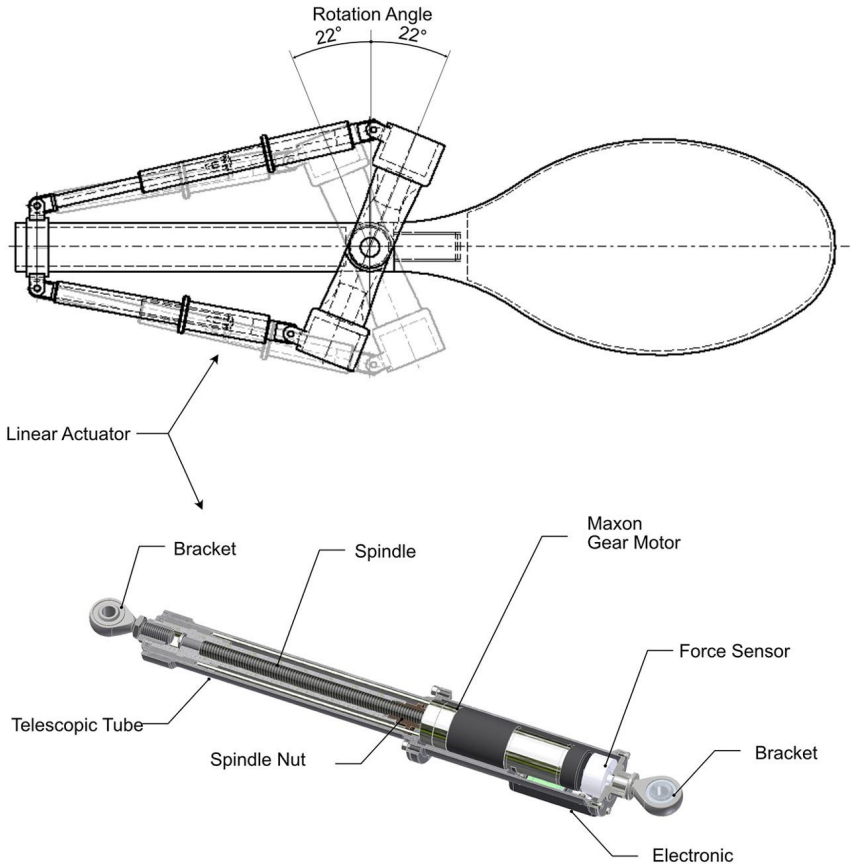


Figure 12.3 Overview of the parallel kinematic of the main body.

12.2.2.2 Ankle joint

Nature has evolved and optimized the shape and arrangement of living beings over millions of years. These optimizations are governed by the laws of physics and should be taken into account when developing biologically inspired robots. For example, there is a scaling problem in the development

of robots inspired by insects. The structure and the length ratios cannot be scaled linearly because the load-bearing cross-sectional areas increase by the square and the volume is responsible for the weight.

In addition, it can be assumed that on soft ground with high weight, a sufficient foot surface is required that is more similar to an elephant or camel foot than an insect foot.

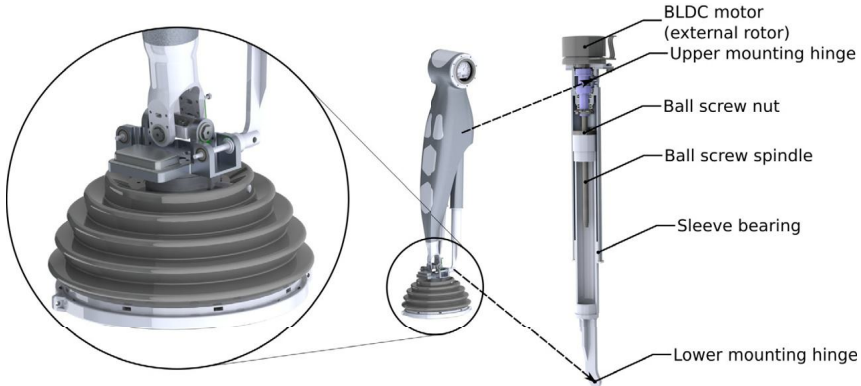
Due to the size of Mantis, a flat foot with a spherical joint was developed to avoid sinking too deeply into loose ground. This mechanism is stable because the theoretical pivot point is below the surface on which the robot is moving. However, the adaptability is limited and therefore this passive foot was combined with an active mechanism to adapt the foot specifically to the ground with comparatively slow and small actuators and to realize the adaptation to not detected ground unevenness by the passive, non-actuated foot.

The ankle joint of Mantis has two active DOF. It consists of a universal joint that is actuated by two linear actuators, as depicted in Fig. 12.4. Employing this parallel mechanism instead of two rotary actuators in series has considerable advantages in this application:

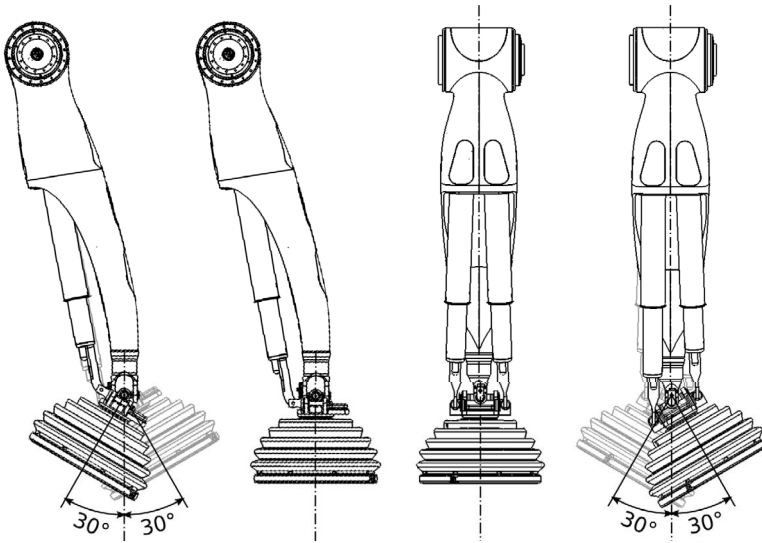
1. The relatively heavy actuators can be placed close to the base of the limb, thereby reducing its inertia
2. The universal joint itself can be built very small and light
3. The linear actuators create additional load paths in parallel to the limb's structure
4. The linear actuators are self-locking and therefore no extra power is needed to hold positions

Each linear actuator consists of a BLDC motor that drives a ball spindle. The spindle is fully encased in telescopic aluminum cylinders – one static cylinder connected to the actuator base and one moving cylinder connected to the nut and the output hinge. The inner cylinder glides in a sleeve bearing mounted inside the outer cylinder. Thus, the aluminum tubes increase the actuator's stiffness and aid in preventing stability failure.

Each actuator is connected to the limb and to the foot via a gimbal joint each. To not over-constrain the mechanism, the moving cylinder can rotate with respect to the actuator base about the spindle axis. In theory this changes the position of the actuator slightly, because the nut rotates about the spindle in this movement. In the working range of the ankle mechanism, however, these rotations are so small that they do not noticeably affect the actuator position.



a) Lower limb, ankle joint and linear actuator.



b) Neutral position and joint limits.

Figure 12.4 Lower limb of Mantis with ankle joint.

Since Mantis’ feet are flat (often, feet of quadro- and hexapod walking robots are ball shaped), the actuated ankle joint is required to keep the feet parallel to the ground plane. However, small variations and bumps can be accounted for by an additional passive mechanism in the foot. This mechanism allows the foot to self-adjust to local ground features via a ball joint consisting of two concentric hemispheres. To keep the robot stable,

the center of rotation of this mechanism lies below the foot's contact point with the ground.

12.2.3 Electronics design

Since the complexity of algorithms running on embedded hardware is increasing, the predominantly used term *low-level processing* is often not fully applicable anymore, which is why *first-level processing* is used here for this type of electronics. A driving aspect of shifting algorithms to controllers that are distributed across the robot are the equally distributed sensors. Generally, to support the possibility to design autonomous behavior with rising complexity and dealing with more than a limited set of tasks, sensors of different modalities are needed almost everywhere in a robot. Mantis has a total of 88 position encoders, 14 six-axis force-torque sensors, 2 IMUs, 2 HD cameras, 1 lidar, 122 temperature sensors, 191 current measurement sensors, and 12 tactile sensors comprising 40 sensing elements. The overall data volume generated by this system amounts to 629 MB/s. The designed hardware architecture that handles the processing of the sensor data and controlling of the robot consists of 62 FPGAs, 14 microcontrollers, and one standard x86 central processing unit integrated into the system [3].

12.2.4 Software design

All components of the Mantis software stack are developed in the ROCK⁴ framework. ROCK provides realtime capability as it is based on the Orocos Real Time Toolkit and was specifically designed for the development of robotic systems. A ROCK network consists of several ROCK tasks, which are compiled C++ executables with configurable input and output ports. In order to run the robotic system, ruby scripts are executed, which start and connect all required ROCK tasks, e.g., drivers and controllers, to put the robot into a functional running state. This script-based startup procedure allows to define and run different configurations to adapt to task or project specific requirements. Moreover it allows components to be added, removed or swapped at runtime, which not only facilitates development, but also enables dynamic reconfiguration to adjust to environmental changes or failure of components. Furthermore, this approach allows the software stack to be connected either to hardware drivers or a simulation tool. Keeping the software identical for simulation and application eases maintainability, facilitates development, and enhances robustness.

⁴ Rock: The Robot Construction Kit (<http://rock-robotics.org>).

12.3 Modeling and control

12.3.1 Modeling

Two distinct types of parallel kinematics mechanisms of type 1-RR \underline{P} R and 2-SPU+1U are used in the Mantis robot. The detailed geometric model for the 1-RR \underline{P} R slider crank mechanism used in the hip and torso joints is available in Chapter 3. The geometric model for 2-SPU+1U orientational parallel mechanism used in the ankle joints is discussed in Chapter 4.

12.3.2 Low level control

Due to the fact that the Mantis robot has more than 40 actuators and a multitude of sensors, a powerful low-level control was needed. Therefore, some dedicated electronics for single components like actuators or force-torque sensors were used, which are distributed all over the robot to reduce the cabling effort, realize the low-level control locally and improve the modularity of the system (see Fig. 12.5).

For the communication between these components the NDLCCom protocol is used [16] which offers the possibility to connect different hardware components like PCs, microcontroller- or FPGA-based boards in a communication network. In this network, all components can communicate with each other so that local control loops are possible.

Such a local control loop is used within the gripper to control the grip force. For this purpose, a force sensor was integrated into the spindle mechanism of the gripper. The electronics of this force sensor sends the force information to the motor PCB, which adapts the motor position and/or current to produce the desired grip force.

Another local control loop is realized within the ankle joints. As mentioned in Sec. 12.2.2.2, the ankle joints consist of two linear actuators which drive an 2-DOF universal joint. To be able to control the two angles of the universal joint by the mid- or high-level control instead of driving the linear joints directly, the inverse kinematics are implemented within the control FPGAs of the local electronics. In contrast to the ankle joints used in the Charlie robot (see Sec. 4.2.1) here each of the linear actuators has its own control electronics and therefore these two electronics must be synchronized to drive the two motors and obtain smooth trajectories.

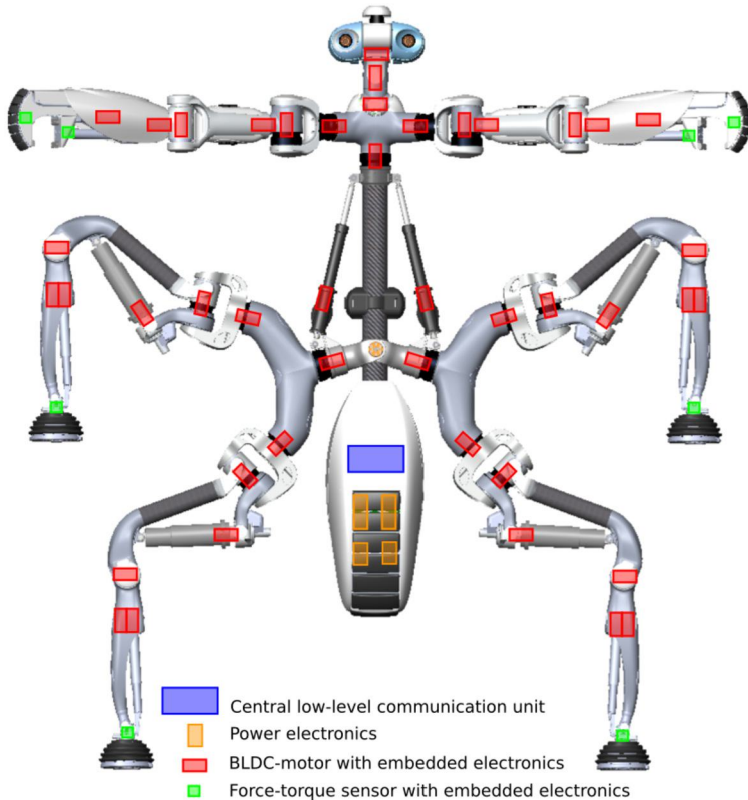


Figure 12.5 Mantis low-level control overview.

12.3.3 Mid level control

12.3.3.1 Locomotion control

Mantis is controlled via a modular, generic Central Pattern Generator (CPG) approach [17] that is also used in other walking systems such as SpaceClimber [18] or Charlie [19]. A CPG generates a clock, which triggers consecutive leg movements. The resulting open-loop Cartesian foot trajectories are then superposed by reactive modules to adapt to the surface. Finally, inverse kinematics are used to generate the target values for the position-controlled joints.

The overall concept follows key principles of a behavior-based control architecture, where the overall robot behavior is generated by the interaction of multiple processing units, also called behavior modules. Two basic guidelines are followed to generate a generic and reusable control archi-

ture. First, the overall control is split into general-applicable behavior modules, which are required for every system, and robot-specific behavior modules that extend the general architecture and exploit the capabilities of the target system. Second, behavior modules are grouped to larger behavior modules to implement a hierarchical structure that efficiently reuses functionalities.

Every behavior module provides a specific functionality contributing to the overall robot behavior. On the top layer, in- and outputs of the overall control as well as the main behavior modules are defined, e.g., the *CPG* behavior module to derive gait-dependent step cycles for the limbs and for the body, a *posture controller*, several *limb controllers* for every motion-supporting arm or limb, a *head controller*, as well as data processing nodes, which provide module-specific information of the current robot state. The architecture is a mixture of open-loop trajectory-generating modules and trajectory-adapting reactive modules to adapt to the environment. For Mantis, only specific body, leg, and arm kinematics were implemented.

Due to the modular and configurable architecture, the available four-legged crawl of Charlie and six-legged wave gait and tripod of Space-Climber could directly be reused on Mantis by just applying the corresponding parameterizations. A four-legged trot is due the morphology and limited dynamics not possible. Mantis additionally features a five-legged walking behavior which is just using one of the arms for locomotion. In this walking mode, no shifting of the body posture is needed as in the four-legged posture. Thus, it has similar efficiency compared to six-legged walking, but frees one arm for holding objects on the costs of slightly higher load on the arm that is used for locomotion. The transition between postures is handled through routines that adapt the posture parameters in a pre-given order.

12.3.3.2 Manipulation control

A whole-body control approach is used to manipulate objects while observing constraints such as stability, thus considering all body joints to bring the prippers into desired grasping poses. The huge amount of data is directly processed in the FPGA of each hand to reduce computation and communication load of the main CPU. For good grasping poses, the robot uses color and depth data from the cameras and lidar integrated in its head. This data gets fed to an ANN model that was pre-trained on labeled RGB-D images. For this pre-training of the neural network, the Cornwell data set and the data preprocessing methods from [20] are used. In order to reduce

the vast search space for possible grasps Bayesian optimization is used. The thus determined grasp poses are ranked using the ANN's output as a score. Before initiation of the actual closing movement of the gripper Mantis' manipulator has to be moved towards the object. This approach, which is part of the grasping procedure, is learned using imitation learning. A human demonstrates the grasping trajectory, which then gets represented by a Dynamic Movement Primitive and is transferred onto the robot using the learning platform described by [21] [3].

12.3.4 High level control

The motion controller as well as the sensor and actuator drivers are embedded in a modular architecture implemented in ROCK⁴. In order to provide a framework-independent interface to interact with the robot, one ROCK task is wrapping the `robot_remote_control` communication library [22]. This lightweight library offers two configurable communication channels, one for reliable commands and one for non-blocking telemetry transfer. A variety of generic data types are already defined and can easily be extended. Data transfer methods can be configured for each channel, already offering a range from fast inter-process communication to latency capable UDT transport.⁵

12.4 Conclusion and outlook

12.4.1 Success stories

Mantis was designed from scratch in the project LIMES, thereafter involved in the project BesMan and then further developed in PRO-ACT.

12.4.1.1 LIMES

Development of a highly mobile multi-legged walking robot with the ability to straighten up the upper body in order to use the front extremities as manipulation devices was the main goal of the project LIMES. The resulting prototype Mantis should demonstrate that such kind of systems can support future extraterrestrial missions with taking soil samples from difficult-to-access regions or assembling and maintaining infrastructure on rough and unstructured surfaces.

⁵ udt.sourceforge.io.

Beside the electromechanical development of the robot, the project focused on generating and optimizing different locomotion behaviors for traversing varying surface structures and subsoils with the aid of a simulation environment and machine learning methods.

Mantis demonstrated its mobility by traversing different substrates and obstacles. Its manipulation capabilities were additionally demonstrated at international fairs (e.g., HMI). Thanks to its cameras and many DoF, Mantis was additionally able to detect humans, follow their movements with posture changes and receive commands of the visitors through visual markers.

12.4.1.2 BesMan

The main goal of the project BesMan was the development of generic dexterous manipulation strategies that are independent of a specific robot morphology. The three key components were trajectory planning, whole-body sensor-based reactive control, and dynamic control into a modular, robot-independent, and easily-reconfigurable software framework that allows reusing components to describe a variety of complex manipulation behaviors. In addition, a situation-specific behavior shall be learned by means of a highly-automated machine learning platform, which incorporated an interface to a human operator, who via demonstration shew the robot how to deal with unforeseen situations. Mantis was involved performing an autonomous manipulation task in a spaceflight scenario and learning behavior through human demonstration [23].

12.4.1.3 PRO-ACT

In the EU project PRO-ACT, Mantis utilized its combined locomotion and manipulation capabilities within a lunar construction context. The aim was, that Mantis, being part of a multi-robot team, cooperatively mapped an area of interest and cooperatively transported equipment from the lander to the target region. In addition, the unfolding of a gantry crane for in-situ-resource utilization, was targeted. However, not being able to meet with all robots in the final phase of the project, most of the collaborative tasks could not be tested and demonstrated. In this way, working remotely across Europe on Mantis was made possible and different autonomy levels were demonstrated [9]. Instead, Mantis showed additional combined locomotion and manipulation tasks, e.g., pulling the 150 kg SherpaTT through fine sand. In order to facilitate working with the heterogeneous robots and the simulation, a framework-independent Application Programming

Interface (API) was developed as a common interface to control either the real world robots or the simulation. This so-called “Robot Common API” is based on the lightweight robot remote control library [22] and enables project partners to reliably command the robot and get non-blocking telemetry updates without the need to interact with the robot-specific framework.

12.4.2 Design limitations and lessons learned

Mantis is a comparably large and heavy walking robot, which makes it hard to handle. Transporting the system, retrieving the system from unstructured terrain, or simply activate the system before being able to place it on the ground requires several persons and plenty of space. The main driver for the development was the requirement to manipulate objects of about 10 kg. Even though this was achieved, Mantis’ size and weight requires joints with demanding nominal and peak torques. Very careful handling is required because potential malfunctions can easily harm operators or Mantis’ own structural components. Even nominal motions produce high tear and wear on structures and joints. In addition, the required high gear ratio in the joints prevents the system from precise current measurements and thus applying modern torque-based control approaches. The approach of a stiff position control in combination with force-torque sensors for haptic feedback provides terrain adaption capabilities, but also has its limitations in terms of proper reactions in influences that are not detected by sensors.

12.4.3 Future work

The potential of Mantis’ excellent sensor disposition shall be further exploited in the project NoStrandAMust. In NoStrandAMust, Mantis will use its proprioceptive sensors to investigate the ground interaction. The data will be processed by AI methods to learn ground interaction models. These models will then be used to classify the currently traversed surface and to predict the performance of the actual walking pattern. With the help of the ground interaction models, Mantis will know which walking pattern is most efficient and whether stability can be guaranteed. In case all possible walking behaviors are known to have low efficiency or, the area is marked in the map and avoided in upcoming path planning iterations to prevent the systems from maneuvering into dangerous areas.

References

- [1] M. Manz, S. Bartsch, F. Kirchner, Mantis – a robot with advanced locomotion and manipulation abilities, in: Proceedings of the 12th Symposium on Advanced Space Technologies in Robotics and Automation, 2013.
- [2] D. Spenneberg, K. McCullough, F. Kirchner, Stability of walking in a multilegged robot suffering leg loss, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, vol. 3, IEEE, 2004, pp. 2159–2164, <https://doi.org/10.1109/ROBOT.2004.1307382>.
- [3] S. Bartsch, M. Manz, P. Kampmann, A. Dettmann, H. Hanff, M. Langosz, K. von Szadkowski, J. Hilljegerdes, M. Simnofske, P. Kloss, M. Meder, F. Kirchner, Development and control of the multi-legged robot MANTIS, in: Proceedings of ISR 2016: 47st International Symposium on Robotics, VDE, 2016, pp. 1–8.
- [4] J.S. Mehling, P. Strawser, L. Bridgwater, W.K. Verdeyen, R. Rovekamp, Centaur: NASA's mobile humanoid designed for field work, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 2928–2933, <https://doi.org/10.1109/ROBOT.2007.363916>.
- [5] F. Rehnmark, R.O. Ambrose, B. Kennedy, M. Diftler, J. Mehling, L. Bridgwater, N. Radford, S.M. Goza, C. Culbert, Innovative robot archetypes for in-space construction and maintenance, in: AIP Conference Proceedings, vol. 746, American Institute of Physics, 2005, pp. 1070–1077, <https://doi.org/10.1063/1.1867231>.
- [6] B.H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirotka, B. Cooper, Athlete: a cargo handling and manipulation robot for the moon, *Journal of Field Robotics* 24 (5) (2007) 421–434, <https://doi.org/10.1002/rob.20193>.
- [7] M. Günther, T. Weihmann, Climbing in hexapods: a plain model for heavy slopes, *Journal of Theoretical Biology* 293 (2012) 82–86.
- [8] S.-j. Zhang, J. Tong, K. Hapeshi, D.-h. Chen, Optimising body layout design of limbed machines, *Journal of Bionic Engineering* 4 (2) (2007) 117–122.
- [9] W. Brinkmann, L.C. Danter, T. Stark, A. Dettmann, M. De Benedetti, S. Govindaraj, I. Sanz Nieto, A. But, F.J. Colmenero, E. Heredia, M. Alonso, Mantis within a multi robot team for lunar exploration and construction tasks, in: 2022 IEEE Aerospace Conference, IEEE, 2022, pp. 1–15.
- [10] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D.V. Dimarogonas, D. Kragic, Dual arm manipulation—a survey, *Robotics and Autonomous Systems* 60 (10) (2012) 1340–1353.
- [11] G.A. Landis, Teleoperation from Mars orbit: a proposal for human exploration, *Acta Astronautica* 62 (1) (2008) 59–65.
- [12] S. Govindaraj, J. Gancet, D. Urbina, W. Brinkmann, N. Aouf, S. Lacroix, M. Wolski, F. Colmenero, M. Walshe, C. Ortega, et al., PRO-ACT: planetary robots deployed for assembly and construction of future lunar ISRU and supporting infrastructures, in: Proceedings of the 15th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2019), 2019.
- [13] S. Govindaraj, I. Sanz, A. But, W. Brinkmann, A. Dettmann, L.C. Danter, N. Aouf, M.S. Bahraini, A. Zenati, H. Savino, J. Stelmachowski, F. Colmenero, E. Heredia, M. Alonso, J. Purnell, K. Picton, L. Lopes, Multi-robot cooperation for lunar base assembly and construction, in: Proceedings: International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2020.
- [14] W. Brinkmann, A. Dettmann, L.C. Danter, C. Schulz, T. Stark, A. Brandt, Enhancement of the six-legged robot Mantis for assembly and construction tasks in lunar mission scenarios within a multi-robot team, in: Proceedings: International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2020.

- [15] S. Govindaraj, W. Brinkmann, F. Colmenero, I. Nieto, A. But, M. De Benedetti, L. Danter, M. Alonso, E. Heredia Aguado, S. Lacroix, D. Kleszczynski, J. Purnell, K. Picton, N. Aouf, L. Lopes, Building a lunar infrastructure with the help of a heterogeneous (semi)autonomous multi-robot-team, in: Proceedings of 72nd International Astronautical Congress (IAC), 2021.
- [16] M. Zenzes, P. Kampmann, T. Stark, M. Schilling, NDLCOM: simple protocol for heterogeneous embedded communication networks, in: Proceedings of the Embedded World Exhibition & Conference, Nuremberg, Germany, 2016, pp. 23–25.
- [17] A. Dettmann, Experience-based behavior adaptation of kinematically-complex robots, <https://doi.org/10.26092/elib/480>, 2021.
- [18] S. Bartsch, T. Birnschein, M. Römmermann, J. Hilljegerdes, D. Kühn, F. Kirchner, Development of the six-legged walking and climbing robot spaceclimber, *Journal of Field Robotics* 29 (2012).
- [19] D. Kühn, Design and development of a hominid robot with local control in its adaptable feet to enhance locomotion capabilities, Ph.D. thesis, Universität Bremen, 2016.
- [20] I. Lenz, H. Lee, A. Saxena, Deep learning for detecting robotic grasps, *The International Journal of Robotics Research* 34 (4–5) (2015) 705–724.
- [21] J.H. Metzen, A. Fabisch, L. Senger, J. de Gea Fernández, E.A. Kirchner, Towards learning of generic skills for robotic manipulation, *KI. Künstliche Intelligenz* 28 (1) (2014) 15–20.
- [22] L. Danter, S. Planthaber, A. Dettmann, W. Brinkmann, F. Kirchner, Lightweight and framework-independent communication library to support cross-platform robotic applications and high-latency connections, in: Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2020.
- [23] L. Gutzeit, A. Fabisch, M. Otto, J.H. Metzen, J. Hansen, F. Kirchner, E.A. Kirchner, The BesMan learning platform for automated robot skill learning, *Frontiers in Robotics and AI* 5 (2018), <https://doi.org/10.3389/frobt.2018.00043>.

This page intentionally left blank

CHAPTER 13

Sherpa, a family of wheeled-leg rovers

Florian Cordes, Ajish Babu, and Tobias Stark

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

13.1 Introduction

This chapter presents a novel wheel-on-leg structure for planetary exploration rovers. The legs or suspension units are built using parallel mechanisms. In the following, we describe the problem the rover is designed to address, its biological inspiration and application scenarios.

13.1.1 Problem description

The rovers of the Sherpa family are all designed as parts of multi-robot exploration systems. Hence, some design drivers originate from their role within the multi-robot scenario. As for Sherpa and SherpaTT, the scenario is that of lunar polar crater exploration. For Sherpa, this means transporting the six-legged exploration system CREX up to a crater rim, deploy it there, and support the overall mission by making use of several modular elements, the so-called payload-items [1,2]. In the case of SherpaTT, the scenario is that of being deployed directly into a crater at the lunar south pole in order to do soil sampling and sample return in the context of search for water ice [3,4]. A second rover, the micro rover Coyote III [5] acts as a shuttle with which SherpaTT needs to interact and exchange payload-items with.

Using mobile robots in extraterrestrial exploration provides the possibility to take measurements and soil samples several kilometers off the landing site, [6]. All robots that have been deployed in exploration of Moon or Mars so far are carrying all instrumentation and capabilities as one individual system. These robotic systems all feature passive suspension, for example

☆ This work is part of several projects, namely RIMRES (Grant: 50RA0904); TransTerra (Grant: 50RA1301); FT-Utah (Grant: 50RA1621); OG6 FACILITATORS (Grant: GA 730068); OG10 ADE (Grant: GA 821988); ROBDEKON (Grant: 13N14675).

NASA's Mars Exploration Rovers (MERs) [7] and Mars Science Laboratory (MSL) [8] on Mars or China's Yutu rover on Moon.

Despite their proven durability, the areas and types of terrain these robots negotiate are quite conservative in terms of locomotive challenges. Yet, scientifically interesting sites are often located in difficult-to-access areas and consequently are out of safe reach for the currently deployed types of robots [9–11].

Hence, the accessibility of scientifically interesting areas might pose contradicting requirements on the design of a robotic system: A longer distance from the landing site to the sampling site is preferably covered by energy-efficient (that is, wheeled) locomotion. Steep slopes and vertical cliffs might need to be overcome to reach the sampling site. Here, a locomotion approach with discontinuous ground contact (i.e., walking/climbing capabilities) is required. Consequently, it might be beneficial to combine different locomotion capabilities into one system, additionally to the Multi-Robot System (MRS) approach described above.

The Sherpa family of rovers provides such a new type of rover locomotion system to meet above challenges. The novel approach is that of an active suspension system in order to (i) maintain an energy-efficient wheeled locomotion, while (ii) allowing for active ground adaptation and (iii) enabling traverses of walking locomotion in case of otherwise unmanageable terrain. With this type of suspension, wheels can be lifted off the ground to be freed in case they are stuck in very soft soil. Even sequences of walking become possible which, however, should be limited to very challenging situations for reasons of energy efficiency. A side effect of this design is that a failure in a wheel drive could be overcome by lifting this leg and continuing on three wheels.

13.1.2 Biological inspiration

The first generation of Sherpa featured only one Degree of Freedom (DoF) for lifting the leg end point. Instead, the “knee” DoF had only a short linkage and was intended for better ground adaption of the wheel on slopes. Experience showed that this was rarely necessary, thanks to the flexible wheels. However, the design compromised walking motion patterns and prohibited pure z-motion of the leg endpoint [12].

The suspension system of the second Sherpa generation then took inspiration from the animal kingdom for improved locomotive capabilities. Just as used in the SCORPION robot [13] or other insect-inspired robots,

the suspension system features one DoF for moving the leg end point forward/backward (the Thorax joint) and two DoFs for lifting the leg end point (Basal and Distal joints). With this combination of DoF, a free positioning of the leg endpoint within the workspace of a leg is possible, since three joints are used for the three position coordinates of a leg end point [14]. More details on the workspace and kinematics of SherpaTT's suspension are provided in Section 13.3.

13.1.3 Application scenarios

Originally, the Sherpa development was for space exploration scenarios. Modularity, flexibility, and cooperation with other robots were in the foreground of the developments. Since the system design is flexible and the suspension system allows for adaption to various environments, other application scenarios are feasible. Search and Rescue scenarios have much in common with space scenarios in terms of rugged terrain, limited communication, and exploration requirements, such as annotating maps with measurements of environmental parameters. In fact, SherpaTT is currently deployed in a robotic decontamination scenario in landfills and works together with a semi-autonomous excavator.¹ In another project, the system SherpaTT is currently evaluated in an agricultural setting, where the locomotive capabilities are compared to commercially available agricultural robotic systems. With the maritime version SherpaUW, the application in sub-sea scenarios is also feasible. Here, checking of trenches for sub sea cables, inspection of infrastructure, and manipulation tasks (for example, closing of valves) is possible.

13.2 Mechatronic system design

This section deals with the design of the SherpaTT rover. It highlights the overall design and focuses on the serial-parallel kinematic structure of the legs, constituting the locomotive system. In total, three different members of the Sherpa rover family have been built up, Fig. 13.1. The Sherpa rover with a single parallelogram structure per leg is the initial design, this system is briefly described in Section 13.2.1.1. The improved design of SherpaTT is described in Section 13.2.1.2. This rover makes use of two serially coupled parallel structures per leg. Sherpa and SherpaTT are directly compared

¹ <https://www.youtube.com/watch?v=Wnw1ihHqFgQ>.



Figure 13.1 The Sherpa family of rovers.

against each other in [12]. With SherpaUW, an underwater version of SherpaTT using the exact same leg kinematics was built. Since the kinematics of the legs are identical, the descriptions provided in this chapter are focusing on SherpaTT.

13.2.1 Mechanical design

In this section, we briefly describe the initial Sherpa suspension design. Then we focus on the current design of SherpaTT, which is also valid for SherpaUW.

13.2.1.1 Initial design: Sherpa rover

The rover system Sherpa is depicted in Fig. 13.1a in an indoor laboratory environment. The system has a total mass of about 160 kg, each leg as well as the manipulation arm weights about 25 kg, [15]. It is powered by a battery pack with a nominal voltage of 44.4 V with 8 Ah capacity. Main sensors for autonomous locomotion are a stereo camera and a tiltable laser range finder mounted at the front face of the main body. Key aspects of the system are described in [1].

Sherpa features an active suspension system constituted from four legs, each with 6 DoF. The suspension can be used to lower the body to the ground. The workspace of the legs also allows lifting them with the body on the ground or stepping on high obstacles [12] (e.g., to overcome them).

Sherpa is part of a modular MRS and features six Electromechanical Interfaces (EMIs) in total: four payload bays, each equipped with a passive EMI, are mounted around the central manipulation tower, the manipulation arm itself makes use of an active EMI, and a second active EMI is

mounted below the central body. More details on the modularity within the Sherpa rovers have been published in [2,16,17]. With the EMIs, a modular exchangeability and extendability is created in the MRS [1].

13.2.1.2 Final design: SherpaTT

The rover system SherpaTT is depicted in Fig. 13.1b. This rover was tested in several field test campaigns for locomotive capabilities as well as autonomous control, see Section 13.4.1 for more details.

The rover has a total mass of approximately 200 kg; each leg as well as the manipulation arm are weighting approximately 26 kg. The rover is powered from two battery packs with a nominal voltage of 44.4 V with 10 Ah capacity each. Power source management is conducted by a central power management board which is able to switch automatically between the two internal batteries, an external power connection and the modular power bus established by the EMIs. Source switching takes place in case of voltage drop, empty batteries, or when a higher prioritized power source becomes available (e.g., when plugging in the external power supply).

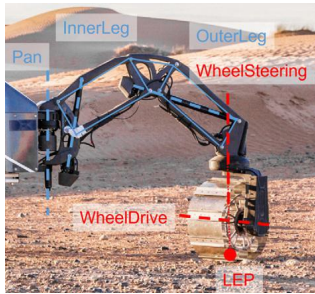
For mapping and navigation purposes a rotating lidar sensor is mounted on top of the rover. The sensor rotates with the first arm joint, hence allowing to compensate for occlusion by the arm. Additionally, a tiltable laser range finder is mounted on the front along with a wide-angle camera.² The front laser is mainly being used for manipulation tasks in front of the rover. The camera is used by human operators for situation awareness in remote operations.

As in the previous design of Sherpa, SherpaTT features six EMIs in total; four passive EMIs are mounted around the central manipulation tower, one active EMI is mounted at the bottom of the central body and one active EMI is used as end-effector for the manipulation arm.

13.2.1.3 SherpaTT: kinematic suspension system design

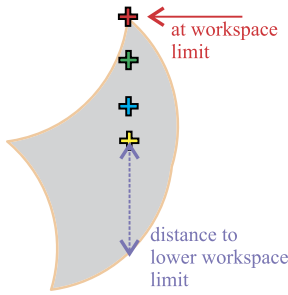
The suspension features five active DoF per leg, as indicated in Fig. 13.2a. The InnerLeg and OuterLeg joints are two serially connected parallelograms, with a knee-like structure connecting both. The introduction of the knee link enables internal movements: The rover's body can be moved

² In various projects, instead a dedicated sensor package was mounted on the front, for example, the “AvionicsBox” from the ADE-project, as shown in Fig. 13.4c.

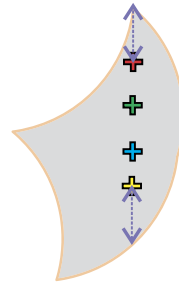


(a) SherpaTT's leg structure: Two serially connected parallelograms.

(b) 3D workspaces of all leg end points.



(c) Ground adaption (see following sections): one LEPs at limit, no further adaption possible with keeping the all wheels in ground contact.



(d) Shift all LEPs, keep relative distance.

Figure 13.2 SherpaTT's leg kinematics and workspace (top). Illustration of effect of body height shifting on available workspace for individual leg end points (bottom).

in 6 DoF with all four wheels being stationary on the ground. Both parallelograms in the leg structure have the same baseline of 400 mm.

The WheelSteering axis is placed directly over the Leg End Point (LEP) of the leg. This requires a higher torque for steering the wheel, when compared to having an off-axis wheel, which can support the steering by rolling. However, orientation of the wheel and position of the LEP are decoupled by this design, facilitating locomotion control of the system. A Force-Torque Sensor (FTS) is mounted at the flange of each wheel to enable precise load balancing between the wheels as well as ground contact loss detection. With active load balancing comes the opportunity of active wheel unloading for steering support and hence a relaxed torque requirement for the steering actuator [14]. In SherpaTT, this process is called *SteeringSupport* and is described further in Section 13.3.3.

13.2.2 Electronic design

As mentioned before, SherpaTT is powered by a voltage source of 44.4 V (nominal). This voltage could either come from one of the two battery packs, the modular power bus established by the EMIs or from an external source. Switching between these different sources is done by the Power-Management board to enable a continuous power supply at all times, with additional LiPo-watchers monitoring the voltage of each battery pack. For mid-level and high-level control a standard PC is used which is connected via Ethernet to different sensors like lidar or camera, and also to a communication bridge that handles the communication with all the joints and sensors within the legs and the manipulator. This low-level communication uses the NDLCOM protocol [18] with a daisy-chained LVDS-based communication, which is also used in most robots developed at DFKI. For motor commutation and control, an FPGA-based electronics is used and for low-level sensor (pre-)processing microcontroller boards are employed. A special feature of SherpaTT is the use of the different EMIs (in the manipulator, at the bottom and in the EMI bays). These EMIs have either their own STM32-based electronics (active ones in manipulator and bottom) or are connected to a Module-Management board (passive ones), which handle all the low-level communication and control of the EMIs. Additionally, all the EMIs offer an Ethernet connection to Sherpa's internal network. In case there was a need of additional sensors or computers (e.g., from project partners), these could be easily mounted on one of the EMIs or on special rails around Sherpa's body.

13.2.3 Software design

The control and, correspondingly, the software design can be grouped into reactive and deliberative layers. The reactive layer reacts directly to the sensor input by sending the joint commands. The deliberative layer, on the other hand, accumulates sensor data to build detailed information of the environment and then plans actions for the immediate future and longer term. For the low-level control of the joints and data acquisition of sensors, Field Programmable Gate Array (FPGA)-based electronics are used to implement the controllers and to communicate with the higher-level computers.

The reactive layer forms the mid-level control and is described in Section 13.3.3. Similarly, the deliberative layer forms the high-level control and described in Section 13.3.4. The mid-level and high-level control software is based on ROCK (Robot Construction Kit) framework, which is described [19]. The low-level control is described in Section 13.3.2.

13.3 Modeling and control

13.3.1 Modeling

The leg of SherpaTT is modeled as a serial kinematic where the parallel kinematic is abstracted as serial joints. A conversion between the linear actuator length pushing the parallelogram and the virtual rotational joint is performed locally in the joint actuator's control electronic, see Section 13.3.2. Once the leg is considered as a serial kinematic, well known modeling procedures can be employed.

13.3.1.1 Inverse kinematics

The inverse kinematic of the leg is computed by considering only the first three joints (pan, inner-leg, and outer-leg). WheelSteering and WheelDrive do not contribute to the position of LEP. In order to compute the inverse kinematic, the LEP coordinate system is converted to cylindrical coordinates with respect to the leg coordinate system placed in the pan joint.

The objective is to compute the joint angles α_i , β_i , and γ_i , corresponding to the pan, inner-leg, and outer-leg joints. Let the desired cylindrical coordinate for LEP be $[r, h, \theta]^T$, corresponding to radius, height, and angle, respectively. Since the coordinate origin coincides with pan joint $\alpha_i = \theta$. The computation for β_i and γ_i given r and h is as explained below. Let

$$t_0 = \frac{r^2 + h^2 + L_{il}^2 - L_{ol}^2}{2L_{il}}, \quad (13.1)$$

$$t_1 = r^2 + h^2 - t_0^2, \quad (13.2)$$

$$t_2 = \frac{r^2 + h^2 - L_{il}^2 - L_{ol}^2}{2L_{il}L_{ol}}, \quad (13.3)$$

where L_{il} and L_{ol} are link lengths for the inner-leg and outer-leg, respectively.

A solution exists only if $t_1 \geq 0$, else the solution is imaginary. For real solutions, the following solutions are defined for inner-leg

$$\beta_i = \frac{2 \arctan(h \pm \sqrt{t_1})}{r + t_0} \quad (13.4)$$

and the following for the outer-leg

$$\gamma_i = \beta_i \pm \arccos(t_2). \quad (13.5)$$

There are multiple solutions and the correct solution is chosen by finding the one that gives the correct forward kinematics.

13.3.2 Low-level control

For the low-level control of SherpaTT's joints, an FPGA-based electronics is used, which could also be found in a multitude of other robots at DFKI. To minimize the cabling effort, these electronics are mounted close to the associated motors, therefore only power and communication cables are necessary between the joints. In the SherpaTT system we have two different types of joints: 1. The "normal" revolute joints and 2. the linear joints in the inner and outer leg which drive the parallel structures described in Section 13.2.1.3. To be able to control all these joints with angular position and/or velocities, the kinematical model of the parallel structure is computed locally in the corresponding actuator electronics.

For sensor (pre-)processing, like the force-torque sensors at the wheels, microcontroller-based boards are used, which are fully integrated in the overall communication and control structure of the Sherpa rovers.

13.3.3 Mid-level control

The mid-level control software of SherpaTT is called Motion Control System (MCS). The general structure of the MCS is shown in Fig. 13.3. This section gives an overview and some details of the different modules. A more detailed discussion is provided in [4,14].

The MCS provides a reactive layer of controllers that are necessary for the proper functioning of the higher level software (e.g., autonomy modules). The main functionalities of MCS are

- (i) *Driving* for moving the platform in a desired direction using the wheels.
- (ii) *Movement of legs* to change the footprint and/or to react to sloping terrain.
- (iii) *Assistive controllers* that reduce the complexity for the higher level and also ensures the safety of the robot.

The inputs to this layer are the high-level commands (e.g., platform velocity) and sensor data. The outputs are the commands for the joints.

The driving command is called *MotionCommand 3D* and consists of the desired robot velocities in the x and y direction, along with the yaw velocities. These commands are transformed to desired steering and wheel commands by the steering computation and wheel speed computation

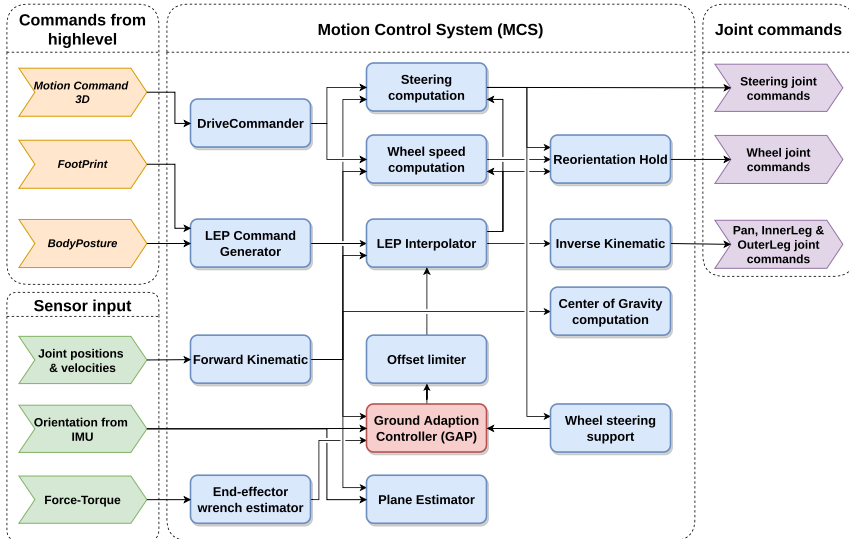


Figure 13.3 Motion control system.

modules respectively. A *reorientation hold module* is used to coordinate the movement if any of the current steering joints is not aligned with the desired command.

The footprint of the robot can be commanded either directly using the *Footprint command*, which gives individual pose for each leg, or using the *BodyPosture command*, which gives the pose for the robot chassis. These two commands are merged together in the *LEP command generator* and then passed on the *LEP interpolator*, which computes a cylindrically interpolated trajectory for each leg. The trajectory is used by the *inverse kinematic module* to compute the desired joint positions.

The central component of the MCS is the *Ground Adaption Process (GAP)*, which itself is composed of various sub-modules. All modules of GAP are writing x, y, z -offsets to the leg-endpoints in order to adapt the wheel position according to the current terrain and chosen locomotion mode. The result is a (reactive) movement of the leg end point around its set-point. The set-point of the wheel is originating from the footprint and body posture command inputs. The main modules of GAP, which are described in more detail in the following paragraphs, are

- *Force Leveling Control (FLC)* – responsible for keeping wheels in ground contact with desired load share;

- *Roll-Pitch Adaption (RPA)* – responsible to keep body’s roll and pitch in desired value relative to gravity, independent of terrain;
- *Wheel Steering Support (WSS)* – unloads wheels in case of WheelSteering actuator torque stall;
- *Body Height Control (BHC)* – adapts the body height in order to maximize the usable workspace of the legs.

13.3.3.1 FLC: force-leveling control

In FLC, the forces measured at the wheels are used to keep all wheels in ground contact and, moreover, keep the desired ground contact force. In the basic case, the desired contact force is that resulting from the position of the center of gravity within the rover’s support polygon.

Two different versions of the controller have been implemented and tested. The first one, as detailed in [14], uses separate controllers for each leg to move it vertically and keep it close to an estimated ideal force. However, this controller tended to oscillate [4].

The second version of FLC is called cross-leveling, it is simplified as it uses only one PID controller, while providing higher responsiveness and accuracy at the same time. The principle idea is to add the forces on diagonally opposite legs into one value and try to keep the difference between the cross forces to a minimum. The complete computation is detailed below.

As a first step, the reference forces for each wheel/diagonal need to be calculated. Hence, the current footprint is considered and the center of mass of the robot, laying within the footprint, determines the current contact forces $f_{z,ref,i}$ under the assumption of static equilibrium. The wheels on the front-left, front-right, rear-left, and rear-right legs are represented with numbers 0, 1, 2, and 3, respectively. Only 2D positions of the LEPs $((x_i \ y_i)^T)$ as well as of the Center of Gravity (CoG) $((x_c \ y_c)^T)$ are required for calculating the reference forces. A vector $t = (0 \ 0 \ mg)^T$ is constructed, containing zero-moments around x- and y-axis and gravitational force, where m is the mass of the robot and g is the acceleration due to gravity. The vector of expected reaction forces at each LEP is the vector of reference forces $f_{z,ref} = (f_{z,ref,0} \ f_{z,ref,1} \ f_{z,ref,2} \ f_{z,ref,3})^T$. Eq. (13.6) shows the resulting underdetermined equation system.

$$\mathbf{A} \cdot \mathbf{f}_{z,ref} = \mathbf{t}. \quad (13.6)$$

The matrix \mathbf{A} is defined as provided in Eq. (13.7).

$$\mathbf{A} = \begin{pmatrix} x_0 - x_c & x_1 - x_c & x_2 - x_c & x_3 - x_c \\ y_0 - y_c & y_1 - y_c & y_2 - y_c & y_3 - y_c \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (13.7)$$

Constructing the Moore–Pensrose pseudoinverse $\mathbf{A}^+ = \mathbf{A}^T \cdot (\mathbf{A} \cdot \mathbf{A}^T)^{-1}$ allows to calculate $\mathbf{f}_{z,ref}$ according to Eq. (13.8).

$$\mathbf{f}_{z,ref} = \mathbf{A}^+ \cdot \mathbf{t}. \quad (13.8)$$

In each time step of MCS execution, $\mathbf{f}_{z,ref}$ is recalculated, updating the reference values according to the current footprint and CoG location within the support polygon.

The actual forces $f_{z,act,i}$ for each leg are the values received from the force-torque sensors. The cross forces for front-left and rear-right wheels, both reference and actual, are computed as

$$f_{z,ref}^c = f_{z,ref,0} + f_{z,ref,2}, \quad (13.9)$$

$$f_{z,act}^c = f_{z,act,0} + f_{z,act,2}. \quad (13.10)$$

The cross forces, $f_{z,ref}^c$ as reference and $f_{z,act}^c$ as actual, are used to create PID controller which outputs an offset. This offset is added to the LEP offset commands for front-left and rear-right legs. The offset is removed from front-right and rear-left leg commands.

It is noteworthy that the FLC only adapts the forces according to the current state of the robot. If the center of mass is close to the rear legs due to driving on a slope, the controller will act accordingly. Finding an optimal distribution on the wheels, say the same force on each wheel, would be the task of an according module manipulating the position of the CoG within the support polygon.

13.3.3.2 RPA: roll-pitch adaption

In case of a rover with passive suspension, e.g., a rocker-bogie, the main body's roll and pitch angle are influenced by the terrain and cannot be actively controlled. With the internal mobilities of SherpaTT's suspension system, tracking of ground contact forces as described in Section 13.3.3.1 is possible in parallel to roll-pitch control of the central body. In [14] it was demonstrated that both control objectives do not negatively influence each other when being active in parallel.

Benefits of active roll and pitch control are the ability to (i) change the position of the rover's CoG within the support polygon, (ii) support specific sensors or solar panels with orientation restrictions mounted on the rover platform [20], (iii) facilitate payload deployment with SherpaTT, and (iv) increase the usable workspace for improved ground adaption, as highlighted in Section 13.3.3.4.

The RPA module in SherpaTT's ground adaption process takes the Inertia Measurement Unit (IMU) readings and compares it against the commanded values of roll (θ_r) and pitch (θ_p). Roll and pitch errors are combined into angle-axis form [21]. The required offset for each wheel is calculated using the distance $d_i = \|\mathbf{d}_i\|$ of LEP i to the rotation axis (in the xy-plane). Depending on the sign of Θ_e and the position of the LEP with respect to \mathbf{e} , the generated offset is then positive or negative.

13.3.3.3 WSS: WheelSteering support

In planetary exploration, situations might occur in which a wheel is stuck between some rocks, dug deep into the soil or similar, so that the wheel cannot be steered any more. The WSS module of SherpaTT's GAP is a module calculating new reference forces for the wheels in order to unload individual wheels. As such, the WSS is tightly connected to the FLC module described in Section 13.3.3.1. The support of the WheelSteering joints is only possible in conjunction with an active force leveling control.

Different triggers for a WheelSteering support event are possible, including

- the electrical current of a WheelSteering joint is greater than the predefined maximum $I_{WS,max}$,
- the difference between actual steering angle φ_i and reference $\varphi_{ref,i}$ is greater than the predefined maximum $\Delta\varphi_{max}$ while the joint is not moving ($\dot{\varphi}_i \leq \dot{\varphi}_{min}$), and
- time in system state `reorientation hold` (see Section 13.3.3.5) is longer than predefined maximum $t_{ro,max}$.

A current limiter implemented on the low-level joint control prohibits exceeding $I_{WS,max}$. In effect this current limitation acts as a torque limitation for each joint. Hence, the second trigger becomes active, once a WheelSteering joint cannot act against the resistance of the ground due to its torque limit. The timed trigger is a backup in case of errors in sensor readings of the other triggers. This time-based trigger initiates a subsequent unloading of all four wheels, while the other triggers initiate unloading of a single wheel.

Unloading of wheels is conducted by manipulating the reference forces for the FLC module. The reference force for the wheel j to be unloaded is reduced by a predefined value f_s , which is shifted to the remaining three wheels, such that

$$f_{z,ref,j}^{WSS} = f_{z,ref,j}^{IFE} - f_s \quad j: \text{wheel with steering support} \quad (13.11)$$

$$f_{z,ref,i}^{WSS} = f_{z,ref,i}^{IFE} + \frac{f_s}{3} \quad i \neq j \quad (13.12)$$

In case of cross leveling, this has the effect of generating a weak contact axis. The wheel in need of WSS is then placed on the weak contact axis to be able to align the wheel correctly.

13.3.3.4 BHC: body height control

All modules of the GAP are writing offsets individually to the wheel positions. The final output might be, that all offsets have the same sign, effectively resulting in a change of body height with possible negative effect either on ground clearance or stability due to a higher center of gravity.

Furthermore, the usable workspace for ground adaption is reduced when all offsets have the same sign, since the leg end points are moved towards the boundaries of the workspace. Fig. 13.2c illustrates the case, where all LEPs are close to the lower workspace limit, with one LEP being at the limit. No further stretching of this leg is possible, resulting in ground contact loss or improper roll-pitch adaption.

The BHC module is used to keep a desired body height and to maximize the usable workspace for the other ground adaption modules by shifting all LEPs such that the distance from the highest LEP to the upper workspace limit and that from the lowest LEP to the lower workspace limit are similar, while keeping the relative distances between the LEPs. The base offset $o_{fix}^{BHC} = -b_z$ defines the center of all LEPs and defines the desired body height. For maximizing the usable workspace, all LEPs are shifted to be around this value. Fig. 13.2d shows the effect of shifting the LEPs with $o_{fix}^{BHC} = 0$, i.e., moving the LEPs around Pose A .

$$o^{BHC} = K_b \cdot \frac{(\min_i(d_{up,i}) - \min_i(d_{down,i}))}{2} + o_{fix}^{BHC} \quad \begin{array}{l} o^{BHC} \text{ same offset for all LEPs} \\ K_b: \text{proportional gain BHC} \end{array} \quad (13.13)$$

In a further extension of this approach, roll and pitch of the body can be manipulated to further increase the workspace for the LEPs. This can,

of course, only be conducted, when the roll and pitch of the body are not required at certain values for current rover operations. An automatic roll-pitch command manipulation is currently not implemented in the BHC module, however, an experiment with manual pitch control in steep slopes showed to increase the workspace and, furthermore, indicates an improvement of force distribution between the wheels, as documented in [14].

13.3.3.5 Further MCS modules

Further modules, not explicitly illustrated above, are part of the MCS. These modules are briefly highlighted in this section.

A *Force Fuse* module is used to monitor internal stress of the suspension system. The xy-vector of forces at a wheel is monitored, ideally no xy-force is present when the system is on flat ground. When a threshold is exceeded, all motions of the robot are stopped for unloading. Currently, this module acts like a fuse: once triggered, manual intervention is required.

A so-called *Reorientation Hold* is triggered, when a sudden change in the velocity command requires the wheels to substantially change their orientation, e.g., in order to follow a narrow arc turn after a straight line drive. Such a command sequence results in a jump in the reference angle for a *WheelSteering* joint, hence, the reference angle and the actual angle have a relative high delta. In order to avoid imprecise locomotion, internal stress in the mechanical structure and possible wheel slippage, such a high delta triggers the reorientation hold. The robot stops driving until all wheels are oriented properly to follow the currently given trajectory.

An additional *Radius Input* module is implemented to facilitate the interfacing with high-level processes. The module takes a footprint radius r_{fp} as input and makes use of the *FootPrint* command \mathbf{g} to position all LEPs on the perimeter of a circle with r_{fp} around the center of the robot's body. This allows to change the footprint size with one variable for autonomous traverse through narrow passages.

A *Ground Plane Estimator* module is implemented to assess the terrain slope below the robot. This module makes use of the current roll and pitch angle of the body and the current leg end point positions. In a least square approach, a plane is fitted through all ground contact points and related to the current roll and pitch angle. This way, a rough estimate of the instantaneous slope below the robot is possible. In the future, this can be used to further adapt the robot's pose reactively to the terrain.

13.3.4 High-level control

The high-level control modules of the robot take care of Self-Localization and Mapping (SLAM), path planning and execution of the robot base, and trajectory planning and execution of the manipulator. These functionalities are then used to create missions with more autonomy features.

The SLAM is a custom implementation, Slam3d, which is open-source and can be found in [22]. The library has interfacing for both ROS and ROCK. Slam3D provides a frontend for graph based SLAM in 3D space. Different solvers for the graph optimization can be implemented as back-end by providing the interface. Slam3D can use point-cloud data, Global Positioning System (GPS) and wheel odometry data as constraints for the optimization of the graph.

A path planner generates a path from a given pose to a target pose based on the obstacles and terrain map generated by the SLAM module. The path planning is a custom implementation based on [23] in ROCK. It uses search-based planning on using graphs in discrete environments. A set of motion primitives is generated a priori based on kinematic constraints of the robot. These primitives are then combined to form a graph after taking collision and validity of the state into account. The final path is computed using the ARA* algorithm developed by Likhachev in [24]. The generated path is followed using the trajectory tracking algorithm described in [25].

The ROCK motion planning for manipulator, as described in [26], is a custom implementation based on Open Motion Planning Library ([27]). Additionally, it also interfaces to other optimization-based planners as well. The kinematic library and collision computations are abstracted such that other algorithms can be easily interfaced.

13.4 Conclusion and outlook

Due to the demonstrated capabilities of the Sherpa rover family, it was and will be deployed in several projects and mission demonstration scenarios. The scenarios include space exploration, the domain it was originally developed for, but also terrestrial applications are conducted and, for the sub-sea version, underwater use cases. This includes further search and rescue scenarios, for example, in the context of decontamination and deconstruction, agricultural settings as well as forestry scenarios. In the following paragraphs, we highlight some of the success stories of the robot as well as design limitations and future work.



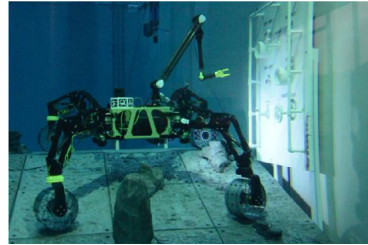
(a) SherpaTT in Utah, USA 2016



(b) SherpaTT in Morocco 2018



(c) SherpaTT in Northern Germany 2021



(d) SherpaUW in the underwater testbed at DFKI

Figure 13.4 Sherpa rovers in different applications.

13.4.1 Success stories

In this chapter, we present some of the projects and field experiments in which one of the Sherpa rovers was involved.

13.4.1.1 USA 2016

At the end of 2016, SherpaTT had its first major field test,³ Fig. 13.4a shows the rover in the desert of Utah during a locomotion test. The field test was conducted under German national funding with grant number 50RA1621. Apart from individual system tests with focus on locomotion capabilities, the results of which are detailed in the next paragraph, a multi-robot cooperative sample return mission was emulated, including a ground control station in Bremen, Germany [28,29].

The experiments on locomotion performance of SherpaTT have been published in [14], and [21] and are summarized and compared in [4]. According to the experimental evaluation, active ground adaptation is able to limit the ground contact force errors below 2% of the rover's gravitational force. The absolute error of roll and pitch control can be kept below 1°

³ Video summary of the Utah field test is available at <https://www.youtube.com/watch?v=pvKIzldni68>.

in all tested scenarios, encompassing high-frequency obstacles in laboratory experiments as well as high slope settings in real world field tests. The Root Mean Square (RMS) error is reduced to 0.2° or below in all tested terrains and footprints. With these results, the precision of body orientation control is demonstrated in rough natural terrain. Such precise body orientation control helps in multi-robot scenarios to deploy modular payloads and for orientation control of sensors that might be fixed to the body.

It is obvious that an active suspension has actuators which need to be powered. Yet, the conducted experiments in Mars analogue terrain showed that the developed suspension system requires only around 3%–6% of the total rover power to keep all wheels in ground contact. The mean absolute power for the suspension actuation can be as low as 7 W and is around 12 W in most cases for these experiment sets. Compared to the potential for fault recovery, and general flexibility, we consider this power overhead tolerable.

13.4.1.2 Morocco 2018

The Morocco field testing was conducted in the frame of the EU project FACILITATORS (GA 730068) with partners from across Europe.⁴ Focus of the field tests was to accomplish a long traverse (≥ 1 km) in ≤ 8 h autonomously [30]. Fig. 13.4b shows the rover during a manually controlled traverse on a rocky slope.

The long-traverse mission, which had to be realized in the analog simulations, required significant energy-autonomy of SherpaTT. A battery extension of SherpaTT would have been possible, yet the time margin for this solution was considered to be too small. Hence, a range extender in form of a small fuel-powered generator was added to the rover. Although this technology is not compatible with a real mission to Moon or Mars, DFKI decided to evaluate the feasibility of this solution in order to extend the operational time and range of the robot.

Overall, the field trials were very successful. The SherpaTT had very little down-time and the DFKI Sherpa support crew was able to resolve any hardware or software problem very quickly in the field. A drone flight prior to the tests was used to generate a coarse map of the test field, comparable to a satellite coverage of a prospective landing area on Mars. The employed ESROCOS/ERGO software [31] for autonomous navigation

⁴ Video summary of the Morocco field test is available at <https://www.youtube.com/watch?v=-zqve9baOzM&t=0s>.

worked fine and, guided by this software, the robot was able to achieve the 1 km autonomous long-distance run as planned.

13.4.1.3 Germany 2021

The field test for very long traverses with autonomous decision making (ADE) for rescheduling and opportunistic science was planned to take place in a Mars analogue setting on Fuerteventura in 2020. The project ADE was conducted under EU grant GA 821988. The main challenge for ADE was to demonstrate the techniques needed to realize a planetary rover system with very long traverse capabilities (kilometers per sol) by independently taking the decisions required to progress, reduce risks and seize opportunities of data collection. The rover was required to travel independently from a starting point (e.g., a lander) towards an end point (for example a cache of sample), perform independent opportunistic science on the way and return to the lander with the acquired soil sample. Due to the coronavirus pandemic and the accompanying travel restrictions, the test was postponed to 2021 and eventually took place close to the SherpaTT laboratories in northern Germany,⁵ with the European partners being connected to the rover and the site via internet. Despite the hindrances imposed by the pandemic, the testing was successful. The rover operated under varying autonomy modes and drove considerable distances autonomously, the longest run was about 500 m in less than 3 hours. Results of the field trials are detailed in [32].

13.4.1.4 ROBDEKON

The SherpaTT robot is part of the project ROBDEKON⁶ [33], where a team of robots is deployed to clean up sites that are hostile or dangerous for humans. One of the scenarios includes retrieving waste barrels from a disposal site by operating a robotic excavator remotely.

The task for SherpaTT in this scenario is to build a map of the environment and to drive between different locations in the map autonomously, searching for barrels. Once a barrel is found, the operator uses the suite of sensors attached to the robot, including a high-definition camera at the end-effector, to analyze the barrel closely to assess the condition. Once a

⁵ Video summary of field test in Northern Germany available at https://www.youtube.com/watch?v=BQqBR_5SwPM.

⁶ Video summary of the ROBDEKON project is available at <https://www.youtube.com/watch?v=Wnw1ihHqFgQ>.

barrel is determined to be viable for retrieval, the excavator robot, ARTER, is deployed to retrieve the barrel.

13.4.1.5 SherpaUW

To show that the technologies developed with the Sherpa and SherpaTT rovers are also transferable and applicable in sub-sea scenarios, the underwater version SherpaUW was built (see Fig. 13.4d). This robot was successfully tested in the underwater testbed at DFKI Bremen in a scenario where it formed a team with the more traditional autonomous underwater vehicle Leng.⁷ It was also demonstrated that the active suspension system is advantageous when driving on unstructured terrain at the sea floor. Since it requires no extra energy to maintain its posture and position compared to traditional underwater vehicles, SherpaUW is well suited for repeated and precise sampling or surveillance tasks. Possible areas of application include surveillance and maintenance for underwater infrastructure, exploration and resource utilization, and research on maritime deep sea life.

13.4.2 Design limitations and lessons learned

Even though the rovers already demonstrated their exceptional locomotion capabilities, further developments of the control system need to be undergone to fully exploit the capabilities of the locomotive system's hardware. One example is the force leveling control, which currently adapts the rover to the expected forces at each wheel resulting from terrain inclination and footprint configuration. These reference forces should be manipulated in an additional reactive module in order to evenly distribute the load to all wheels. On slopes, it might be beneficial to shift even more weight to the wheels higher in the slope than to those on the lower end of the slope. This would lie in the responsibility of a reactive force shifter that takes the current incline of the terrain as input.

The so-called reorientation holds of the system can be avoided by moving the wheel on a trajectory that ensures a smooth position and velocity trajectory for the WheelSteering joint. In the OG10 ADE project [32], the behavior was considered in the high-level path planning, yet it should be handled in the middleware as it is a rover-specific behavior, which should be handled independently of high-level control.

⁷ Video of SherpaUW in the underwater testbed at DFKI in Bremen <https://www.youtube.com/watch?v=R9LRd7jGiH0>.

So far, all reactive offsets are written to the z-component of the leg endpoint only. A virtual spring module shall make use of all three dimensions of the LEP offsets: When a force acts on the wheel, for example resulting from a rock the wheel is driving onto, a force-proportional displacement of the wheel is conducted using the offsets, which models the behavior of a mechanical spring. Improved obstacle negotiation is aspired using such a module.

Alternative drive modes to the OmniDrive, employed for the experiments presented in this chapter, are being developed in a currently running project. The comparison of different drive modes in terms of terrain performance would then allow to establish a metric for locomotion mode selection depending on terrain types.

13.4.3 Future work

As presented in this chapter, the Sherpa rover family was originally developed in the context of future space missions, but we showed that it could be deployed in other scenarios, like search and rescue missions and in case of the waterproof version SherpaUW, also in underwater scenarios. One of the next projects where SherpaTT will be used in the context of an exploration mission of planetary surfaces is the CoRob-X project.⁸ Here the rover will again be part of a robotic team with other autonomous and cooperating robots which should demonstrate how to access very hard-to-reach areas on planetary surfaces and achieve a science return that is so impossible for a single robot. Another use case that will be evaluated in the next years is in agricultural applications. For this purpose, SherpaTT is currently used in the project RoBivaL,⁹ in which its locomotion capabilities will be compared to other robotic systems under agricultural conditions.

As the SherpaTT platform is a research platform, it is constantly being improved by the experiences gained from experiments and field tests. For example, because SherpaTT was loaded with a large amounts of extra payload, one of the levers to which the wheels are mounted broke in one of the last projects. Therefore it was re-engineered and adapted to the new requirements and load cases.

One big downside of the actual rover design of SherpaTT is that its body was not designed waterproof. This is a serious concern, especially when experiments and field tests take place in rainy regions like northern

⁸ <https://robotik.dfki-bremen.de/en/research/projects/corob-x/>.

⁹ <https://robotik.dfki-bremen.de/en/research/projects/robival/>.

Germany. For this reason all parts – especially the rover body – will be redesigned and made waterproof for increased outdoor capabilities in the next years.

References

- [1] T.M. Roehr, F. Cordes, F. Kirchner, Reconfigurable integrated multirobot exploration system (RIMRES): heterogeneous modular reconfigurable robots for space exploration, *Journal of Field Robotics Special Issue on Space Robotics, Part 2* 31 (2013) 3–34, <https://doi.org/10.1002/rob.21477>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21477>.
- [2] W. Brinkmann, F. Cordes, T.M. Roehr, L. Christensen, T. Stark, R.U. Sonsalla, R. Szczuka, N.A. Mulsow, D. Kuehn, Modular payload-items for payload-assembly and system enhancement for future planetary missions, in: 2018 IEEE Aerospace Conference (AeroConf'18).
- [3] R. Sonsalla, F. Cordes, L. Christensen, S. Planthaber, J. Albiez, I. Scholz, F. Kirchner, Towards a heterogeneous modular robotic team in a logistic chain for extraterrestrial exploration, in: 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'14), Montreal, Canada, 2014, https://www.researchgate.net/publication/265085869_Towards_a_Heterogeneous_Modular_Robotic_Team_in_a_Logistic_Chain_for_Extraterrestrial_Exploration.
- [4] F. Cordes, Design and experimental evaluation of a hybrid wheeled-leg exploration rover in the context of multi-robot systems, phdthesis, University of Bremen, Bremen, Germany, Dec. 2018, <http://nbn-resolving.de/urn:nbn:de:gbv:46-00106941-11>.
- [5] R. Sonsalla, J.B. Akpo, F. Kirchner, Coyote III: development of a modular and highly mobile micro rover, in: 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA'15), 2015.
- [6] R. Volpe, Rover technology development and mission infusion beyond MER, in: 2005 IEEE Aerospace Conference (IAC'05), 2005, pp. 971–981, <https://doi.org/10.1109/AERO.2005.1559388>, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1559388&newsearch=true&queryText=richard%20volpe>.
- [7] R. Lindemann, C. Voorhees, Mars exploration rover mobility assembly design, test and performance, in: 2005 IEEE International Conference on Systems, Man and Cybernetics (SMC'05), vol. 1, 2005, pp. 450–455, <https://doi.org/10.1109/ICSMC.2005.1571187>.
- [8] M. Heverly, J. Matthews, J. Lin, D. Fuller, M. Maimone, J. Biesiadecki, J. Leichty, Traverse performance characterization for the Mars science laboratory rover, *Journal of Field Robotics* 30 (6) (2013) 835–846, <https://doi.org/10.1002/rob.21481>.
- [9] P.S. Schenker, T.L. Huntsberger, P. Pirjanian, E. Baumgartner, H. Aghazarian, A. Trebi-ollennu, P.C. Leger, Y. Cheng, P.G. Backes, E.W. Tunstel, J. Propulsion, L.S. Dubowsky, Robotic automation for space: planetary surface exploration, terrain-adaptive mobility, and multi-robot cooperative tasks, in: David P. Casasent, Ernest L. Hall (Eds.), *Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision Conference*, vol. 4572, Boston, MA, USA, 2001, pp. 12–28, <https://doi.org/10.1117/12.444181>, 2001.
- [10] T. Huntsberger, A. Stroupe, H. Aghazarian, M. Garrett, P. Younse, M. Powell, TRESSA: teamed robots for exploration and science on steep areas, *Journal of Field Robotics* 24 (11–12) (2007) 1015–1031, <https://doi.org/10.1002/rob.20219>.
- [11] I.A. Nesnas, J.B. Matthews, P. Abad-Manterola, J.W. Burdick, J.A. Edlund, J.C. Morrison, R.D. Peters, M.M. Tanner, R.N. Miyake, B.S. Solish, R.C. Anderson, Axel

- and DuAxel rovers for the sustainable exploration of extreme terrains, *Journal of Field Robotics* 29 (2012) 663–685, <https://doi.org/10.1002/rob.21407>.
- [12] F. Cordes, C. Oekermann, A. Babu, D. Kuehn, T. Stark, F. Kirchner, An active suspension system for a planetary rover, in: 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'14), Montreal, Canada, 2014.
 - [13] D. Spenneberg, F. Kirchner, *Climbing and Walking Robots Towards New Applications*, I-Tech Education and Publishing, 2007, Ch. The Bio-Inspired SCORPION Robot: Design, Control & Lessons Learned, http://sciyo.com/articles/show/title/the_bio-inspired_scorpion_robot__design_control__lessons_learned.
 - [14] F. Cordes, F. Kirchner, A. Babu, Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain, *Journal of Field Robotics* 35 (7) (2018) 1149–1181, <https://doi.org/10.1002/rob.21808>, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21808>.
 - [15] T.M. Roehr, F. Cordes, F. Kirchner, RIMRES: a project summary, in: 2013 IEEE Conference on Robotics and Automation (ICRA'13), Workshop Proceedings, Karlsruhe, Germany, 2013.
 - [16] Z. Wang, F. Cordes, A. Dettmann, R. Szczuka, Evaluation of a power management system for heterogeneous modules in self-reconfigurable multi-module systems, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11).
 - [17] A. Dettmann, Z. Wang, W. Wenzel, F. Cordes, F. Kirchner, Heterogeneous modules with a homogeneous electromechanical interface in multi-module systems for space exploration, in: 2011 IEEE International Conference on Robotics and Automation (ICRA'11).
 - [18] M. Zenzes, P. Kampmann, M. Schilling, T. Stark, Simple protocol for heterogeneous embedded communication networks, in: Proceedings of the Embedded World Exhibition and Conference, 2016.
 - [19] S. Joyeux, J. Albiez, Robot development: from components to systems, in: 6th National Conference on Control Architectures of Robots, INRIA Grenoble Rhône-Alpes, Grenoble, France, 2011, p. 15, <https://hal.inria.fr/inria-00599679>.
 - [20] G. Labrèche, F. Cordes, Using a rover's active suspension system as a 2-axis solar tracker mechanism, in: 15th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'20), 2020, <https://robotik.dfki-bremen.de/en/research/publications/11155/>.
 - [21] F. Cordes, A. Babu, F. Kirchner, Static force distribution and orientation control for a rover with an actively articulated suspension system, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017), 2017, <https://github.com/dfki-ric/slam3d>.
 - [22] N. Limpert, S. Schiffer, A. Ferrein, A local planner for Ackermann-driven vehicles in ROS SBPL, in: 2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2015, pp. 172–177, <https://doi.org/10.1109/RoboMech.2015.7359518>.
 - [23] M. Likhachev, G. Gordon, S. Thrun, ARA*: Anytime a* with provable bounds on sub-optimality, in: Conference on Neural Information Processing Systems (NIPS) 2003.
 - [24] A. Micaelli, C. Samson, Trajectory tracking for unicycle-type and two-steering-wheels mobile robots, Research Report RR-2097, INRIA, 1993, <https://hal.inria.fr/inria-00074575>.
 - [25] B. Asadi, S. Natarajan, Motion Planning for Manipulators, in: Proceedings of the RIC Project Day “Framework & Standardization” and “Manipulation & Control”, vol. 14-03, 19.6.2014, Selbstverlag, Bremen, 2014, pp. 120–121.

- [27] I.A. Şucan, M. Moll, L.E. Kavraki, The open motion planning library, *IEEE Robotics & Automation Magazine* 19 (4) (2012) 72–82, <https://doi.org/10.1109/MRA.2012.2205651>, <https://ompl.kavrakilab.org>.
- [28] S. Planthaber, M. Maurus, B. Bongardt, M. Mallwitz, L.M.V. Benitez, L. Christensen, F. Cordes, R.U. Sonsalla, T. Stark, T.M. Roehr, Controlling a semi-autonomous robot team from a virtual environment, in: *Human Robot Interaction Conference (HRI'2017)*, Vienna, Austria, 2017.
- [29] R.U. Sonsalla, F. Cordes, L. Christensen, T.M. Roehr, T. Stark, S. Planthaber, M. Maurus, M. Mallwitz, E.A. Kirchner, Field testing of a cooperative multi-robot sample return mission in Mars analogue environment, in: *14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA'17)*, Leiden, The Netherlands, 2017, https://www.researchgate.net/publication/323279915_Field_testing_of_a_Cooperative_Multi-Robot_Sample_Return_Mission_in_Mars_Analogue_Environment.
- [30] W. Brinkmann, F. Cordes, C.E.S. Koch, M. Wirkus, R. Dominguez, A. Dettmann, T. Vögele, F. Kirchner, Space robotic systems and artificial intelligence in the context of the European space technology roadmap, in: *Proceedings of Space Tech Conferences*, 2019.
- [31] M.M. Arancón, G. Montano, M. Wirkus, K. Hoeffinger, D. Silveira, N. Tsiogkas, J. Hugues, H. Bruyninckx, I. Dragomir, A. Muhammad, ESROCOS: a robotic operating system for space and terrestrial applications, in: *14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2017)*, 2017.
- [32] J. Ocón, I. Dragomir, F. Cordes, R. Dominguez, R. Marc, V. Bissonnette, R. Viards, A.-C. Berthet, G. Reina, A. Ugenti, A. Coles, A. Coles, A. Green, R. Howard, L. Kunze, ADE: enhancing autonomy for future planetary robotic exploration, in: *72nd International Astronautical Congress (IAC)*, Dubai, United Arab Emirates, 2021.
- [33] J. Petereit, J. Beyerer, T. Asfour, S. Gentes, B. Hein, U.D. Hanebeck, F. Kirchner, R. Dillmann, H.H. Götting, M. Weiser, M. Gustmann, T. Eglöffstein, ROBDEKON: robotic systems for decontamination in hazardous environments, in: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2019, pp. 249–255, <https://doi.org/10.1109/SSRR.2019.8848969>.

CHAPTER 14

Recupera exoskeletons

Martin Mallwitz^{a,e}, Michael Maurus^{a,e}, Shivesh Kumar^a,
Mathias Trampler^a, Marc Tabie^a, Hendrik Wöhrle^{a,d}, Elsa A. Kirchner^c,
Henning Wiedemann^b, Heiner Peters^a, Christopher Schulz^a,
Kartik Chari^a, Ibrahim Tijjani^a, and Frank Kirchner^{a,b}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bWorking Group Robotics, University of Bremen, Bremen, Germany

^cInstitute of Medical Technology Systems, University of Duisburg-Essen, Duisburg, Germany

^dInstitute of Communication Technology, Dortmund University of Applied Sciences and Arts, Dortmund, Germany

14.1 Introduction

The development of exoskeletons as an active external support structure for the human body has made significant progress in recent decades thanks to miniaturization and digitalization. The decisive factor for acceptance and handling is mechanical transparency [1] for the user, in the sense of perceived restriction of movement. This applies to the application of the exoskeleton as a power assistance, as an input device for remote control or as a robotic rehabilitation device [2].

14.1.1 Motivation for series-parallel hybrid design

Building a robotic support structure around humans, which is not perceived as restrictive, places high demands on the ergonomics of the kinematic setup. Human joints are complex structures in terms of their mobility and can often only be approximated with rigid mechanics by combinations of rotational and linear joints. In addition to the bony skeletal structure, they consist of cartilage, capsules, tendons, ligaments and muscles. In order to adequately represent the shoulder joint, at least three, preferably five, rotational mechanical joints have to be combined. The joint axes often cannot be mapped directly, since the construction space is limited by the human body. Collisions of the mechanics with the human body must be prevented during motion. The use of parallel kinematics can separate the main pivot

^e The two authors contribute equally to the work.

point of a mechanism from the drive axis, thus making optimum use of the installation space. At the same time, the movement can be restricted in a defined way. This is achieved by positively guided movement of numerous links of the mechanism. The advantages of parallel kinematics include more freedom for installing the actuators, high stiffness and a good dynamic behavior, but it increases the number of moving components and the effort in modeling and controlling the system. Their application must therefore be justified. A combination of serial and parallel kinematics in many cases leads to very good overall results in terms of motion space, force transmission and installation space. This design philosophy forms the basis of RECUPERA exoskeletons, as shown in Fig. 14.1.

14.1.2 Application scenarios

The RECUPERA exoskeletons were developed as a training device for stroke rehabilitation [3] and modified for the purpose of teleoperation.

14.1.2.1 Rehabilitation scenario

Exoskeletons and here especially active exoskeletons can be used for the rehabilitation of patients, e.g., after a stroke [4]. Compared to the use of exoskeletons for the compensation of paralysis, especially after spinal cord damage, this application is not yet very well established. However, there are some studies showing that robot-assisted rehabilitation shows great effects [5]. Active exoskeletons are able to introduce forces into the patient's body and thus support them in their movements or make them possible in the first place. The support adapted to the specific requirements of the patient is called assist as needed [6].

14.1.2.2 Teleoperation scenario

Exoskeletons can also be applied for teleoperation. Teleoperation in general means the remote control of a robotic system [7,8]. In our case, we understand teleoperation as the remote control of a robotic system equipped with one or two manipulators. Teleoperation can be a very useful tool when an on-site presence is either too expensive (e.g., space exploration missions), too dangerous (e.g., search and rescue) or not applicable at all (e.g., deep sea). The exoskeleton used has to be easily controllable and mapped to the target system with respect to the application. For example, when teleoperating a humanoid robot, it is preferable to map not only the hand positions but additionally the elbow positions to improve operator immersion and

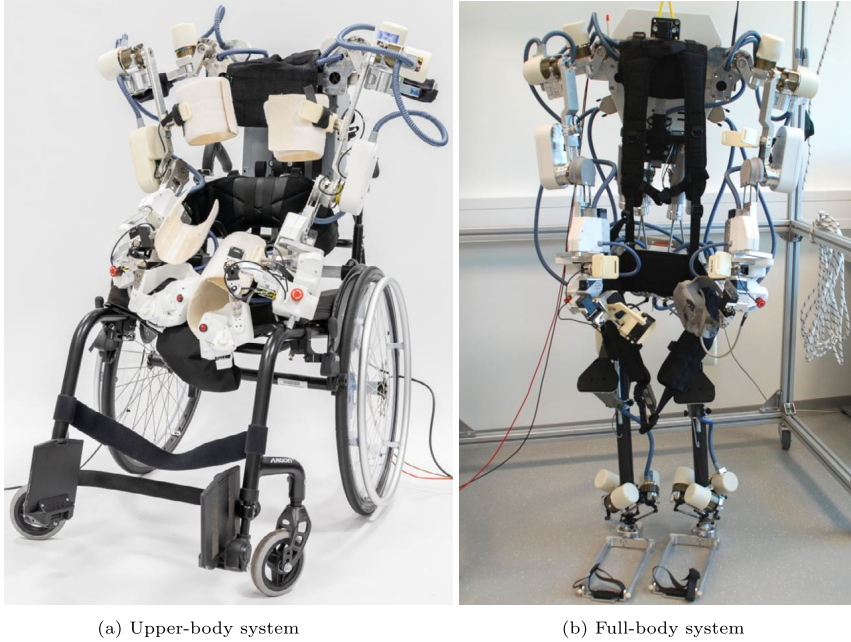


Figure 14.1 The RECUPERA upper-body exoskeleton is a part system of the full-body exoskeleton mounted on wheelchair.

mapping of the workspaces of the two different systems. Active exoskeletons also have the advantage of being a haptic interface [9]. This means that it is possible to give force feedback to the operator from forces occurring in the controlled target system and measured at force-torque sensors in order to provide better support, especially for remote manipulation tasks.

Organization

In this chapter, we present the RECUPERA full-body exoskeleton and the RECUPERA upper-body exoskeleton subsystem. Section 14.2 describes the mechatronic components and explains the modular concept in terms of mechanical sub-mechanisms and decentralized control units. Section 14.3 details the kinematic modeling, the dynamic model of the exoskeleton and the 3-stage control design and their relation to the application. The achieved results of the rehabilitation application and teleoperation are discussed in Section 14.4. Finally, a summary and outlook is provided in Section 14.5.

14.2 Mechatronic system design

The RECUPERA exoskeleton was developed in order to provide a support and training device for stroke patients. In cooperation with a medical device manufacturer, the general requirements for actuation, range of motion (ROM), degrees of freedom (DOF), safety and application scenario were defined. The exoskeleton was designed as a safe, lightweight and modular system in both the mechanical and electrical sense.

Two configurations were built to be able to perform the training application both in sitting and standing position. The first configuration includes the arm structure and is mounted on a wheelchair, while the second is a full-body system with active hip, knee, ankle, and spine to support its own weight (see Fig. 14.1).

The RECUPERA exoskeleton is also used to remotely control another robot. The target system is the humanoid robot RH5 MANUS [10], which was designed to support a human in assembly tasks. This includes complex gripping tasks of objects designed for humans. The requirements change noticeably due to the new application, where an input device for grasping with force feedback is needed.

The main changes are the addition of the wrists with two active DOF, larger torques and forces acting on the human and robotic arms, and new possibilities for fine manipulation with the human hand.

14.2.1 Mechanical design

In order to adapt the exoskeleton to the needs and movement spaces of humans, serial and parallel kinematics were combined and equipped with the institute's own drives. The system is thus a modular serial-parallel hybrid robot with numerous adaptation options, components of which are presented below. Fig. 14.2 gives an overview about sub-mechanisms and its used actuators, as Table 14.1 shows the ROM.

14.2.1.1 Actuators

Due to the lack of suitable commercial offers, DFKI has developed its own drives for the robotic systems for a torque range of 3 Nm to 500 Nm (in case of rotary drives) or force range of 1 kN to 5 kN (for linear drives). Two types of these drives are used in the RECUPERA exoskeleton along with some commercial servos and are presented below.

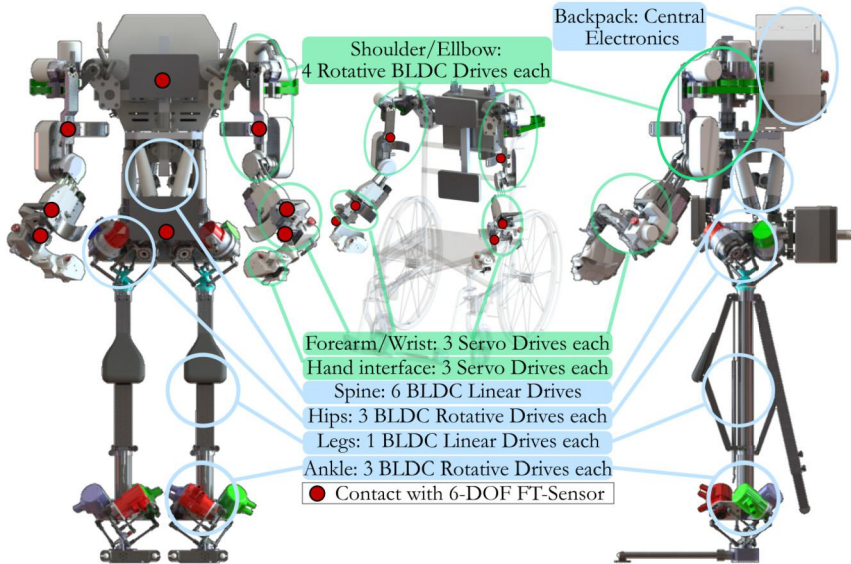


Figure 14.2 CAD overview of the robotic systems. Components marked in green are part of both systems, components marked in blue belong exclusively to the full-body exoskeleton.

DFKI rotative drives

The arms and ankles, as depicted in Fig. 14.2, are equipped with drives consisting of a combination of a BLDC installation kit from TQ-SYSTEMS and a HARMONICDRIVE gearbox. This provides a high power density and low backlash (< 1 arcmin) drive unit. Magnetic off-axis high-resolution position sensors MU from iC-HAUS are used for the rotor commutation and the detection of the absolute joint position.

DFKI linear drives

These drives essentially consist of a combination of a TQ-SYSTEMS BLDC motor and a ball screw. The motor directly drives the nut of the ball screw. The spindle itself is stationary, which is particularly advantageous for long spindles in combination with high speeds. The commutation and incremental measurement is done with the help of an iC-HAUS MU sensor. For absolute measurement, a linear potentiometer is used in the leg drives and a rope potentiometer in the STEWART-GOUGH PLATFORM drives. The backlash in the drives is (< 0.01 mm).

Table 14.1 Overview of range of motion.

Body part	Motor	Movement	Range
Shoulder	RD 50x8	Abduction / Adduction	-87° to 40°
		Ext. Rotation / Int. Rotation	-40° to 75°
Elbow	RD 50x8	Flexion / Extension	-170° to 30°
		Flexion	0° to 145°
Forearm	DY XH540W	Pronation / Supination	-88° to 88°
Wrist	DY XH430W	Palmar Flexion / Dorsiflexion	-43° to 43°
		Radial abduction / Ulnar add.	-20° to 40°
		Forward / Backward	-0.143 m to 0.122 m
Spine	RD 38x8	Left / Right	-0.153 m to 0.153 m
		Up / Down	-0.056 m to 0.057 m
		Flexion / Extension	-33° to 33.5°
		Lateral Bending	-33° to 33°
		Rotation	-87° to 87°
Hip	RD 70x10	Flexion / Extension	-20° to 37°
		Adduction / Abduction	-15° to 35°
Leg	RD 38x12	Lateral Rotation / Medial Rot.	-20° to 37°
		Up / Down	0.46 m to 0.71 m
Ankle	RD 50x8	Dorsi Flexion / Plantar Flexion	-20° to 37°
		Eversion / Inversion	-15° to 35°
		Adduction / Abduction	-20° to 37°

Commercial servos

DYNAMIXEL-X servo drives from ROBOTIS are used in the forearm for pronation and supination and in the wrist. They consist of a BLDC motor combined with a spur gear and are equipped with their own electronics including a position sensor. MKS-DS95 model servo drives are installed in the hand interface and actuate the fingers.

14.2.1.2 Sub-mechanism modules

Depending on the function and range of movement in the overall system, the sub-mechanisms are designed and combined to form a complete system. Fig. 14.3 gives a detailed view of the individual mechanisms.

Table 14.2 Overview of used actuators with following abbreviations: TQ-SYSTEMS ROBODRIVE(RD), HARMONICDRIVE (HD), Muliturn (MT), rotative (rot), linear (lin), ROBOTIS DYNAMIXEL (DY).

Motor	Type	Gear	Torque	Max Speed	ROM
RD 50x8	rot	HD CPL14A-50/100:1	18/28 Nm	700/350 °/s	MT
RD 70x10	rot	HD CPL25-160:1	92 Nm	132 °/s	MT
DY XH540W	rot	152.3:1	7.1 Nm	420 °/s	MT
DY XH430W	rot	353.5:1	3.4 Nm	180 °/s	MT
MKS-DS95	rot	Metal Gear	0.3 Nm	1132 °/s	360°
RD 38x12	lin	Ballscrew FBR 8x2	570 N	266 mm/s	130 mm
RD 38x12	lin	Ballscrew FGR 8x2	570 N	266 mm/s	420 mm

Shoulder mechanism

The human shoulder joint is a complex structure that allows a very high degree of mobility in six DOF. The RECUPERA shoulder mechanism simplifies this joint into a ball-and-socket joint that allows rotation in three axes, as depicted in Fig. 14.3a. This is an almost exact replica of the human shoulder joint, as the connection to the exoskeleton is not rigid and thus compensates for missing translational movements. It consists of a serially-connected chain of joints, the middle joint of which is a parallel-guided mechanism. The first joint in the chain allows for abduction and adduction, the second for internal and external rotation and the third for anteversion and retroversion of the arm. All three joints are arranged so that their axes of rotation intersect in the shoulder of the user. In order to increase the range of movement and to avoid a collision with the user, the second shoulder joint is designed as a planar six-bar double parallelogram mechanism. This allows the drive to be placed outwards without having the same axis of rotation. Two coupling joints are fixed to the back structure so that the end-effector rotates around an ideal point.

Elbow and forearm

The forearm mechanism has been extended by two active DOF in the wrist and the torques of the drives have been increased to allow adequate force feedback. The elbow drive is an RD 50x8 BLDC motor with a 50:1 re-

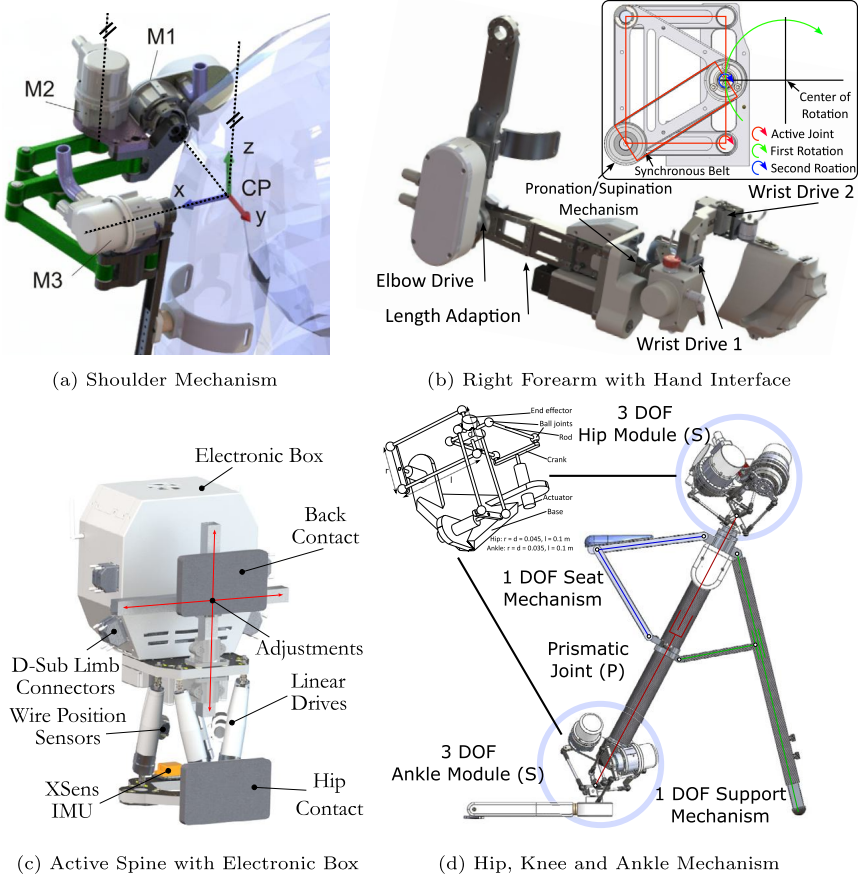


Figure 14.3 Sub-mechanism modules.

duction gear. The pronation and supination of the forearm is enabled by a parallel coupling gear inspired by the Harmony exoskeleton [11]. Fig. 14.3b shows the entire forearm and the detailed kinematics of the forearm is depicted in Fig. 14.6. At the lower lever of the parallelogram, the drive turns the entire mechanism. In the process, the left toothed belt pulley is also rotated and the rotation is transmitted to the output by means of a toothed belt, thus performing a movement around the ideal pivot point. Symmetrical selection of the swing arms results in an exact circular movement and power transmission in a ratio of 1:1. Two mechanical stops limit the total range of movement to 176° . This prevents that the singularity is reached, where all joints of the parallelogram are arranged in a line. The subsequent

joints of the wrist are arranged as a serial chain. The first joint allows radial abduction and ulnar adduction, the second palmar flexion and dorsiflexion.

Hand interface

The rehabilitation hand interface is designed as a curve guided coupling mechanism with one DOF and an end-effector movement range of 90° . The mechanism is driven by an MKS-DS95 servo with 0.3 Nm and can apply a maximum gripping force of 5 N. A HONEYWELL FSG15N force sensor is connected between the gear and the output and is used to control the gripping force.

The teleoperation hand interface (see Fig. 14.3b) was designed to enable an operator to remotely control a robot, with a focus on grasping tasks. For this, it features three surfaces with force feedback to create the sensation of grasping an object via a surrogate motion.

The surfaces are mounted on linear sliders with a stroke of 20 mm and are driven by an MKS-DS95 servo motor each. The resistive force, applied by the operator's fingers, is measured by HONEYWELL FSG15N force sensors. One slider corresponds to the index finger, one to the thumb and one to middle, ring and little finger. This allocation was chosen since in many tasks, the last three fingers are often used together, whereas thumb and index finger are used more individually. Since the sliders are driven by the servos via levers, the maximum feedback force is not constant throughout the ROM, but varies between 14 N and 20 N.

In addition to the sliders with gripping surfaces, each hand interface features two buttons, a joystick with integrated push button and an emergency stop. The hand interface is ergonomically shaped and provides a hand rest to assist the operator in keeping a relaxed hand posture.

Torso structure

The torso design utilizes the advantages of a parallel kinematic machine in terms of the force-to-weight ratio and the inherent limitation of the ROM. The STEWART-GOUGH PLATFORM in Fig. 14.3c consists of six variable length drives operating in parallel. It allows three rotational and three translational motions and is classified as 6-[UPS] mechanism, with the underlined letter indicating the active joint. It is highly mobile, highly dynamic and has high rigidity. The linear spindle drives are shown in Section 14.2.1.1 and are equipped with an additional absolute linear sensor and mechanical limits. All drive electronics are based on the baseplate and are

unified as Actuator Control Unit, see Section 14.2.2.1. The movement of the back assists in standing up and sitting down and increases the reach of the arms. The electronic box is mounted on the top plate of the mechanism. An XSENS MTI300 inertial sensor is also mounted on the base plate of the hip joint connection.

Hip and ankle

The hip and foot are equipped with a specially developed almost spherical parallel mechanism (ASPM) that acts as a 3 DOF swivel joint. For this purpose, three rotative drives each drive the end-effector via a spatial quadrilateral consisting of a coupling rod with two ball pivots. In the end-effector, the quadrilaterals are perpendicular to each other and intersect at their centers. In mechanism theory, the ASPM is classified as a PKM of type 3- $[\underline{R}-[2-SS]]$ [12,13]. Mechanical end stops limit the three actuators and thus also the overall mechanism. A special feature is the equal distribution of the tension within the mechanism; due to the ball heads, only compression and tension forces are transmitted via the coupling rods. A force acting in the direction of the axis of rotation is absorbed by the bearing without active torque.

Knee joint

The legs of the exoskeleton cover approximately the full human ROM with 7 DOF fully actuated kinematics. Instead of a usual $\underline{S}-\underline{R}-\underline{S}$ architecture, in which a spherical joint is used to actuate the hip and ankle joint respectively and a rotational joint is used in the knee, the knee joint used here is replaced by a prismatic coupling between the hip and ankle joint. The resulting $\underline{S}-\underline{P}-\underline{S}$ architecture significantly reduces the bending stresses occurring in the leg structural components. Furthermore, the legs have additional passive kinematics in the area of the knee joints, which unfold from a certain shortening of the prismatic actuator. This serves to unfold an additional support as well as a seat structure in which the wearer of the exoskeleton can rest without the actuators having to utilize any electrical power (Fig. 14.3d). The unfolding threshold on the prismatic actuator was chosen such that the passive kinematics remain closed during a normal walking pattern and only unfold when the user flexes the knees further. The knee mechanism is actuated by a linear actuator consisting of a BLDC motor and a ball screw. For sitting and standing, the 3 DOF hip and ankle drives are used additionally. With the linear actuator force capability provided in Table 14.2, the two legs can apply a total force of 1120 N.

With a total weight of the exoskeleton of 42 kg, an additional weight of up to approximately 70 kg can be supported by the leg design in an upright posture.

14.2.1.3 Safety aspects

Safety in the use of an exoskeleton plays a fundamental role in order not to endanger the user. For the mechanical design, this essentially means limiting the forces and ROM of the robot. In order to maintain the dynamics and freedom of movement of the user, a negotiation process is necessary. In the RECUPERA exoskeleton, the drives were designed according to these principles and have integrated movement-limiting mechanical stops. Additionally, in the joints with high DOF such as the back, hips and ankles, the use of parallel kinematics provides intrinsic movement limitation. The human contacts to the exoskeleton are not rigid, but designed with Velcro. The back connection offers enough freedom to compensate for any misbehavior of the robot.

14.2.1.4 Interface with human

The upper-body exoskeleton is connected to the human at three contact points per arm and at two contact points in the back. The contacts on the upper arm, forearm and hand interface are equipped with 6-axis ATI NANO25 force/torque sensors and can thus measure the forces that occur between the exoskeleton and the human. In the full-body exoskeleton, 6-axis ATI NANO25 sensors are also installed in the contact points on the back and hips. A loop in the foot mechanism enables contact with the human foot. The hip and back contacts are realized by straps with quick-release fasteners, the contact in the upper arm by Velcro fasteners. Fig. 14.2 documents the position of the contacts and their sensory equipment.

14.2.1.5 Adaption to different human sizes

The exoskeleton is designed for people with an approximate body height of 1.6 m to 1.9 m. The necessary adaptation options are provided in the shoulder, arms, back and leg structures. The upper arm length is adjustable by 55 mm, the forearms have a possible length adaptation of 50 mm, as well as two different length elbow adapters that allow for an additional extension of 50 mm. The wrist can be adjusted by up to 15 mm horizontally and vertically, and the hand interface can be moved forward by 13 mm. The shoulder can be adjusted in height (79 mm) and width (130 mm). In addition, the STEWART-GOUGH PLATFORM and the height of the legs

can be adjusted to a defined starting level within a height of 100 mm. But this takes affect to the possible task space.

14.2.2 Electrical and electronic design

The electrical and electronic design of both the wheelchair and the full-body exoskeleton is based on a hybrid centralized-decentralized control scheme. Fig. 14.4 shows an overview of all actuators and the underlying network structure (see also Section 14.2.2.1 for details). The control on the actuator level is performed in a decentralized manner and provides real-time capabilities to support a multi-level safety strategy. Both the mid- and high-level controls are computed in the central processing system of the exoskeleton (see Fig. 14.4 and Section 14.2.2.2). They provide complex functionality, such as kinematics/dynamics computations and functions to interact with the user.

14.2.2.1 Decentralized actuator-level controllers

Every actuator is controlled by a dedicated modular Actuator Control Unit (ACU) which is placed close to the corresponding actuator. The ACUs are self-designed and have been developed to specifically control BLDC motors. An ACU typically contains three separate subunit PCBs: one PCB for power electronics, one for data acquisition and communication and one PCB for computing. Multiple different ACU PCB subunits can be combined in order to fulfill specific requirements of each actuator. Table 14.3 shows an overview of the configurations used in the RECUPERA exoskeleton. The control and communication functionality is realized using a dedicated hardware design in the Xilinx XC6SLX45 Spartan 6 FPGA on the computing PCB subunit. The used controller is implemented as a cascaded position-velocity-current PID algorithm (see Section 14.3.3.1). Every ACU is supplied with two different voltages. For the motor phases, a voltage of 48 V is used, while 12 V is used for the computing and communication part. Both voltages and the related currents are continuously monitored by the FPGA. For safety reasons, the FPGA implements a configurable, firmware-based fuse. Additionally, each ACU contains a hardware fuse.

14.2.2.2 Central electronics for mid and high level control

The central electronic system of the exoskeleton is located in a backpack. It contains all components that are required for the power supply, mid-

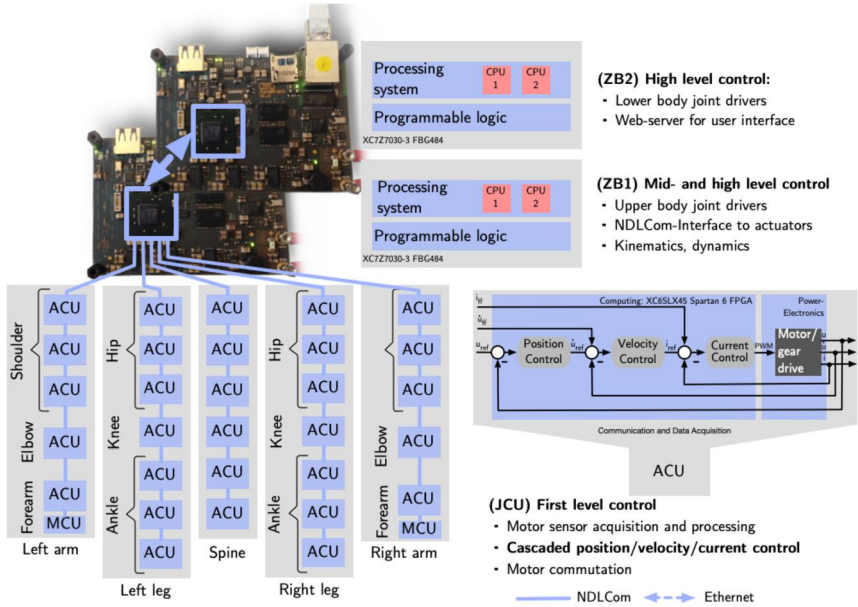


Figure 14.4 The exoskeleton is controlled by two central systems, called ZynqBrain (ZB), and a network of decentralized Actuator Control Units (ACU) for motor control. Each motor is controlled locally by an adjacent ACU. The distributed ACUs are connected via an NDCom network [14]. Bottom right: cascaded actuator-level control architecture implemented on the FPGA of each ACU; u is the angular position, \dot{u} is the angular velocity and i is the motor current. The reference values are provided by the mid-level control on the ZBs. All ACUs continuously send telemetry status data to ZB1 with 1 kHz using the NDCom network.

and high-level control, software for the user interface, communication and networking as well as safety features like a wireless emergency switch. The main computing system consists of two dedicated Pico-ITX PCBs (70×100 mm) called *ZynqBrain*. It contains a Xilinx Zynq ZC7030 [16] System on Chip. A ZC7030 consists of two sections: a Processing System (PS) (which is a dual-core ARM Cortex-A9 CPU running at 1 GHz) and a Programmable Logic (PL) section. Furthermore, each ZynqBrain contains 512 MB DDR3 SD-RAM [17]. For dedicated communication with the ACUs, each ZynqBrain contains five duplex high-speed and low-voltage differential signaling (LVDS) interfaces (see also [18]).

Table 14.3 Configurations of the Actuator Control Units (ACU). Each ACU consists of zero or more PCBs for **Power** electronics, **Data** acquisition and communication, **Computation** or **Microcontroller**. To control a motor, it senses the motor position via iC-MU [15] **Absolute** encoders or **Relative** encoders (Hall sensors). In addition, **Linear** potentiometers (WayCon) or draw **Wire** sensors (WayCon) are used for linear drives to sense the length.

	Location	ACU PCBs	Drive Unit	Sensors
Upper-body	Shoulder	P, D, C	RD 50x8-28	2xA
	Elbow	P, D, C	RD 50x8-14	2xA
	Forearm	D, C, M	Dy XH540W	2xA
Dy XH430W			1xA	
Lower-body	Spine	P, D, C	RD 38x12	A, W
	Hip	P, D, C	RD 70x10-160	A, R
	Knee	P, D, C	RD 38x12-2	A, R, L
	Ankle	P, D, C	RD 50x8-100	2xA

14.2.2.3 Power management

The DFKI's own Central Power Management Board (CPMB) serves as internal power supply and battery management. This allows the entire system to be supplied with the required voltages. The central computing unit is operated with 5 V, the decentralized ACU with 12 V for logic circuit and 48 V for power electronics. The CPMB can switch between an external power supply and a battery, as well as perform its charging function. Two additional voltage converters are located in the elbow and supply the DYNAMIXEL-X motors with a 12 V voltage separate from the logic voltage, as well as the model servos in the hand interface with 6 V. In the full-body system, the individual assemblies' arms, legs and the back can be switched on and off with two programmable electrical fuses and their power consumption can also be monitored. A more detailed description can be found in [18].

14.2.2.4 Safety aspects

To ensure safe operation of the exoskeleton, safety mechanisms were designed on three levels. The 48 V voltage supply for the motor power can

be switched off externally on both systems by means of emergency buttons on the arms, hand interface and on the full-body system on the back. A wireless emergency button operated by an external person and a foot pedal for the user provide the same functionality. Hard and soft limits for position, speed and current are specified at the decentralized ACU level and are decoupled from the mid and high software levels. When the hard limits are exceeded, this also causes the 48 V voltage to be interrupted. The soft limits are used as a control value limit and in the case of the position, exceeding the limit is prevented by moving in the opposite direction. On the software level, as a third element, there are further setpoint limits that cannot be exceeded.

14.3 Modeling and control

In this section, we present the kinematic and dynamic modeling of the possible exoskeleton configurations, the control architecture and software design for various rehabilitation therapies and teleoperation.

14.3.1 Modular robot description models

In its application, the RECUPERA exoskeleton is not only controlled in terms of position, but also based on forces and torques. For this purpose, the dynamic parameters of mass, center of gravity, moments of inertia and axes of rotation as well as their orientations of the individual robot links are required. The values can partly be determined experimentally or can be extracted from the CAD model. This is done with the SW2URDF tool [19]. To do this, the robot must be divided into the link-joint structure. Coordinate systems are assigned to the links and joint axes from origin to end-effector. The SW2URDF tool reads the values calculated in the CAD software, converts them into the Universal Robot Description Format (URDF) and links them to the exported meshes in STL data format.

The commissioning of a complex robotic system requires functioning sub-assemblies. Troubleshooting the entire system is complex and time-consuming due to the numerous possible errors on the hardware and software side. It makes sense, if possible, to commission and test individual sub-assemblies. Also, if there is a need to combine sub-assemblies in different ways, the advantage of a modular structure comes into play. Fig. 14.5 shows an overview of the different sub-assemblies used, which

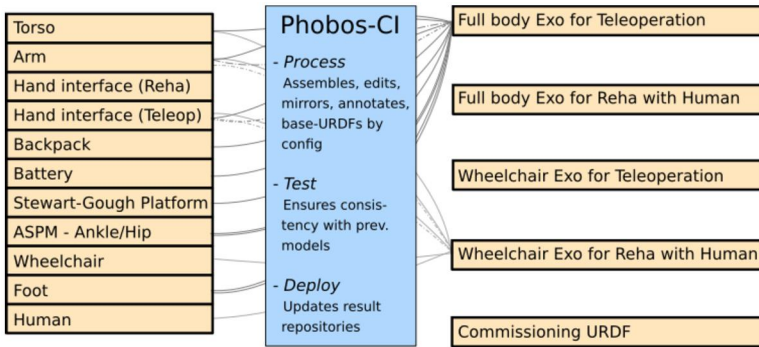


Figure 14.5 Overview of the exported (left), processed and used models (right). The PHOBOS-CI used to process the models is explained in Chapter 17.

are maintained using a Continuous Integration (CI) pipeline (see Chapter 17). Using this CI during the commissioning the various models are held consistent with each other.

14.3.2 Kinematics and dynamics

RECUPERA exoskeleton is a highly complex series-parallel hybrid mechanism with 34 DOF, where 24 DOF are actuated with parallel submechanism modules. Overall, the exoskeleton can be seen as a tree-type composition of 5 series-parallel hybrid submechanisms involving 2 legs, 2 arms and a torso. Hence, the loop closure function (LCF) of the overall system can be composed by combining the LCF of the 5 individual sub-assemblies.

Analytical LCF of RECUPERA arm

The RECUPERA exoskeleton arm is a 7 DOF series-parallel hybrid mechanism which contains a double parallelogram linkage at the shoulder joint and a parallelogram linkage in wrist joint (see Fig. 14.6 for its schematic and topological graph). The loop closure function of the parallelogram-like linkages can be composed from the mimic joint definition in URDF [20]. We choose an identical set of independent (\mathbf{y}) and active (\mathbf{u}) joints from their topological graph shown with red edges ($J_{100,200}$, $J_{200,232}$, $J_{300,400}$, $J_{400,500}$, $J_{500,511}$, $J_{600,700}$, $J_{700,800}$). The spanning tree joints (excluding the loop joints shown by dotted blue edges) are collected in tree joint position

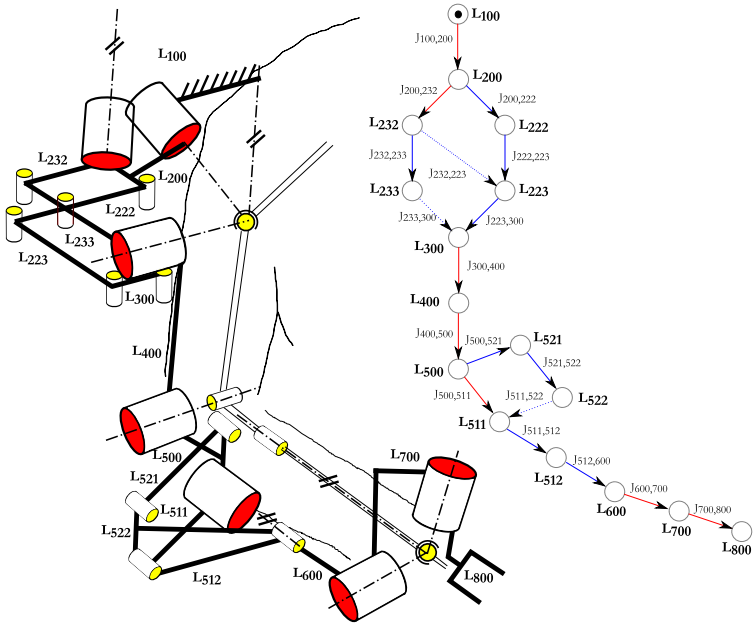


Figure 14.6 Single arm schematic and its topological graph.

vector q and its relation with y , i.e., LCF (γ), is shown in Eq. (14.1).

$$\begin{aligned}
 q = \begin{bmatrix} q_{100,200} \\ q_{200,222} \\ q_{222,223} \\ q_{223,300} \\ q_{200,232} \\ q_{232,233} \\ q_{300,400} \\ q_{400,500} \\ q_{500,521} \\ q_{521,522} \\ q_{511,522} \\ q_{500,511} \\ q_{511,512} \\ q_{512,600} \\ q_{600,700} \\ q_{700,800} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_{100,200} \\ q_{200,232} \\ q_{300,400} \\ q_{400,500} \\ q_{500,600} \\ q_{600,700} \\ q_{700,800} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14.1}
 \end{aligned}$$

Numerical LCF of RECUPERA leg

The RECUPERA exoskeleton legs are each 7 DOFs with a 3 DOF ASPM representing the hip and ankle modules, and a 1 DOF prismatic joint that mimics the knee. A comprehensive kinematic analysis of ACTIVE ANKLE is provided in [13,21,22]. Since, it is not possible to get rotative inverse kinematics of the ASPM in a fully analytical fashion, the LCF of the ASPM modules in hip and ankle joints are resolved numerically.

Hybrid numerical-analytical LCF of overall RECUPERA system

Since, it is straight-forward to solve the inverse kinematics of STEWART-GOUGH PLATFORM, the overall LCF of the RECUPERA system can be achieved in a hybrid numerical-analytical manner using the approach defined in [23], where the arms (γ_2, γ_3) and torso (γ_1) submechanisms are solved analytically (using Eq. (14.1) for left and right arms) and the leg submechanisms (γ_4, γ_5) are solved numerically. The overall LCF of the RECUPERA system at position, velocity, and acceleration levels are given by Eqs. (14.2), (14.3), and (14.4), respectively.

$$\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1^T & \boldsymbol{\gamma}_2^T & \boldsymbol{\gamma}_3^T & \boldsymbol{\gamma}_{4,num}^T & \boldsymbol{\gamma}_{5,num}^T \end{bmatrix}^T \quad (14.2)$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{G}_2 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{G}_3 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{G}_{4,num} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{G}_{5,num} \end{bmatrix} \quad (14.3)$$

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_1^T & \mathbf{g}_2^T & \mathbf{g}_3^T & \mathbf{g}_{4,num}^T & \mathbf{g}_{5,num}^T \end{bmatrix}^T \quad (14.4)$$

Dynamics

Once the LCF of the overall system is available, the equations of motion of the explicitly constrained series-parallel hybrid system can be derived in both forward and inverse manners. The inverse dynamic model is solved in real time to enable torque control of the system.

14.3.3 Exoskeleton control

14.3.3.1 First level control

As described in Sec. 14.2.2, the first level control architecture for each joint is implemented on FPGAs, each driving a single actuator using a cascaded position, a velocity, and a current control loop. Each of the control cascades

can be directly selected for control. In Fig. 14.4, we can see the actuator level control architecture. With the help of motor current measurements, it is possible to use torque control for the motors. Further, the actuator level modularity enables the implementation of decoupled safety checks at the mid level controllers. Position, velocity and current are limited to a maximal value and in case of a sensor failure the controller is stopped automatically. This low-level architecture meets the requirements for the implementation of therapy concepts and teleoperation and is a solid foundation for both kinematic and dynamic control implemented in the mid-level architecture.

14.3.3.2 Mid-level control

The mid-level control architecture implements the kinematic and dynamic model of the system and associated control approaches for rehabilitation therapies and teleoperation. In the (1) *Gravity Compensation* (GC) mode, the weight of the system is compensated with the help of an inverse dynamic model of the exoskeleton arms. GC mode can also be used to take into account the dynamics of the human arms. The input to this model is the actuator positions read from the position encoders (see Table 14.3). The output is the reference torque values, which are then converted into motor current and sent to the current controller implemented in the ACU. The GC mode is used to implement a transparent behavior of the system and represents the *basic operation mode* of the system, on which most of the other modes are based on. Wrenches measured at the force-torque sensors of the exoskeleton arms can optionally be applied to assist human control of the exoskeleton. To support repetitive movement therapies for stroke patients, (2) *Teach & Replay* (TR) mode can be used. This mode has two phases: First, gravity compensation for the affected arm is enabled so that a therapist can easily move the arm. The equipped touch sensor on the forearm (see Fig. 14.3b) recognizes the intention of the therapist to teach a movement and stores the trajectory (position and velocity readings from the involved ACUs) in the system's storage device. Secondly, the trajectory can be replayed according to a trigger by the patient or therapist. During the replay, the exoskeleton executes the trajectory movement in the cascaded position-velocity control mode in the ACU. Additionally, mirror therapy is supported using the (3) *Mirror* (M) mode. Here, movements from the healthy arm can be transferred to the unhealthy arm in a mirrored fashion. In this mode, the healthy arm is gravity-compensated and the unhealthy arm is in cascaded position-velocity control mode. The actuator positions read from the healthy arm are mirrored and sent to the ACUs

of the unhealthy arm. Further, sitting and standing features for the lower part of the full-body exoskeleton are implemented at this level. As a new feature for telemanipulation, a robotic arm or the arms of a dual armed robot can be controlled using the (4) *Teleoperation* (TO) mode. The mapping between source system (exoskeleton) and target system (e.g., RH5 MANUS) is done by a Cartesian mapping of the poses of selected end-effectors and can be scaled in Cartesian space. For that, a predefined pose mapping between end-effector frames is needed by adding additional links to the end-effectors in the URDF with a correcting transformation. For force feedback, the force-torque sensor wrenches at the target system are mapped to and applied at the corresponding frames of the source system. Wrenches can also be scaled and capped before they are applied to the exoskeleton. Additionally, the three trigger buttons on the left and right hand interfaces can be mapped to gripper joints.

14.3.3.3 High level control

A web-based GUI is provided for high level control of the exoskeleton and can be accessed using a mobile phone, tablet or PC. The web server is hosted on ZynqBrain2 and based on the Python Flask framework [24]. The GUI allows the user/therapist to select the operation mode of the exoskeleton, which can be either one of the different therapy modes (GC, M, or TR) or the TO mode. It is also possible to manage different patient/operator profiles storing specific information like ID or recorded movements. As the exoskeleton can be adjusted to the user, shoulder width, upper arm length, forearm length and hand size values can be entered in the GUI to automatically create the corresponding user-specific URDF files with the adjusted segment lengths and inertia using PHOBOS. Moreover, it allows the operator to use the exoskeleton in different settings: single arm, dual arm, full body, etc. Both left and right sided users can be supported. For the TO mode, force feedback can be manually activated and capped. As force feedback changes the poses of the exoskeleton hand interfaces, which in return changes the mapped end effector poses of the teleoperated robot, it is also possible to disable position control, so that the operator can have force feedback without changing the target system.

14.3.3.4 Software design

The high- and mid-level control is implemented using the Robot Construction Kit (Rock) [25]. It is based on the component model of the Orocos Real Time Toolkit (RTT) and an object request broker (ORB)

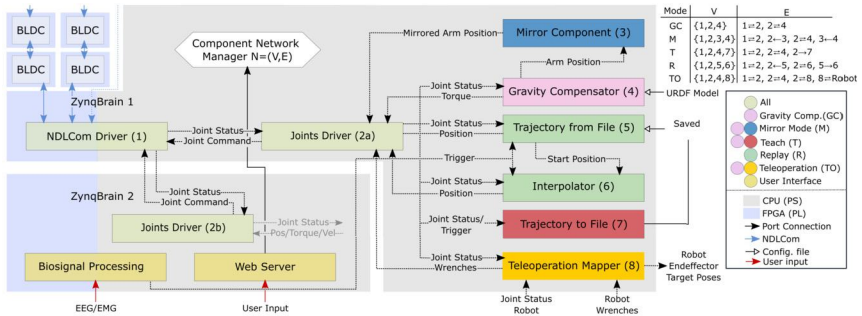


Figure 14.7 Software architecture overview: A Component Network Manager configures, connects and starts the subset of components (V) required for a specific mode. The corresponding directed connections (E) are described in the table at the top right corner. The components required for each mode are also represented by colors. A web server application hosted on ZynqBrain 2 is used as user interface.

implementation called omniORB. Rock tasks, similar to ROS (Robot Operating System) nodes, encapsulate different functionalities, run independently and provide input and output for other tasks (see Fig. 14.7). Each task can be configured individually. This enables a very flexible way to adjust the system and to distribute computational demanding components among the two ZynqBrains. Additionally, a web server is running on the second ZynqBrain, providing access to the previously described web GUI written in JavaScript (see Section 14.3.3.3). Furthermore, some functionalities can be triggered by biosignals like electroencephalogram (EEG) or electromyogram (EMG), which can be processed directly on the embedded processors [3,17].

14.4 Results and discussion

This section presents the experimental results of the rehabilitation therapy modes and the teleoperation implemented in the wheelchair configuration. The rehabilitation modes were also tested in clinical settings with affected individuals, see [26]. Since the upper body design is identical for both configurations, the results are equally valid between them. Further, we present a comparison with current state of the art exoskeletons for rehabilitation and teleoperation purposes.

14.4.1 Gravity compensations mode

Transparency of the exoskeleton to the user requires a good gravity compensation model, which is also needed for a good usability for the therapist. In our experiments, described in [22], the norm of mean absolute error (MAE) in joint space of four different balanced poses was between 0.12 Nm & 0.26 Nm, which demonstrates the good quality of the model. In the experiments, the exoskeleton user was able to move its arms freely within the limits of the system, as depicted in Fig. 14.8a. When using the gravity compensation mode, it is also possible to include the weight of the human arms into the model for compensation. Additionally, gravity compensation mode is being used for the get-in helper mode, enabling the human operator or patient to easily enter the exoskeleton. This can be done in less than a minute on average for healthy users if no adjustments to the system are needed.

14.4.2 Rehabilitation

14.4.2.1 Teach and replay

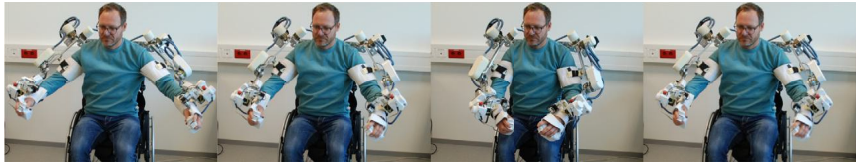
The teach and replay mode gives a therapist the possibility to pre-train movements and later replay these as a sequence or individually. The patient can benefit from a self-intended movement start, since the replay of a pre-trained movement can be triggered via residual muscle activity measured with surface EMG and thus give the patient the possibility to train self-paced, shown in Fig. 14.8b.

14.4.2.2 Mirror mode

The mirror mode mimics a mirror therapy by directly transferring movements from the healthy arm to the affected one. In this mode, the non-affected arm controls or moves the exoskeleton in the gravity compensation mode. All movements are mirrored to the affected side which is running in the position control mode (see Fig. 14.8c). With this mode, the patient is able to do self-determined training.

14.4.2.3 Gravity compensation with human arm model

As an addition to the pure gravity compensation mode, the weight of the arm of the user can be modeled as well. In this way, only very small residual muscle force is sufficient to move the affected arm. This mode gives patients the opportunity to do self-paced training of voluntary movements, without



(a) Gravity Compensation Mode: base mode for mechanical transparency.



(b) Teach and Replay Mode: recorded movements are replayed.



(c) Mirror Mode: all movements are mirrored from the right to the left side.

Figure 14.8 Selected modes of exoskeleton use.

any constraints like in the other modes where the movements are pre-trained or mirrored from the non-affected side.

14.4.3 Teleoperation

In teleoperation mode, we are able to remotely control both the arms of our humanoid robot RH5 MANUS. The elbow and wrist poses of the exoskeleton were mapped to RH5 MANUS, while wrenches measured at the force-torque sensors in the wrists of the humanoid were applied at corresponding links of the exoskeleton. Using the trigger buttons at the hand interfaces, the fingers of the two-finger and four-finger grippers could be controlled independently. With this setup, it was possible to grasp and pick up an object with one hand, rotate it and then put it back on a table (see Fig. 14.9).

In further tests, we were also able to grasp an object like a box or a soft ball with both hands at the same time. For this, we did not use the fingers to grasp the object, but used the hands for contact.

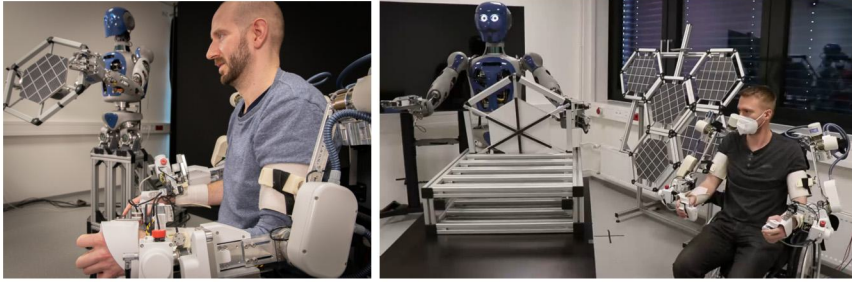


Figure 14.9 Exoskeleton teleoperating RH5 MANUS.

In all our teleoperation tests, Cartesian workspace scaling has proven to be a helpful tool, especially for robots with different workspaces than the exoskeleton. Workspace scaling can also be used to scale down the human movement and therefore enable a very precise pose control. Also, the scaling can be chosen according to the current task.

14.4.4 Comparison with similar exoskeleton systems

A classification of the exoskeleton can be done by featuring active DOF, ROM, mobility, number of tracked limbs and weight. A comparison with similar upper body exoskeleton systems in Table 14.4 shows that the RECUPERA upper body exoskeleton has the best lightweight design. The RECUPERA exoskeleton is a mobile system mounted to a commercially available wheelchair. It has a high amount of active DOFs with a low weight. The total weight of an arm after enhancing the system to active 7 DOF is 7.14 kg. This also includes an active hand interface for teleoperation. By modifying the arms, the ROM of the elbows and forearms was significantly increased. Although it does not cover the complete human workspace, it supports most activities of daily living tasks. The alternative inclusion of an active shoulder girdle would have increased shoulder ROM to achieve full working range, but would also have limited design complexity and thus mobility. In the full-body system, the total working range of the arms is increased by the active back. The active back and legs enable rehabilitation therapies of the upper body in sitting as well as in standing.

Table 14.4 Comparison of updated RECUPERA upper-body system to other upper-body exoskeletons and the range of motion of activities of daily living.

Device	RECUPERA	ANYexo [27]	Harmony [11]	SUEFUL-6 [28]	ADL [11,29,30]
Weight (kg)	14.3	12.98	31.2	-	-
DOF	14	7	14	6	-
Bilateral	yes	no	yes	no	-
Torso harness	yes	no	yes	no	-
Gripper	yes	no	no	no	-
Wheelchair	yes	no	no	yes	-
Abd./Adduction °(Nm)	87/40(28)	170/0(40)	170/60(34)	-	131/54
Ex./Int. rot. °(Nm)	40/75(18)	105/105(40)	79/80(34)	0/90(35)	76/62
Flex./Ext. °(Nm)	170/30(28)	170/50(40)	160/45(34)	0/90(9)	131/51
Elbow flex. °(Nm)	145(18)	145(40)	150(13)	120(9)	148
Pro/supinat. °(Nm)	176(7.1)	-	172(1.25)	140(4)	167
Wrist Abd./Add. °(Nm)	20/40(3.4)	-	-	20/30	30/40
Wrist Flex./Ext. °(Nm)	43/43(3.4)	-	-	60/50	60/60

14.5 Conclusion and outlook

The RECUPERA exoskeletons are two modular, lightweight, safe and ergonomically adaptable robotic systems that serve to capture and guide human movement. The upper-body part system attached to a wheelchair, as a partial variant of the full-body system, reduces complexity and transfers its own weight to the floor via the wheelchair. However, the full-body system supports its own weight and enables a wider range of movement due to an active back. The active legs enable a sitting posture through an extendable seat mechanism. Both systems can be used for rehabilitation applications and for teleoperation. Since the systems are designed as rigid exoskeletons, the range of motion and system dynamics must be optimally suited for mechanical transparency for the user. This applies to both teleoperation and rehabilitation applications. Here, further improvements can be made. The mechanical components can be optimized with regard to medical-technical specifications. In order to increase the system dynamics, a change to drives with a reduced gear reduction could be considered.

The RECUPERA exoskeleton was used in two application scenarios. Using gravity compensation mode as a basic function, the exoskeleton is self-supporting and allows free movement of the user. The humanoid robot RH5 MANUS was remotely controlled. Through the exoskeleton a haptic impression of the object manipulation was given using force feedback. Additionally, Cartesian workspace scaling enables a precise remote control. When manipulating an object similar to a ring using a dual arm power grasp, this introduced a kinematic closed loop of both the arms, which were now connected through the object. Due to latencies introduced by the two systems, the interaction forces of the two arms with the object and therefore with the other arm could not be compensated by the human operator and the system was rocking up. Additional tests with activated compliance mode for the wrists at mid-level control of RH5 MANUS reduced this effect. Further research needs to be done in this direction to enable dual arm power grasping.

As a rehabilitation application, mirror therapy and EMG triggered Teach & Replay mode as well as the arm weight compensation model were tested with patients. It was observed that depending on the severity of the patient's restriction different modes were supporting rehabilitation therapy best.

The assist-as-needed technique in assistive robotic rehabilitation has proven to promote motor recovery and induce neuroplasticity by encouraging the patient to actively participate in the movements [31]. This is in stark contrast to the above discussed Teach & Replay and Mirror modes, where the exoskeleton does the entire work. To deviate from this and give the patient full control of initiating and executing these movements up to their current capabilities, this technique will be integrated into the exoskeleton design in the near future. In essence, this technique encourages the patient to carry out movements depending upon their muscular ability and assistance will be provided only when they are deviating significantly from the desired trajectory to guide them towards the target.

To this end, the task is divided into three categories – initial calibration, online estimation of muscle torques and assistive control. In the calibration phase, Maximum Voluntary Isometric Contraction would be used to generate a preliminary estimate of muscle strength. Promising progress has been made in real-time estimation of the joint torques using the EMG signals. Several experiments have been conducted at DFKI in this regard wherein this mapping has been achieved for single joint force using a linear regression model and for multiple joints using a two-layer artificial neural

network model, for which further improvements and validations are still needed. The results will be published in the near future.

Furthermore, to ensure that the patient is provided with a free zone around the desired trajectory, a tunnel-based torque controller is planned to be implemented. As the performance of the controller depends on the accuracy of the inverse dynamic model, it forms a major bottleneck and its quality needs to be further investigated in the future.

References

- [1] F. Just, Ö. Özen, P. Bösch, H. Bobrovsky, V. Klamroth-Marganska, R. Riener, G. Rauter, Exoskeleton transparency: feed-forward compensation vs. disturbance observer, *at-Automatisierungstechnik* 66 (12) (2018) 1014–1026.
- [2] R. Gopura, D. Bandara, K. Kiguchi, G.K. Mann, Developments in hardware systems of active upper-limb exoskeleton robots: a review, *Robotics and Autonomous Systems* 75 (2016) 203–220.
- [3] E.A. Kirchner, N. Will, M. Simnofske, L.M.V. Benitez, B. Bongardt, M.M. Krell, S. Kumar, M. Mallwitz, A. Seeland, M. Tabie, H. Wöhrle, M. Yüksel, A. Heß, R. Buschfort, F. Kirchner, Recupera-Reha: exoskeleton technology with integrated biosignal analysis for sensorimotor rehabilitation, in: *Transdisziplinäre Konferenz SmartASSIST*, 2016, pp. 504–517.
- [4] E.A. Kirchner, J. Bütefür, Towards bidirectional and coadaptive robotic exoskeletons for neuromotor rehabilitation and assisted daily living: a review, *Current Robotics Reports* 3 (2022) 21–32.
- [5] K.J. Miller, A. Gallina, J.L. Neva, T.D. Ivanova, N.J. Snow, N.M. Ledwell, Z.G. Xiao, C. Menon, L.A. Boyd, S.J. Garland, Effect of repetitive transcranial magnetic stimulation combined with robot-assisted training on wrist muscle activation post-stroke, *Clinical Neurophysiology* 130 (8) (2019) 1271–1279, <https://doi.org/10.1016/j.clinph.2019.04.712>.
- [6] A.U. Pehlivan, D.P. Losey, M.K. O'Malley, Minimal assist-as-needed controller for upper limb robotic rehabilitation, *IEEE Transactions on Robotics* 32 (1) (2016) 113–124, <https://doi.org/10.1109/TRO.2015.2503726>.
- [7] M. Folgheraiter, M. Jordan, S. Straube, A. Seeland, S.K. Kim, E.A. Kirchner, Measuring the improvement of the interaction comfort of a wearable exoskeleton, *International Journal of Social Robotics* 4 (3) (2012) 285–302.
- [8] M. Mallwitz, N. Will, J. Teives, E.A. Kirchner, The CAPIO active upper body exoskeleton and its application for teleoperation, in: *Proceedings of the 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2015)*, ESA, 2015.
- [9] R. Sonsalla, F. Cordes, L. Christensen, T.M. Roehr, T. Stark, S. Planthaber, M. Maurus, M. Mallwitz, E.A. Kirchner, Field testing of a cooperative multi-robot sample return mission in Mars analogue environment, in: *Proceedings of the 14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2017.
- [10] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 01–07.
- [11] B. Kim, A.D. Deshpande, An upper-body rehabilitation exoskeleton harmony with an anatomical shoulder mechanism: design, modeling, control, and performance evaluation, *The International Journal of Robotics Research* 36 (4) (2017) 414–435.

- [12] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, *Journal of Intelligent & Robotic Systems* 94 (2019) 303–325, <https://doi.org/10.1007/s10846-018-0792-x>, <https://link.springer.com/article/10.1007/s10846-018-0792-x#citeas>.
- [13] M. Simnofske, S. Kumar, B. Bongardt, F. Kirchner, Active ankle – an almost-spherical parallel mechanism, in: 47th International Symposium on Robotics (ISR), 2016.
- [14] M. Zenzes, P. Kampmann, T. Stark, M. Schilling, NDLCom: simple protocol for heterogeneous embedded communication networks, in: Proceedings of the Embedded World Exhibition & Conference, Nuremberg, Germany, 2016, pp. 23–25.
- [15] iC-MU Magnetic Off-Axis Absolute Position Encoder, <http://www.ichaus.de/ic-mu>.
- [16] Xilinx Inc., UG585 Zynq-7000 All Programmable SoC Technical Reference Manual, 1st edition, February 2015.
- [17] H. Wöhrle, M. Tabie, S.K. Kim, F. Kirchner, E.A. Kirchner, A hybrid FPGA-based system for EEG- and EMG-based online movement prediction, *Sensors* 17 (7) (2017), <https://doi.org/10.3390/s17071552>.
- [18] M. Yüksel, L.M.V. Benitez, D. Zardykhan, F. Kirchner, Mechatronical design and analysis of a modular developed exoskeleton for rehabilitation purposes, in: 10th International Conference on Electrical and Electronics Engineering (ELECO), 2017, pp. 711–716.
- [19] Solidworks to URDF exporter, http://wiki.ros.org/sw_urdf_exporter. (Accessed 30 September 2022).
- [20] S. Kumar, M. Simnofske, B. Bongardt, A. Müller, F. Kirchner, Integrating mimic joints into dynamics algorithms: exemplified by the hybrid Recupera exoskeleton, in: Proceedings of the Advances in Robotics, AIR '17, ACM, New York, NY, USA, 2017, pp. 27:1–27:6, <https://doi.org/10.1145/3132446.3134891>.
- [21] S. Kumar, A. Nayak, B. Bongardt, A. Mueller, F. Kirchner, Kinematic Analysis of Active Ankle Using Computational Algebraic Geometry, Springer International Publishing, Cham, 2018, pp. 117–125.
- [22] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the Recupera exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019).
- [23] R. Kumar, S. Kumar, A. Müller, F. Kirchner, Modular and hybrid numerical-analytical approach – a case study on improving computational efficiency for series-parallel hybrid robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022, 2022.
- [24] FLASK, web development one drop at a time, <http://flask.pocoo.org/>.
- [25] ROCK, the Robot Construction Kit, <http://www.rock-robotics.org>.
- [26] Recupera-Reha exoskeleton first patient test, suffering from a right sided paralysis after stroke, <https://www.youtube.com/watch?v=2Llq63VEFRg>. (Accessed 6 October 2022).
- [27] Y. Zimmermann, A. Forino, R. Riener, M. Hutter, ANYexo: a versatile and dynamic upper-limb rehabilitation robot, *IEEE Robotics and Automation Letters* 4 (4) (2019) 3649–3656.
- [28] R.A.R.C. Gopura, K. Kiguchi, Development of a 6DOF exoskeleton robot for human upper-limb motion assist, in: 2008 4th International Conference on Information and Automation for Sustainability, 2008, <https://doi.org/10.1109/ICIAFS.2008.4783986>.
- [29] D. Magermans, E. Chadwick, H. Veeger, F. Van Der Helm, Requirements for upper extremity motions during activities of daily living, *Clinical Biomechanics* 20 (6) (2005) 591–599.
- [30] H. Schmidtke, I. Jastrzebska-Fraczek, *Ergonomie: Daten zur Systemgestaltung und Begriffsbestimmungen*, Carl Hanser Verlag GmbH & Company KG, 2013.
- [31] S.Y.A. Mounis, N.Z. Azlan, F. Sado, Assist-as-needed control strategy for upper-limb rehabilitation based on subject's functional ability, *Measurements & Control* 52 (9–10) (2019) 1354–1361.

CHAPTER 15

RH5 Pedes humanoid

Melya Boukheddimi^a, Ivan Bergonzani^a, Shivesh Kumar^a,
Heiner Peters^a, and Frank Kirchner^{a,b}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bWorking Group Robotics, University of Bremen, Bremen, Germany

15.1 Introduction

15.1.1 Problem description

Robots are increasingly omnipresent in our daily lives. This goes from the small vacuum cleaner robot to the humanoid robot Pepper [1] that can be found at the largest airports. Robots are thus brought to interact and cooperate with human beings. Beyond the problem of creating more powerful and agile robots. Nowadays, the challenge of safe, comfortable and fearless interaction with human beings is the core of humanoids' development. To address this issue, more and more studies have been focused on the significance of designing an anthropomorphic robot [2–5]. In this work, we propose a new whole-body humanoid robot designed with the perspective to mimic the human being (mass, inertia, length, and appearance), in order to:

- perform human-like tasks in an optimal manner;
- perform fast and precise movements;
- be aware of its environment and itself in it;
- interact with human beings in a safe and comfortable manner;
- cooperate and help human beings to accomplish their tasks;
- and reach high payloads and dynamic capabilities.

To this end, we present the RH5 Pedes whole-body humanoid robot with a bio-inspired design.

15.1.2 Application scenarios

The proposed design was driven by the desired dynamic tasks. These tasks were formulated as use cases to evaluate the capabilities of the robot. The evaluations were conducted in two phases: upper body and whole body trials.

Upper-body evaluation movements:

- Consecutive arms and body motions at maximal joints velocity [6].
- Heavy weight lifting motions, to assess the payload performance of the arms and body parallel mechanisms [7].
- Fast and synchronized motions with the music's beat to create artistic and human comfort feelings [8].

Whole body evaluation movements:

- Movement of a single walking cycle.
- Three consecutive walking cycles.

In this work, only whole-body motion evaluations will be presented.

15.2 Mechatronic system design

15.2.1 Mechanical design

This section describes the mechanical design of the RH5 Pedes robot (see Fig. 15.1). The robot has an overall weight of 57 kg (excluding battery) with a height of 1.89 m and is the result of combining the upper body of the robot RH5 Manus and the legs of the robot RH5.

15.2.1.1 Design motivation

The main focus in the design of the humanoid robot was to come as close as possible to human proportions in terms of inertia and power density. In particular, the focus was on reducing the overall weight and size of the robot compared to the previous version. In addition, the arms and legs were optimized in terms of their inertia, while at the same time considering the highest possible stiffness of the structural components. Therefore, topology optimization methods for the design of structural components were used in combination with a series-parallel hybrid design concept. As described in Section 15.2.1.7, parallel kinematics can provide advantages which are especially useful for joints with limited range of motion (ROM). This can be:

- superposition of forces in parallel configurations;
- non-linear transmission ratio;
- higher joint stiffness;
- possibility to shift masses closer to proximities of the robot in order to reduce the inertia of extremities.

A serial architecture on the other hand usually allows larger ROM, which is useful for joints such as the robot's shoulder joint. In the following, the mechanical design of the subsystems of the robot is discussed. An overview

of the maximum velocity, range of motion and maximum torque provided at each joint is given in Table 15.2.

15.2.1.2 Torso

A hip and a torso structure connected by a 3 DOF spherical joint together form the torso of the robot. The pitch and roll motion is actuated by a 2-SPU+1U parallel mechanism. A similar mechanism has already been described in [9]. The yaw motion is actuated by a rotary motor placed in series with this mechanism, which allows a big ROM for this joint.

The body structure contains most of the robot's electronic components and at the same time serves as a rigid connector between legs, arms, body joint and head. Topology optimization methods have been used to ensure high stiffness. To allow good accessibility to the electronic components, some areas of the body structure had to be left out. An aluminum casting process has been chosen to allow great freedom in the design process on the way to a rigid structure, taking into account all design criteria. The optimization process resulted in a single support structure for the torso and hip, respectively. The overall weight of the torso has been reduced from 21 kg in RH5 [10] to 12 kg in RH5 Pedes. The robot carries a backpack, which is fastened only with four twist-locks. This allows a quick change between a simple back cover and a 5 Ah, 48 V LiFePo battery pack.

15.2.1.3 Manipulator arm

The manipulators each have 7 DOF, of which 3 DOF are shoulder joint, 1 DOF is elbow, and 3 DOF are wrist. The shoulder joint was designed as serial kinematics to ensure the largest possible range of motion. To increase the manipulation workspace of the robot, the first axis of the shoulder joint was tilted forward by 15° and upward by 14° with respect to the XZ plane. Actuator units, each consisting of a BLDC motor and a harmonic drive gear, were used to actuate the shoulder joint. The gear reduction ratio was chosen to be 1:100 despite the negative effects on dynamics retractability and transparency, in order to be able to safely manipulate a payload of at least 5 kg at the end effector. A 1-RRPR lambda mechanism is used for the actuation of the elbow to realize a discontinuous effective gear ratio. Thus, a high torque is provided when the elbow is flexed and a high velocity is available to move the joint from the extended rest position. At a joint angle of 91° , the highest torque can be used. The highest velocity is obtained at a joint angle of 0° . A linear actuator consisting of a BLDC motor directly

driving the nut of a ball screw with a pitch of 2 mm is used to actuate the lambda mechanism. Shifting the drive mass towards the shoulder joint has a positive effect on the mass distribution and thus the overall inertia of the arm during movements around the shoulder joint. The 3 DOF wrist has three rotational axes, all intersecting at a single point. A rotary actuator (BLDC motor and Harmonic Drive gear, ratio 1:100) was used for the roll motion. A more complex 2 DOF, 2-SU[RRPR]+1U parallel kinematic mechanism follows for the pitch and yaw motion. The mechanism previously described in [11] can be seen as two four-bar mechanisms coupled in parallel. Linear actuators are used for actuation, in which a BLDC motor drives a ball screw with a pitch of 1 mm. The kinematics were designed and optimized to allow a large ROM in the pitch direction while keeping the mechanism well conditioned throughout its full range of motion. At the distal end of the wrist, there is a mechanical interface where various end effectors can be mounted. An ATI 6-axis force-torque sensor was placed close to the interface to measure the torques and forces applied by the endeffector. All structural components of the arms were made of cast aluminum, which allowed for a complex design of free-form parts. Beyond the pure functionality as a support structure, the aluminum parts cover further functionalities, such as the housing of electronic components, bearing seats and the design of the outer shape of the robot.

15.2.1.4 Hand

An under-actuated, self-adaptive gripper concept was designed for the use with this robot [12]. A stand-alone finger assembly has been invented that includes an actuator, sensors and motor electronics. The finger serves as basis component for different gripper assemblies. Currently, 2-, 3-, and 4-finger grippers are available that can be attached to the wrist interface.

15.2.1.5 Head

The head is actuated by a differential 3 DOF joint, in which the first and second joint axes are differentially coupled and the third joint axis is passed centrally through the joint so that all three drives are rigidly connected to the base of the neck. The head carries a couple of electronic components and sensors which includes a ZED stereo camera, a Jetson TX2 GPU, an Axiomtek AX93276 CPU, a Velodyne Puck Laserscanner, etc. In addition, an eye design was developed in which visual feedback for the user can be provided by LEDs arranged in a ring around the camera lenses.

15.2.1.6 Leg

The robot is equipped with two identical legs, each with 6 DOF. Each leg has a 3 DOF spherical hip joint. The first two DOFs are designed as serial kinematics and are each driven by a rotary actuator consisting of a BLDC motor and a harmonic drive gear (gear ratio 1:100). The third DOF of the hip joint and the 1 DOF of the knee joint are each actuated in parallel by a lambda mechanism (1-RRPR), which, as previously described in the context of the elbow drive, leads to an advantageous, discontinuous effective transmission of speed and torque to the respective joint axis. The robot's 2 DOF ankle joint is actuated by means of a 2-SPRR+1U parallel kinematics [9], which enables a particularly large ROM in the joint's pitch direction. Actuation itself takes place via two linear drives, each consisting of a BLDC motor and a ball screw with a pitch of 2 mm. Due to the parallel actuation, the power of both drives is available for pure pitch movement. The intersection points of the hip joint axes have a distance of 220 mm and lie at a height of 930 mm above the foot contact area, which corresponds to about half the height of the entire robot. The first axis of the hip joint was tilted inwards by 15° with respect to the XY-plane in order to increase the range of motion of the hip joint. The distance between the hip and knee joint and between the knee and ankle are almost identical at 410 mm and 420 mm. The intersection of the ankle axes is 100 mm above the foot contact area. Four discrete foot contact points span a support polygon of 80 mm x 200 mm. In addition to force sensors in the individual foot contact points, an ATI 6-axis force-torque sensor is available that connects the foot to the ankle joint. A single leg weighs a total of 10.6 kg, of which 7 kg is on the hip and thigh, 2.3 kg on the lower leg and 1.3 kg on the foot. Similar to the arm and torso structures, the leg structures were also designed as aluminum casted parts.

15.2.1.7 Actuation

Depending on the respective speed and torque profile requirements, the various joint axes were configured as either serial or parallel kinematics. Serially arranged rotary actuators have been used on joint axes, where a particularly large range of motion had to be achieved. However, for joints with small range of motion demands, the advantages of parallel kinematics have been exploited. Parallel kinematics offer the possibility of a discontinuous transmission ratio to be adapted to the torque and speed requirements of the specific joint. This effect has been utilized in the joints for the forward movement of the legs (hip pitch, knee, ankle pitch) as well as in the

elbow and wrist joints of the robot. Another advantage of parallel kinematics is that the forces of the actuators can be superposed by arranging at least two linear actuators in parallel on a 2 DOF universal joint, which was used in the robot's ankle, wrist and hip joints, respectively, to amplify torque in the main direction of motion. Furthermore, parallel configurations can increase the joint stiffness and offer the possibility of shifting actuator masses closer to the center of the robot, in order to reduce the inertia of the entire extremities. For both concepts, modular drive units were developed. Drive units based on high-torque BLDC motors (TQ Systems) and Harmonic-Drive gears (HarmonicDrive) were developed for on axis rotary actuation of joints in serial chains. Actuator units of this type were designed in two sizes. A larger version, based on an ILM 70x10 RoboDrive motor and a CPL25 HarmonicDrive gear, is used in the first and second DOF of the shoulder joints, in the torso yaw joint and in the hip joints to actuate the yaw and roll axes. Similarly, a smaller version with an ILM 50x08 RoboDrive motor and a CPL 14 gearbox is used for the third shoulder and the wrist-roll joints. For the use in parallel kinematic joints, linear drive units consisting of a high-torque BLDC motor (TQ Systems) in combination with a ball screw (Eichenberger) were developed in three different sizes. Therein, the motor directly drives the nut of the spindle. Thus, the spindle itself does not rotate and moves along its axis of motion through the hollow shaft of the motor. Drives of this design are used in the hip joints (pitch), the body joint (pitch, roll), and the robot's knee, ankle, elbow, and wrist joints. An overview of ROM, torques and velocities of the linear actuators is given in Table 15.1. Commercially available servo drives (Dynamixel) were used to actuate the head joints.

Serial elasticities have been provided in all leg actuators, which can be used optionally. In the case of the linear actuators, these consist of two helical springs preloaded against each other, respectively. In the rotational actuators of the hip joint, torsion shafts were integrated, which allow a deflection of up to $\pm 5^\circ$ under peak torque. All elasticities can either be locked or replaced by stiff dummy parts, allowing non-elastic operation.

15.2.2 Electronic design

To control the RH5 Pedes robot a hybrid control approach that combines centralized control loops for high-level control and local control loops for low-level motor control is used.

1. Actuator-level decentralized controllers: Each of the individual actuators is controlled by separate electronics located close to the actuator, in

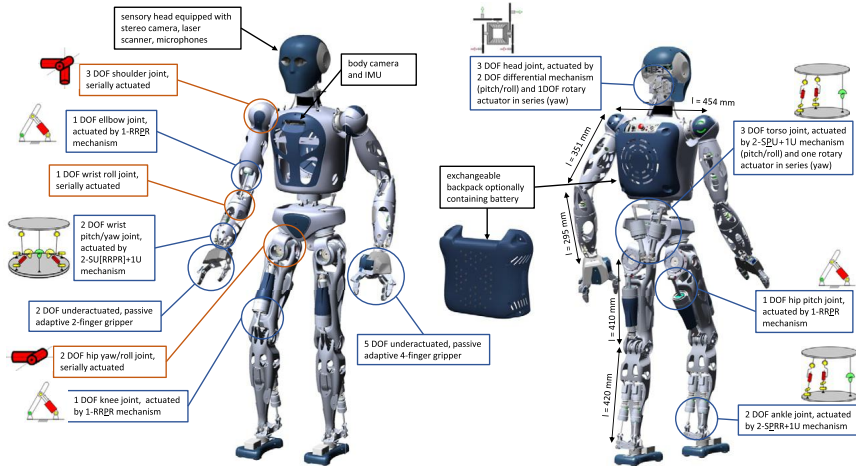


Figure 15.1 Actuation and morphology of the RH5 Pedes humanoid robot (S: Spherical, R: Revolute, P: Prismatic, U: Universal).

Table 15.1 ROM of linear actuators used in RH5 Pedes.

Actuator	ROM (mm)	Max. force (N)	Max. vel. (mm/s)
Wrist	[113, 178]	1094	200
Elbow	[173.8, 295.42]	2000	266
Torso	[205.8, 282.8]	2716	291
Hip3	[272, 431]	4740	175
Knee	[273, 391]	5845	140
Ankle	[221, 331]	2000	265

particular. From a hardware point of view, this modular approach simplifies the cabling effort, since it is sufficient to have common power lines for digital communication with the central controllers. The single electronics consists of one or two motor driver boards, a processing board based on a Xilinx Spartan 6 Field Programmable Grid Array (FPGA) and a board connecting sensors and communication wires. Furthermore, the hardware structure at the control level enables low-level decentralized control that allows low-latency local control loops. These local controllers are implemented as a cascade feedback controller for motor current, speed, and position, running at frequencies of 32 kHz, 4 kHz, and 1 kHz, respectively. In addition, the local controllers provide feedforward connections to the high-level controllers.

Table 15.2 ROM of the RH5 Pedes humanoid robot in its independent joint space (generalized coord. when robot is fixed).

Joint	ROM (°)	Max. Torque (Nm)	Max. Velocity (°/s)
Shoulder1	[−180°, 180°]	157	210
Shoulder2	[−14.5°, 106°]	157	210
Shoulder3	[−180°, 180°]	28	330
Elbow	[−105°, 0°]	48–172	177–633
Wrist Roll	[180°, 180°]	28	330
Wrist Pitch	[−42.5°, 100°]	29–56	364–696
Wrist Yaw	[−32°, 34°]	38–50	386–499
Torso Yaw	[−180°, 180°]	157	210
Torso Pitch	[−20°, 30°]	380–493	184–238
Torso Roll	[−25.5°, 25.5°]	285–386	208–400
Head Roll	[−25.5°, 25.5°]	8.2	275
Head Pitch	[−17°, 20°]	8.2	275
Hip 1	[−180°, 180°]	157	210
Hip 2	[−46°, 67°]	157	210
Hip 3	[−17°, 72°]	357–540	88–133
Knee	[0°, 88°]	337–497	94–139
Ankle pitch	[−51.5°, 45°]	121–304	200–502
Ankle roll	[−57°, 57°]	84–158	386–726

This enables for feedforward control of speed and motor current, hence the amount of feedback can be limited locally to achieve a desired compliant behavior. Note that joint position and velocity can be mapped between the independent joint space and actuator space locally, which is also needed to initialize the incremental encoder position offset of the motor when the absolute position sensor measures the independent joint position.

- Mid- and high-level central electronics control: Joint status and command messages are translated and routed between the actuators, sensors and a central control PC connected via an ethernet by a hybrid FPGA/ARM-based system. A routine to synchronize the translation layer to the command messages was implemented, to maximize the packets transmitted to the central control PC while ensuring an upper limit on the transmission delay. The ROCK Framework is used as the robot middleware on the control PC. The framework includes driver components, which handle the actuator setup and data exchange. It also provides a robot-independent interface to the software components im-

plemented in the higher-level control system. The driver components run periodically at a frequency of 1 kHz. The resulting lap triggering time is 1 ms.

15.3 Modeling and control

15.3.1 Robot modeling and control based on DDP

The motion generation of RH5 Pedes rely on the Optimal Control (OC), more precisely on the Differential Dynamic Programming (DDP) algorithm. The robot is an under-actuated multi-body system involving a floating base that allows it to move through its environment. The Equation of Motion (EOM) of this system is detailed below in order to subsequently formulate the Optimal Control Problem (OCP) used to control the robot.

15.3.1.1 Equations of motion

RH5 Pedes is a full-body humanoid robot with hybrid serial-parallel joints. To control this system, we use a tree-like abstraction model, ignoring the internal closed loop constraints in the trajectory optimization process. The tree abstraction model involves $6 + n$ DoFs with a floating base and $n = 32$ actuators. It is designed with a multi-body articulated structure and K contacts with the environment. The contacts are enforced through holonomic constraints. The dynamic EOM of this system is given below based on the Euler–Lagrange equations [13]:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T \boldsymbol{\tau} + \sum_{k=1}^K \mathbf{J}_k(\mathbf{q})^T \boldsymbol{\phi}_k, \quad (15.1)$$

where

- \mathbf{q} : configuration vector.
- $\mathbf{M}(\mathbf{q})$: inertia matrix.
- $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$: nonlinear and gravity forces.
- $\mathbf{S} = [0_{n \times 6} \quad I_{n \times n}]$: selection matrix of the actuation space.
- $\boldsymbol{\tau}$: actuation joint torques.
- $\mathbf{J}_k(\mathbf{q})$: selection Jacobian matrix for the k^{th} contact.
- $\boldsymbol{\phi}_k = [\boldsymbol{\lambda}_k \quad \boldsymbol{\eta}_k]^T$: vector of the external contact forces ($\boldsymbol{\lambda}_k$) and torques moment ($\boldsymbol{\eta}_k$).

15.3.1.2 Contact constraints stability

The biped robot is constantly in contact with the ground, these contacts are modeled in this work as a second-order kinematic constraint on the contact placement and orientation (6D contact). The k^{th} rigid contact constraint is defined by Eq. (15.2).

$$\mathbf{J}_k \ddot{\mathbf{q}} + \dot{\mathbf{J}}_k \dot{\mathbf{q}} = 0 \quad \forall k \in 1 \dots K \quad (15.2)$$

The EOM under contact constraints is then modified as follows [14]:

$$\begin{bmatrix} \mathbf{M} & \mathbf{J}_c^T \\ \mathbf{J}_c & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\dot{\phi}_c \end{bmatrix} = \begin{bmatrix} \mathbf{S}^T \boldsymbol{\tau} - \mathbf{b} \\ -\dot{\mathbf{J}}_c \dot{\mathbf{q}} \end{bmatrix}, \quad (15.3)$$

where

- $\mathbf{J}_c = [\mathbf{J}_0^T \dots \mathbf{J}_k^T \dots \mathbf{J}_K^T]^T$,
- $\dot{\phi}_c = [\dot{\phi}_0^T \dots \dot{\phi}_k^T \dots \dot{\phi}_K^T]^T$.

In order to improve the stability of the contact constraint and prevent sliding, the contact wrench cone [15] are implemented along with the Baumgarte stabilization gains [16].

The contact wrench cone were defined for each 6D contact constraint as inequality constraints in the OCP as follows [17]:

$$\begin{aligned} \lambda^z &> 0, \\ |\lambda^x| &\leq \mu \lambda^z, \\ |\lambda^y| &\leq \mu \lambda^z, \\ |X| &\geq c_x, \\ |Y| &\geq c_y, \end{aligned} \quad (15.4)$$

where

- μ : the static coefficient of friction,
- c_x and c_y : the position of the Center of Pressure (CoP) respectively to the robot feet dimensions X and Y .

15.3.1.3 Walking task as an optimal control problem

In this work, the walking task was selected to be studied as a use case to evaluate the motion capabilities of the robot. For this purpose, the robot's gait was transcribed as a multi-phase OCP under contact constraints. Based on the constrained dynamics presented above, a common formulation of

the gait OCP can be written as follows:

$$\min_{\mathbf{x}, \mathbf{u}} l_N(\mathbf{x}_N) + \sum_{t=0}^{N-1} l(\mathbf{x}_t, \mathbf{u}_t) dt, \quad (15.5a)$$

$$s.t. \quad \mathbf{x}_0 = \mathbf{f}_0, \quad (15.5b)$$

$$\forall i \in \{0 \dots N-1\}, \quad \mathbf{x}_{i+1} = \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i), \quad (15.5c)$$

where

- $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$: state of the robot,
- $\mathbf{u} = \boldsymbol{\tau} \in \mathbb{R}^{n_u}$: torque control of the robot,
- N : number of nodes of the complete trajectory,
- l_N : terminal cost model of the terminal node,
- l : running cost model applied to all other nodes,
- \mathbf{f}_0 : the initial state of the trajectory,
- \mathbf{f}_i : the robot dynamic discretization.

In this work, we choose to rely on shooting methods [18] using the DDP [14] algorithm to solve such OCPs. In particular, a modified version of DDP is used, the BoxFDDP [19] formulation. The BoxFDDP solver allows us to overcome the single-shot numerical limitations of the original DDP formulation, while constraining the robot torque control limits. To this end, the open-source framework Crocoddyl [20] C++ Library was used. Crocoddyl relies on the open-source C++ library Pinocchio [21][22] for its dynamic computations, which allows us an efficient and recursive resolution throughout the optimization process.

15.3.1.4 Cost model definition

The walking gait OCP is formulated with the same cost model and discretization time for each gait cycle phase. The running cost model is defined in Eq. (15.6).

$$l = \sum_{n=1}^{T-1} \alpha_n \Phi_n(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}), \quad (15.6)$$

l is formulated by summing the following costs, t_S represents the final time of each gait cycle.

- *2D-CoM Cost*: the CoM placement $\mathbf{c}(t)$ tracks the final desired CoM position in the (x, y) axis for each gait cycle $\mathbf{c}^{\text{ref}}(t_S)$.

$$\Phi_1 = \|\mathbf{c}(t) - \mathbf{c}^{\text{ref}}(t_S)\|_2^2$$

Table 15.3 Motion characteristics and weights of the optimization constraints.

Walking	Characteristics			Optimization Constraints					
	Length (m)	Height (m)	Time (s)	Tasks		Stability	Limits	Regularization	
				Φ_{foot}	Φ_{CoM}	Φ_{friction}	Φ_{joint}	Φ_{posture}	Φ_{τ}
Static	0.1	0.05	1.8	$1e^8$	$1e^6$	$1e^2$	$1e^6$	$1e^1$	$1e^{-1}$
Dynamic	0.8	0.05	1.8	$1e^8$	$1e^6$	$1e^2$	$1e^0$	$1e^1$	$1e^{-1}$

- *Feet Cost*: the swinging foot translation $r_{\text{foot}}(t)$ tracks the desired final foot placement for each gait cycle.

$$\Phi_2 = \|r_{\text{foot}}(t) - r_{\text{foot}}^{\text{ref}}(t_s)\|_2^2$$

- *Contact friction cone*: at each contact foot a friction cone cost is added as defined in (15.4),
- *Torque regularization*: optimizes the joint torque control for a feasible dynamic gait.

$$\Phi_4 = \|\tau(t)\|_2^2$$

- *Posture regularization*: which manages the redundancy of the multi-body dynamics. The initial posture of the robot is used as reference \mathbf{q}^{ref} .

$$\Phi_5 = \|\mathbf{q}(t) - \mathbf{q}^{\text{ref}}\|_2^2$$

The terminal cost model at $T = t_{\text{final}}$, the final time of the full generated gait, is the same as the running cost model, except that the torque regularization cost is not included in the cost model. The weights α_i of the OCP are experimentally determined and listed in Table 15.3 for the case of static and dynamic walking.

15.3.1.5 Whole-body control

Once the optimal control motion is obtained, this have to be reproduced on the robot. Ideally, it is possible to directly feed the optimal joints references to PD controllers with high gains; by using this kind of position controller, the robot will *forcely* track the joint trajectories. However, this is not enough to ensure a correct and safe execution of the desired motion. Without any feedback, the robot will be very stiff due to the high gains and incapable of reacting to external disturbances; in addition, fast motions or unmodeled

dynamics could lead to undesired falls. For these reasons, there is a need of an online controller.

Considering the underactuated nature of a humanoid robot and the high number of DoFs, whole-body control emerges as a natural choice for online optimization and stabilization of a complex behavior such as humanoid walking. In fact, this approach aggregates multiple tasks that are simultaneously optimized in a single problem (or a cascade of problems) and provide a solution for the entire system. In this way, we can have a single controller that handles CoM/CoP positioning, Cartesian pose of different body parts, posture regularization and actuation limits.

More information about whole-body control is provided in chapter on whole-body control, which presents an exhaustive overview of different WBC variations and use cases. Here, the WBC problem has been formulated as a quadratic problem (QP) using Task Space Inverse Dynamics (TSID) [23], allowing for simultaneous optimization of contact forces, joints' accelerations and torques:

$$\min_{\mathbf{x}=\dot{\mathbf{q}},\boldsymbol{\tau},\boldsymbol{\lambda}} \mathbf{T}^r + \sum_i \alpha_i \mathbf{T}_i, \quad (15.7a)$$

$$s.t. \quad \mathbf{M}\ddot{\mathbf{q}} + \mathbf{b} = \mathbf{S}^\top \boldsymbol{\tau} + \sum_{k=1}^K \mathbf{J}_k(\mathbf{q})^\top \boldsymbol{\phi}_k, \quad (15.7b)$$

$$\forall k \in K \quad \mathbf{J}_k \ddot{\mathbf{q}} + \dot{\mathbf{J}}_k \dot{\mathbf{q}} = 0, \quad (15.7c)$$

$$\mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}, \quad (15.7d)$$

$$\mathbf{U}\boldsymbol{\lambda} \geq 0. \quad (15.7e)$$

The cost function (15.7a) consists of a regularization term \mathbf{T}_r and weighted sum over different task functions \mathbf{T}_i which include:

- a centroidal angular momentum term $\mathbf{T}_a = \|\mathbf{A}\ddot{\mathbf{q}} + \dot{\mathbf{A}}\dot{\mathbf{q}} - \dot{\mathbf{h}}^*\|_2^2$, where matrix \mathbf{A} is the angular part of the centroidal momentum matrix [24], $\dot{\mathbf{A}}$ is the corresponding time derivative and \mathbf{h} is the angular centroidal momentum reference.
- a tracking term for each Cartesian task $\mathbf{T}_c = \|\mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{a}^*\|_2^2$, where \mathbf{J} is the task spatial Jacobian and \mathbf{a}^* is the Cartesian acceleration reference. The SE3 task can be masked in order to select only the desired axis A special case of Cartesian task is given by the CoM task; in that case the task Jacobian is equivalent to the CoM Jacobian.
- a postural task tracking term $\mathbf{T}_p = \|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^*\|_2^2$ which helps to maintain a generally good posture despite the system redundancy.

Because of the regularization term $\mathbf{T}_r = \|\alpha_{\dot{\mathbf{q}}}\ddot{\mathbf{q}}\|_2^2 + \|\alpha_\lambda \boldsymbol{\lambda}\|_2^2$, the QP solver will prefer solutions with smaller acceleration, torques and forces; during the simulation, the regularization values $\alpha_{\dot{\mathbf{q}}}$ and α_λ have been set to $1e-8$ and $1e-4$ respectively.

The QP problem is constrained by the articulated system dynamics (15.7b), the 6D contacts constraint (15.7c), the actuation/force limits (15.7d), and the friction cone constraint (15.7e), representing Eq. (15.4).

At each control step, the reference values for the Cartesian tasks and the postural task are computed via feedback PD controllers:

$$\mathbf{a}^* = \mathbf{a}_d + \mathbf{K}_d(\mathbf{v}_d - \mathbf{v}) + \mathbf{K}_p(\mathbf{p}_d - \mathbf{p}), \quad (15.8)$$

$$\ddot{\mathbf{q}}^* = \ddot{\mathbf{q}}_d + \mathbf{K}_d(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q}). \quad (15.9)$$

Feet contacts are removed once the optimal control force reference along the z -axis are below a predefined threshold $\lambda_z \leq \mathbf{f}_{rem}$. On the contrary, foot contacts are added again to the QP formulation when the force reference is over a predefined threshold \mathbf{f}_{add} and the absolute Cartesian tracking error is below 1 mm. The contact force threshold values $\mathbf{f}_{rem} \leq \mathbf{f}_{add}$ are chosen differently in such a way to avoid multiple consecutive contact changes in case the force reference presents oscillations.

In order to further stabilize the gait, an additional ankle stabilization method have been used along with the whole-body controller. The ankle admittance control [25,26] guides the measured center of pressure \mathbf{p}_m of each foot toward the desired one \mathbf{p}_d by adjusting the ankle' roll and pitch angles (θ_r, θ_p) in a compliant way. Considering a measured contact wrench $\mathbf{w}_m = [\mathbf{f}'_m, \boldsymbol{\tau}'_m]'$ expressed in the foot frame, the ankle admittance controller will reduce contact torques at the desired CoP point by updating the *commanded* joints positions and velocities as

$$\begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_p \end{bmatrix} = \begin{bmatrix} A_{cop,r} & 0 & 0 \\ 0 & A_{cop,p} & 0 \end{bmatrix} \left({}^f\mathbf{p}_d \times \mathbf{f}_m - \boldsymbol{\tau}_m \right). \quad (15.10)$$

The superscript f over the desired CoP position indicates that the value is expressed in the foot frame. Gains $A_{cop,r}$ and $A_{cop,p}$ regulate how reactive the CoP adjustment is on the y - and x -axis respectively.

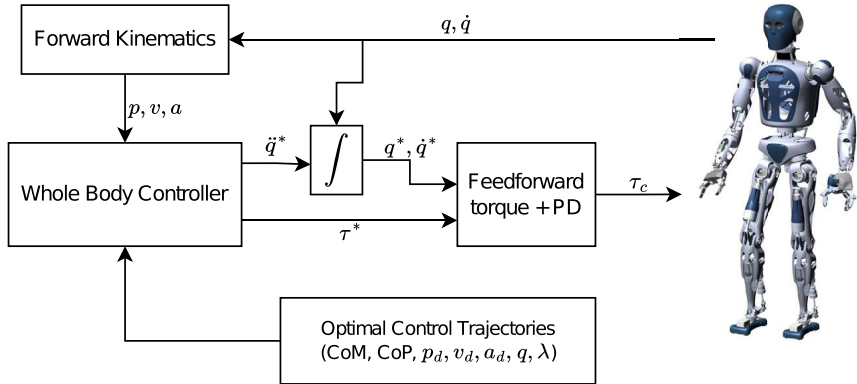


Figure 15.2 Control scheme for the whole-body controlled gait stabilization.

15.3.1.6 Simulation and implementation scheme via whole-body control

Kinematics and dynamics of RH5 Pedes have been validated in a simulated environment through the execution of a dynamic walk behavior. The simulation has been implemented using RAISIM [27]. The whole-body controller torques are sent to joints of the simulated RH5 Pedes via a feedforward torque plus PD controller. The velocity and position targets are computed by integrating the current joints' state with the WBC joint accelerations. For sake of simplicity, the proportional gain was set to the constant value 200 and the derivative gains was equal to 15 for each actuated joint.

The simulation and the controller execution is performed independently inside Rock [28] components running at 1 kHz. The control scheme for of online stabilization is pictured in Fig. 15.2. An overview of every task specification used during the gait experiment is provided in Table 15.4. The dynamic gait problem has been formalized using Cartesian tasks for feet and grippers. More weight has been given to the former, since correct feet placement is key while performing a gait motion. Grippers instead can move away from the reference trajectory more easily so that the arms can be used to stabilize the motion. Additionally, an orientation task has been used on the torso to keep it upright with a more natural pose. All weights have been hand-tuned, however more advanced techniques as tasks learning [29] or automatic WBC controllers selection [30] can be used to obtain better results in a systematic and faster way.

Table 15.4 Left: specification of tasks included in the WBC formulation for the walking behavior of RH5 Pedes; the momentum task used tow different weight for the dynamic walk (d) and the static walk(s). Right: gains for ankle admittance stabilization.

Task	α	K_p	K_d		
CoM	2000	150	20		
Feet	600	300	35		
Torso	20	30	10		
Head	5	30	10		
Grippers	5	30	10		
Posture	0.175	30	10		
Momentum (d/s)	0.1/10.0	1	1		
				$A_{cop,r}$	$A_{cop,p}$
				1e-1	5e-1

15.4 Results and discussion

15.4.1 Results

In order to evaluate the capabilities and human likeness of the RH5 Pedes robot, a static walk trajectory and a more challenging dynamic walk trajectory of 3 walking cycles were generated using OC then stabilized with WBC. The two motions have been generated using the same set of tasks' weights, but have different posture for the arms. Because of the larger lateral motion of the hip during the static walk, the elbows are bent in order to have greater distance from it. The resulting OC simulation for the static and dynamic walks can be observed in Fig. 15.3 and Fig. 15.4 respectively. These qualitative visual results show that the robot performs three successive walking cycles. These trajectories were given to the WBC for online stabilization.

Fig. 15.5 and Fig. 15.6 show respectively the execution of static and dynamic gaits in the simulated environment; tracking results are depicted in Fig. 15.7. RH5 Pedes is able to follow the OC trajectories and to complete successfully the motions.

Instabilities at the ankles level has been observed during motion execution, causing larger error along the y -axis of the CoM profile while performing the dynamic walk. Feet are successfully stabilized using the ankle admittance method, however, since it works by directly updating the commanded references, WBC cannot anticipate its influence but has to compensate in the following control iteration. The ankle behavior can be a consequence of the differences between the optimal control simulation and the simulated environment used for online stabilization, especially for

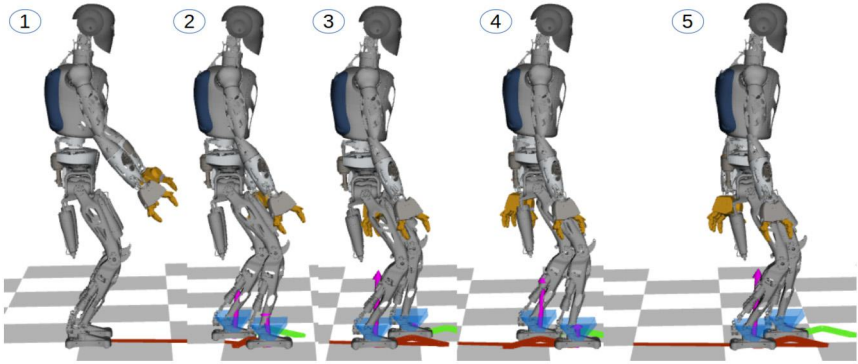


Figure 15.3 Static walking motion with RH5 Pedes robot.

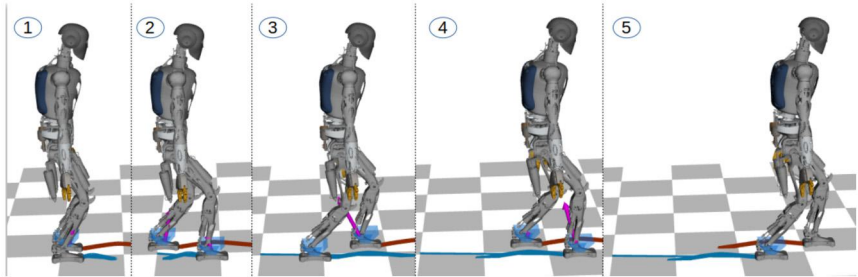


Figure 15.4 Dynamic walking motion with RH5 Pedes robot.

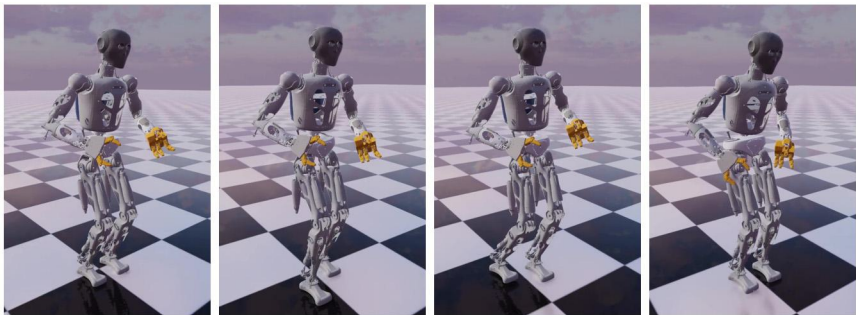


Figure 15.5 From right to left: RH5 Pedes executing a static walk in a simulated environment using WBC.

what concern contacts computation. The sim-to-sim gap is just a fraction of what a sim-to-real difference can be, but it already proves the need for online stabilization and assesses the benefits of whole-body control.



Figure 15.6 From right to left: RH5 Pedes executing a dynamic walk in a simulated environment using WBC.

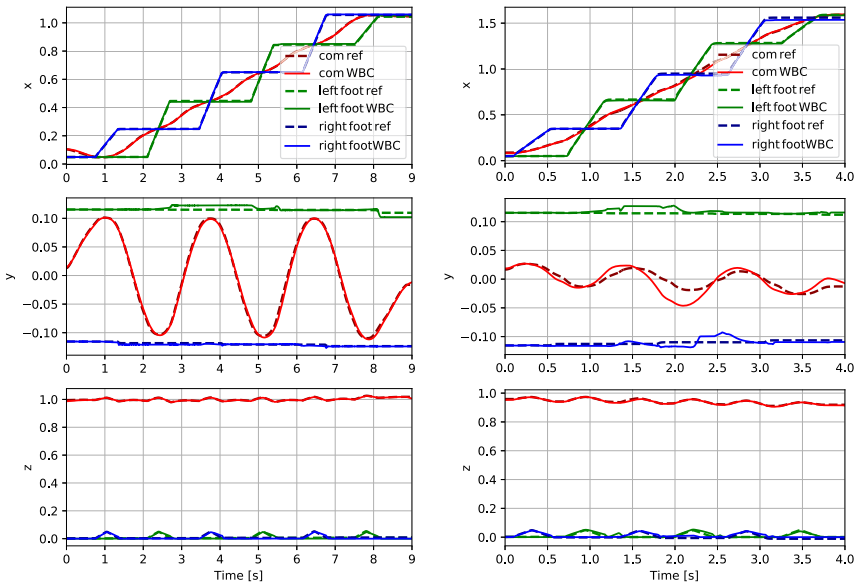


Figure 15.7 CoM and feet tracking using whole-body control during the on-line execution of a 3 gait cycles static walk (left) and a 3 gait cycles dynamic walk (right).

15.4.2 Discussion

In this work, the RH5 Pedes humanoid robot was presented. The design choices were explained and justified. This design was assessed through various generated motions. In this study, the presented motions are whole-body successive walks. The trajectories were generated using OC and stabilized

on-line within a physical simulator using a WBC framework. The obtained results show that the anthropomorphism choices allowed us to generate successfully walking cycles. However, more dynamic movements, such as climbing stairs or jumping, should be generated in the future to challenge and improve the robot's performances. The generated walking steps should be faster to challenge the stabilization of the robot, while including some uncertainties in the ground to make our control more robust. To this purpose, an MPC controller should be implemented to control the real robot in a natural environment containing uneven terrain, for example. The robot's motor design could be modified by decreasing the robot's payload capabilities while increasing the joint speed limits. This modification will allow us to perform a wider range of human-like movements. The design of the robot's feet should also be enhanced in the future, to ensure better stability and adherence to the ground. In this chapter, the considered model is a tree-like abstraction of the full robot. This model ignores the internal closed-loop constraints of the robot, which restricts us in the exploitation of the full capabilities of the robot.

Online trajectory stabilization using WBC could be enhanced via optimized selection of tasks' weights and priorities. Moreover, joint wise PD gains tuning can improve the robot's compliance while keeping comparable performance.

In the future, state estimation should be introduced in the control loop, as it will be essential for a robust stabilization of the real robot. In order to ensure a safe and comfortable human-robot interaction, the image processing via the robot's cameras should be exploited and introduced into the online control.

15.5 Conclusion

In this chapter, the RH5 Pedes robot was presented. Its mechanical design, based on human anthropomorphism, was detailed and justified, with some possible improvements for the future. A gait task has been performed by the robot in order to assess its capabilities while highlighting the next steps for further developments.

References

- [1] A.K. Pandey, R. Gelin, A mass-produced sociable humanoid robot: Pepper: the first machine of its kind, *IEEE Robotics & Automation Magazine* 25 (3) (2018) 40–48.

- [2] J. Złotowski, D. Proudfoot, K. Yogeewaran, C. Bartneck, Anthropomorphism: opportunities and challenges in human–robot interaction, *International Journal of Social Robotics* 7 (3) (2015) 347–360.
- [3] B.R. Duffy, Anthropomorphism and the social robot, *Robotics and Autonomous Systems* 42 (3–4) (2003) 177–190.
- [4] J. Fink, Anthropomorphism and human likeness in the design of robots and human–robot interaction, in: *International Conference on Social Robotics*, Springer, 2012, pp. 199–208.
- [5] C. Bartneck, T. Bleeker, J. Bun, P. Fens, L. Riet, The influence of robot anthropomorphism on the feelings of embarrassment when interacting with robots, *Paladyn* 1 (2) (2010) 109–115.
- [6] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 01–07.
- [7] M. Boukheddimi, R. Kumar, S. Kumar, J. Carpentier, F. Kirchner, Investigations into exploiting the full capabilities of a series–parallel hybrid humanoid using whole body trajectory optimization, in: *2023 IEEE/RISJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023.
- [8] M. Boukheddimi, D. Harnack, S. Kumar, R. Kumar, S. Vyas, O. Arriaga, F. Kirchner, Robot dance generation with music based trajectory optimization, in: *2022 IEEE/RISJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022.
- [9] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: J. Lenarcic, V. Parenti-Castelli (Eds.), *Advances in Robot Kinematics* 2018, 2019, pp. 431–439.
- [10] J. Esser, S. Kumar, H. Peters, V. Bargsten, J. de Gea Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series–parallel hybrid RH5 humanoid robot, in: *2020 IEEE–RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021, pp. 400–407, <https://doi.org/10.1109/HUMANOIDS47582.2021.9555770>.
- [11] C. Stoeffler, A. del Rio Fernandez, H. Peters, M. Schilling, S. Kumar, Kinematic Analysis of a Novel Humanoid Wrist Parallel Mechanism, in: *Advances in Robot Kinematics*, Springer Proceedings in Advanced Robotics, 2022.
- [12] N. Mulsow, P. Kampmann, Underactuated gripper design for the assembly of infrastructure in space, in: *14th International Symposium on Artificial Intelligence, (ISAIRAS–2018)*, 2018.
- [13] R. Featherstone, *Rigid Body Dynamics Algorithms*, Springer US, 2014, <https://books.google.fr/books?id=GJRGBQAAQBAJ>.
- [14] R. Budhiraja, J. Carpentier, C. Mastalli, N. Mansard, Differential dynamic programming for multi-phase rigid contact dynamics, in: *IEEE–RAS Humanoids 2018*, Beijing, China, 2018.
- [15] S. Caron, Q.–C. Pham, Y. Nakamura, Stability of surface contacts for humanoid robots: closed-form formulae of the contact wrench cone for rectangular support areas, in: *ICRA*, IEEE, 2015.
- [16] J. Baumgarte, Stabilization of constraints and integrals of motion in dynamical systems, *Computer Methods in Applied Mechanics and Engineering* 1 (1) (1972) 1–16.
- [17] J. Esser, Highly-dynamic movements of a humanoid robot using whole-body trajectory optimization, Master’s thesis, University of Duisburg–Essen, Nov 2020.
- [18] J. Nocedal, S. Wright, *Numerical Optimization*, Springer Science & Business Media, 2006.
- [19] N. Mansard, Feasibility-prone differential dynamic programming DDP a multiple shooting algorithm, Online, available <https://github.com/loco-3d/crocodyl/tree/master/doc/reports/fddp>, 2020.

- [20] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, S. Vijayakumar, N. Mansard, Crocodyl: an efficient and versatile framework for multi-contact optimal control, in: ICRA, Detroit, USA, 2020.
- [21] J. Carpentier, F. Valenza, N. Mansard, et al., Pinocchio: fast forward and inverse dynamics for poly-articulated systems, <https://stack-of-tasks.github.io/pinocchio>, 2015–2021.
- [22] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, N. Mansard, The Pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives, in: SII, IEEE, 2019.
- [23] D. Andrea, M. Nicolas, R. Oscar, S. Olivier, N. Francesco, Implementing torque control with high-ratio gear boxes and without joint-torque sensors, *International Journal of Humanoid Robotics* 13 (1) (2016) 1550044, <https://hal.archives-ouvertes.fr/hal-01136936/document>.
- [24] D.E. Orin, A. Goswami, Centroidal momentum matrix of a humanoid robot: structure and properties, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 653–659, <https://doi.org/10.1109/IROS.2008.4650772>.
- [25] S. Caron, A. Kheddar, O. Tempier, Stair climbing stabilization of the HRP-4 humanoid robot using whole-body admittance control, in: 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 277–283, <https://doi.org/10.1109/ICRA.2019.8794348>.
- [26] S. Kajita, H. Hirukawa, K. Harada, K. Yokoi, *Introduction to Humanoid Robotics*, Springer, Berlin, Heidelberg, 2014.
- [27] J. Hwangbo, J. Lee, M. Hutter, Per-contact iteration method for solving contact dynamics, *IEEE Robotics and Automation Letters* 3 (2) (2018) 895–902, www.raisim.com.
- [28] ROCK, the Robot Construction Kit, <http://www.rock-robotics.org>.
- [29] M. Charbonneau, V. Modugno, F. Nori, G. Oriolo, D. Pucci, S. Ivaldi, Learning robust task priorities of QP-based whole-body torque-controllers, in: 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), 2018, pp. 1–9, <https://doi.org/10.1109/HUMANOIDS.2018.8624995>.
- [30] E. d’Elia, J.-B. Mouret, J. Kober, S. Ivaldi, Automatic tuning and selection of whole-body controllers, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2022, hal-03740361.

This page intentionally left blank

CHAPTER 16

ARTER: a walking excavator robot[☆]

Ajish Babu, Pierre Willenbrock, Jannik Tiemann, Felix Bernhard, and Daniel Kühn

Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

16.1 Introduction

Heavy work machines, including excavators, wheel loaders, and tractors, have undergone a continuous process of evolution since their initial invention. The areas of application for these machines have also expanded in conjunction with their ongoing development. To date, teleoperation, advanced driver assistance functions, and limited autonomy features have been employed to some extent with these vehicles. As research in the domains of robotics and artificial intelligence progresses rapidly, the potential for more immersive teleoperation and a higher level of autonomy, which are necessary for operation in challenging environments, also increases. One of the primary environmental challenges for heavy-duty vehicles and mobile robots is navigating unstructured terrain with a multitude of obstacles and traversing high-sloped areas with diverse substrates.

To address the challenges posed by unstructured terrain, walking excavators have been developed. These are essentially excavators with the capability to walk. In comparison to traditional excavators, walking excavators offer enhanced mobility. Such vehicles are capable of traversing expansive ditches, elevating themselves onto elevated platforms, and are currently being utilized in mountainous regions to construct infrastructure, thereby facilitating access to otherwise inaccessible terrain. The German Federal Agency for Technical Relief (THW) employs these devices under the designation “rescue spider” in a multitude of applications where standard excavators are constrained in their functionality. One notable instance

[☆] This work is conducted within the project ROBDEKON, funded by the Federal Ministry of Education and Research (Grant Nr. 13N14675).

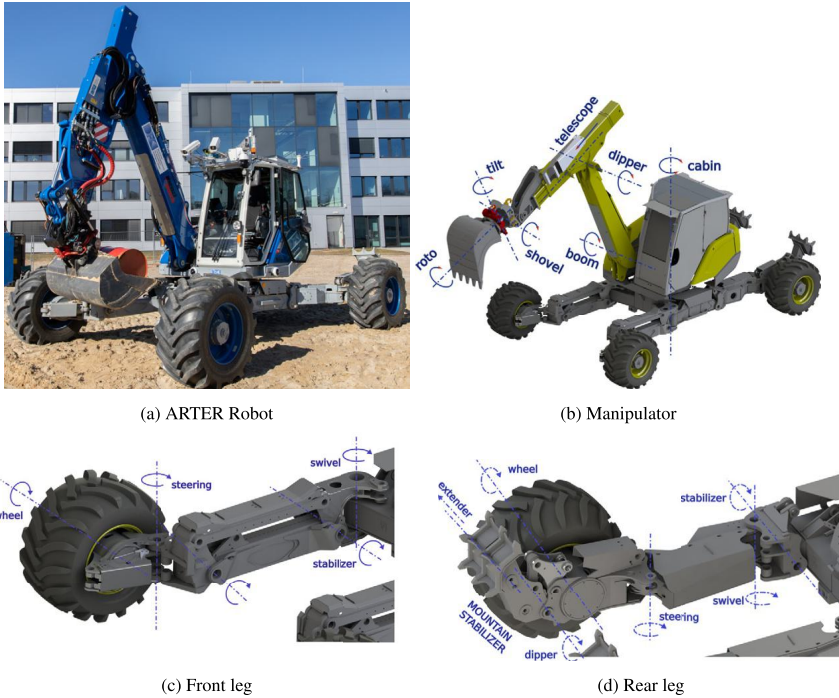


Figure 16.1 Kinematics of ARTER legs showing the joints and the parallel kinematics.

occurred in 2021 in the flooded Ahrtal region of Germany. This is merely a brief overview of the applications of these systems.

A walking excavator, which has a wheel-on-leg design, is capable of adjusting the position of each individual wheel by moving its legs, thus adapting to unstructured terrain. The excavator's hydraulics (typically actuating the parallel linkages) are configured in a manner that enables the manipulator to lift the excavator itself. Consequently, extreme inclines or hillsides can be traversed using such vehicles.

Similar systems with adaptive suspension are widely used as research platforms on uneven terrain. SherpaTT ([1]), ATHLETE ([2]), MAMMOTH ([3]), and ANYmal with Wheels ([4]) are some examples of mobile robotics with a wheel-on-leg design. Jud et al. describe the design and development of the walking excavator robot HEAP ([5]), which has been retrofitted with custom developed hydraulic valves to control the joints. HEAP robot has the ability to adapt to the terrain using these valves.

The ARTER system (Autonomous Rough Terrain Excavator Robot), as shown in Fig. 16.1a, is based on Menzi Muck M545 walking excavator, which is an excavator with excellent off-road mobility. The robot is being developed by DFKI as part of the competence center, ROBDEKON (Robotic Systems for Decontamination in Hostile Environments) [6], which is responsible for the research of autonomous or semi-autonomous robotic systems, considering both nuclear and waste site decontamination scenarios. This work describes the ARTER robot with focus on tasks that were undertaken to retrofit and automate the base vehicle.

The chapter is organized as following. The remainder of this section describes the problem, the biological inspiration and the application scenarios. The mechatronic system design, including the mechanical, electronic, and software designs are detailed in Section 16.2. The modeling of the parallel kinematic and the control structures are described in Section 16.3. This is followed by conclusion and related discussions in Section 16.4.

16.1.1 Problem description

In order to equip the vehicle with the functionalities that are typical for a robot, it is necessary to retrofit it. In the original control configuration, the operator is required to control each actuator valve individually via the joystick. Consequently, the desired motion of the end-effector is difficult to achieve, necessitating the coordinated movement of multiple joysticks and pedals simultaneously. The presence of numerous non-linearities, including friction, hydraulic components, and parallel kinematic constraints, between the joystick signal and the end-effector movement renders automation a challenging endeavor. Furthermore, the implementation of additional sensors and valves is essential for the estimation of the robot's state and the automated control of its joints.

It is imperative to consider additional safety factors during operation, particularly in the context of remote operation. The adaptive suspension of the chassis introduces concerns with regard to tip-over stability, which must be addressed from a safety perspective. The extensive range of postures that can be achieved with this system presents a challenge for the operator in estimating the stability of the vehicle and ensuring that the wheels are properly in contact with the ground. In some instances, the operator may overestimate the stability, which can result in fatal accidents. Additionally, avoiding collisions with both internal and external objects is a significant concern for the operator. Factors such as occluded objects or operator inattention could contribute to these collisions.

16.1.2 Biological inspiration

The use of adaptive suspension and use of manipulator for locomotion gives different locomotion modes for the robot. The locomotion modes can be generally classified as

- (i) *Driving*, which uses only the wheels on uneven ground,
- (ii) *Climbing*, where the manipulator is used to support driving on terrain with steep or slippery slopes, and
- (iii) *Stepping*, where the manipulator lifts the wheels from the ground and step over obstacles.

In addition to wheeled movements, the above modes makes extensive use of legged locomotion as well, which is inspired from animals. Use of limbs for object handling and locomotion, as is the case in modes *Climbing* and *Stepping*, is also widely seen in biological movements.

16.1.3 Application scenarios

Heavy-duty machinery has been utilized in challenging operational settings for an extended period. While the drivers are protected to an extend within the machine, the residual risk to their lives and health remains a concern that must be addressed. One such example of a hostile environment can be traced back to the establishment of landfills for special waste, where the potential future impact of such a site was not adequately considered. The majority of landfills constructed during the 1960s contain a considerable quantity of chemicals and toxins, the majority of which are unidentified. Some of these substances were stored, while others were illegally deposited into shafts and conduits or placed in brittle barrels. The leakage of these hazardous materials represents a significant environmental hazard, as well as a risk to the quality and safety of the surrounding groundwater. To illustrate, one might cite the hazardous waste landfills in Bonfol, Kesslergrube, and Morgenstern. A multitude of other such dumpsites exist, each presenting a distinct set of challenges. In some instances, operational procedures such as sampling are still conducted manually by personnel wearing full protective hazmat suits. It is imperative that practical solutions be implemented without delay to eliminate the necessity for human presence in such zones. Given the variability of the environment and the diversity of tasks, it is not feasible to conduct fully autonomous environmental exploration and decontamination in many cases. Nevertheless, to obviate the necessity for humans to place themselves in peril by entering contaminated zones, tele-operated control of the robots is indispensable.

The utilization of robots in deconstruction and/or decontamination can effectively alleviate the physical burden on humans while simultaneously providing enhanced protection from toxic or contaminated substances. In particular, the removal of material is generally accompanied by the emission of harmful dusts, the contact of which with humans can be difficult to avoid during manual work. In the context of on-site robots, specific requirements are imposed with regard to their mobility, robustness, and artificial intelligence (AI), particularly with respect to the safe interaction in hybrid teams, in order to achieve the necessary autonomy and thus the systems' ability to act. The prerequisites for these developments are already in place. Industrially deployed robots are becoming mobile and are beginning to perceive their environment with the help of multimodal sensors. They are also beginning to act autonomously in environments that are hostile to humans, albeit in their infant stages. Furthermore, they are learning with the help of state-of-the-art learning methods, which allows them to adapt optimally to the prevailing conditions in each case.

The ARTER system is primarily designed to perform its tasks and missions in toxic waste landfills and toxic waste deposition sites. However, it can also be used in other contexts, such as on industrial sites or on locations where search and rescue skills are required. The majority of work involved in the dismantling of disposal sites is typically conducted by humans wearing chemical protective gear. It is important to note that the protective gear in question does not guarantee 100% protection. The preferred approach in dealing with such risks is to focus on developing solutions that can perform the necessary tasks without endangering human personnel in the field. In general, the process of working at a waste site begins with the exploration and measurement of the surrounding area, continues with the collection and analysis of soil samples, and then progresses to the sorting and relocation of the various materials on a large scale. Frequently, the work is subject to additional disruptions, such as fires or explosions (for instance, due to contact with phosphorus materials), the discovery of Second World-War ammunition or other unexpected items.

16.2 Mechatronic system design

This section describes the mechanical design of the system along with the hydraulic, electronic and electrical retrofitting that were undertaken to automate it. The software design is also briefly discussed.

16.2.1 Mechanical design

The ARTER system, which weighs around 13 tons, has a total of 27 degrees of freedom. The chassis forms the base to which five kinematically different structures are connected: *Manipulator*, two *Front Legs*, and two *Rear Legs*.

The *Manipulator* consists of seven joints: *Cabin*, *Boom*, *Dipper*, *Telescope*, *Shovel*, *Tilt*, and *Roto*, as shown in Fig. 16.1b. The drive cabin is connected to the chassis through the *Cabin* joint, which can turn endlessly. This is the primary joint for making the end-effector move laterally. The joints *Boom*, *Dipper*, and *Telescope* move the end-effector in the vertical and longitudinal directions. The joints *Shovel*, *Tilt*, and *Roto* primarily contribute to the angular motion of the end-effector. The additional attachment, Rototilt, provides the *Tilt* and *Roto* joints.

The *Front Legs*, as shown in Fig. 16.1c, have four joints *Swivel*, *Stabilizer*, *Steering*, and *Wheel*. The *Swivel* joint moves the wheel in the lateral direction and the *Stabilizer* in the vertical direction. The *Stabilizer* joint has a parallelogram-like parallel kinematic ensuring that the wheels do no pitch. The *Steering* joint orients the wheel in the desired direction for steering and the *Wheel* joint rotates to give the longitudinal velocity for the vehicle.

The *Rear Legs*, as shown in Fig. 16.1d, have three main differences compared to the *Front Legs*. The *Stabilizer* joint does not have the parallelogram structure and hence the wheels can have pitch angles. The *Stabilizer* joint comes first in the kinematic tree, followed by the *Swivel* joint. There is also an additional structure, *Mountain Stabilizer*, connected to the *Steering* link. This structure is used to anchor to the ground for stability on high slopes and has two joints: *Claw Rotate* and *Claw Telescope*.

Hydraulics retrofit

The original vehicle is equipped with a electronic control system for the hydraulics with only limited capabilities. In the upper carriage, all actuators are controlled with hydraulic pilot valves. In other words, the joysticks or foot pedals control a hydraulic pressure, which in turn controls the hydraulic flow to the hydraulic cylinders of the manipulator and the hydraulic motor responsible for cabin rotation. In the lower carriage, all cylinders are controlled by the integrated programmable logic controller (PLC). The steering cylinders are equipped with proportional valves, which allow for continuous modulation of the hydraulic flow between its fully off and fully on positions. In contrast, all other cylinders are configured as on/off valves. The extension and rotation of the mountain stabilizers is controlled by a

Table 16.1 Joint hydraulic control valves and sensors. Original components are marked as “o”, retrofitted as “x”, and not available as “-”.

Joints	Hydraulic Valve				Sensors		
	Proportional	Proportional Pilot	Direction Control	Linear at actuator	Rotational at Joint	Rotational at related Joint	Pressure
Cabin	-	x	-	-	x	-	-
Boom	-	x	-	-	x	-	x
Dipper	-	x	-	x	-	-	x
Telescope	-	x	-	x	-	-	x
Shovel	-	x	-	-	-	x	x
Tilt	o	-	-	x	-	-	-
Roto	o	-	-	-	o	-	-
Stabilizers	x	-	-	x ^{-a}	x ^{-a}	-	x
Swivel	-	-	o	-	o	-	-
Steering	o	-	-	-	o	-	-
Wheel	o ^b	-	-	-	x	-	-
Claw rotate	-	-	o	-	x ^c	-	-
Claw telescope	-	-	o	-	-	-	-

^a Linear rear/rotational front.
^b All wheels using a common adjustable pump.
^c End stops.

single valve on either side, which is connected to the requisite cylinder by a hydraulic pressure line.

To the extent feasible, the original control structure was maintained to facilitate seamless transition between the system’s behavior and the control scheme, wherein a computer controls the majority of the actuators. This resulted in a methodology whereby hydraulic or electronic “switches” were incorporated at pivotal points to facilitate the continued operation of the original control structure. The transition to computerized control necessitates the concurrence of both the operator and the computerized system.

As illustrated in Table 16.1, the upper carriage incorporates hydraulic switches that facilitate the substitution of control by joysticks/pedals with

proportional valves. The majority of the existing electronically controllable valves were retained, with the exception of those installed in instances where a non-proportional directional valve was installed and did not provide the requisite level of control. In such instances, the incorporation of hydraulic switches and supplementary proportional valves was deemed necessary. In all other instances where electronically controllable valves were present, relays were employed to redirect the control of the valves from the built-in PLCs (indicated by the letter “o” in Table 16.1). The RotoTilt is a unique case, as the RotoTilt PLC already provides remote control capabilities, eliminating the need for any modifications.

In addition to the control of the hydraulic actuators, a number of sensors were installed with the purpose of determining the positions of all joints, as well as the forces at selected joints. In some cases, a linear sensor was installed on the hydraulic cylinder, while in others a rotational sensor was placed on the relevant joint or a related joint, as illustrated in Table 16.1. Additionally, the few sensors that were already in place (marked “o” in Table 16.1) were utilized further.

16.2.2 Electronic design

In order to facilitate autonomous control, an extensive suite of electronics has been integrated into the ARTER system. This comprises electromagnetic solenoids, which are used to control the added hydraulic valves, as well as relays for switching between the electronic control lines. The control lines and the originals, sensors distributed throughout the manipulator and on the legs to determine the machine state, a PLC to control and collect data from all of the aforementioned components, computers and networking devices for the actual evaluation of the inputs and determination of the actuator responses, and safety equipment to allow the operator to assume control at any moment were also included.

A PLC and two port multiplier boxes were integrated into the excavator to facilitate the interfacing of all electronic outputs and inputs with the existing computing infrastructure. The additional PLC is an INTER CONTROL digsy fusion S-P, situated within the cabin. The aforementioned PLC is utilized for interfacing with the assorted CAN buses, select electric control lines, the upper carriage (comprising the cabin and manipulator) valves, and the computers. The port multipliers are a pair of INTER CONTROL ICN-V devices that are connected to the lower carriage valves and certain sensors. Furthermore, an analog output box was incorporated

to facilitate the output of several analog control voltages, and two dual valve controllers were included to regulate additional functions.

Hydraulics control

The hydraulic valves are controlled either directly by the added PLC or indirectly through the port multipliers. The RotoTilt is controlled via a CAN bus using its automatic grading control interface, which allows control of both tilt and rotation. In addition, the RotoTilt software has been expanded to control the RotoTilt controlled tool hydraulics.

When using the computers' built-in PLCs to control the steering, the added PLC uses the analog output box to generate electronic signals similar to the inputs it would otherwise receive from a hydraulic sensor that senses pressure in the control lines from the joysticks. To control the wheel drive, the same analog output box is used to recreate the signal from the drive pedal. The last output signal from the analog output box is used to replicate the signal from the engine speed potentiometer to control the engine speed.

The dual valve controllers are used to control a hydraulic flow valve that limits flow to the actuator in the lower carriage, to switch between steering control by the built-in PLC or direct per-wheel steering by the additional PLCs, and to switch between claw rotation and telescoping control.

Sensors

The steering position is already covered by built-in sensors, and their positions are transmitted over the built-in PLC's CAN bus. The added PLC simply listens to these messages and evaluates and converts them as needed. The RotoTilt system provides the tilt position of the rotor when automatic grade control is engaged. All of the sensors added to the mechanics are connected to one of two CAN buses, one for the upper carriage and one for the lower carriage. The added PLC manages these two CAN buses and evaluates and converts all these sensor measurements.

Sensors for environmental perception and monitoring have also been added. These are used for autonomous and remote control (see Table 16.2). There are three Lidar sensors, a Velodyne sensor at the top center of the cabin roof, and an Ouster sensor on the left and right corners of the cabin roof, tilted about 45 degrees to get a better to provide a better field of view. Two Prosilica GT RGB cameras and three Seek thermal cameras are mounted at the front of the cabin to cover the excavator's the excavator's working area. For accurate positioning, an inertial navigation system with GPS support has been attached as well. Five network-connected cameras

Table 16.2 Sensors for high-level operations.

Type	Model	Purpose
Lidar	Velodyne HDL32E	Self-localization and mapping, octomap for manipulation
Lidar	Ouster OS0-128	Octomap for manipulation
RGB Camera	Prosilica GT1380C	Image processing, colored point clouds
Time-of-Flight Camera	LUCID Vision Helios2	Object detection and pose estimation, tool mounted
RGB Camera	LUCID Vision TRI032S-CC	Image processing, tool mounted
Thermal Camera	Seek S304SP	Thermal mapping
Fish-eye camera	AXIS M3058-PLVE Network Camera	Video feed for remote control
Pan-Tilt-Zoom camera	AXIS Q6135-LE PTZ Network Camera	Video feed for remote control with pan, tilt and zoom functions
Inertial navigation system	Advanced Navigation Ekinox D	Localization based on RTK GPS and inertial data

have been added for remote control and environmental monitoring: A fish-eye panoramic camera on each side of the camera and a pan/tilt/zoom camera on the camera in the upper left front corner of the cabin. All of the above are connected to the computer hardware via Gigabit Ethernet.

Safety

An all-electric system ensures that autonomous control cannot be inadvertently activated and can be deactivated at any time during field testing, returning control to the safety operator seated in the cabin. This is achieved by a set of relays and switch valves that maintain the machine's normal electrical and hydraulic connections when not energized, and only connect the autonomous control components to the machine's systems when energized. This results in a safe system where the autonomous control components are not powered when autonomy is disabled. To enable the autonomy mode, six components must all vote for enable: two radio autonomy stops, a dead man switch, an autonomy stop button inside the cabin, a key switch, and the added PLC.

Computing electronics

The computing infrastructure consists of two computers and a PLC. These are connected in various ways, some directly and some through a managed

level 3 switch. The PLC is only connected to the “Control” computer and has no access to any other network, in order to protect its software from unauthorized access. The Prosilica cameras and Ouster LIDAR sensors are connected directly to the “Perception” computer to provide a maximum bandwidth/ minimum latency path to the data processing software.

The Velodyne LIDAR, the inertial navigation system, the “Control” and “Perception” computers form one network segment, the AXIS and thermal cameras form another network segment with the “Control” computer to keep the video surveillance separate from the autonomous control data streams. The “Control” computer routes between these two networks and provides the gateway, firewall and network address translation to the outside networks (usually the Internet). Access to the web servers on the AXIS, thermal cameras and the video web servers on the “Perception” machine is done with port forwarding on the “Control” computer, making them accessible to the outside network.

16.2.3 Software design

The primary control software runs on Ubuntu 18.04 Linux with Robot Operating System (ROS) as the robotics framework. The PLC uses the CODESYS software suite for programming. The software structure reflects the control structures described in Subsections 16.3.2, 16.3.3, and 16.3.4.

16.3 Modeling and control

The kinematics of the parallel structures are modeled for state estimation and control purposes in the next subsection. The structure for different levels of control are detailed in the subsequent subsections.

16.3.1 Modeling

The movements of the excavator joints are initiated by the respective hydraulic actuators. Actuator and joint data (including position, velocity, force or torque) sometimes differ due to the mechanical connections between them. For higher level control and simulation, the joint data is more relevant, while the actuator data is important for the actual execution of the desired motion. Kinematic analysis of the limbs allows modeling the relationship between joint and actuator positions, velocities, and forces. The established equations can later be used to derive the forces and torques of the robot end effectors from the values of the pressure sensors in the actuators.

Due to mechanical constraints and practical reasons, the joint encoders are mounted at different points of the kinematic chain of each actuator-joint mechanism, requiring an individual analysis of each limb. For some links, the mechanisms involved are well known and the analysis can be derived by reusing previously established equations. Others consist of a combination of known mechanisms or are completely unique. The following examples of the shovel and the dipper joints fall into these categories and show the procedure for deriving the necessary kinematics.

Shovel joint kinematics

Fig. 16.2a shows the shovel mechanism, its joint angle θ and its actuator length s . The mechanism can be separated into two subsystems sharing the common line segment b : The connection between points A, B, and E can be interpreted as an Inverted Slider Crank mechanism (ISC), whereas the subsystem of points B, C, D, and E acts as a four-bar mechanism (FB).

Using [7], the output angle ε of the ISC can be written as a function of the actuator length s as

$$\varepsilon(s) = \arccos\left(\frac{a^2 + b^2 - s^2}{2ab}\right). \quad (16.1)$$

Accordingly, the rotational speed $\dot{\varepsilon}_{isc}$ can be derived and results in a function of the actuator velocity \dot{s} and position s :

$$\dot{\varepsilon}(\dot{s}, s) = \frac{s}{ab \sin(\varepsilon)} \dot{s}. \quad (16.2)$$

The inverses of the equations above yield the actuator position s or its velocity \dot{s} as a function of the output angle ε or additionally its rotational speed $\dot{\varepsilon}$, still acting in the scope of the ISC:

$$s(\varepsilon) = \sqrt{a^2 + b^2 - 2ab \cos(\varepsilon)}, \quad (16.3)$$

$$\dot{s}(\dot{\varepsilon}, \varepsilon, s) = \frac{ab \sin(\varepsilon)}{s} \dot{\varepsilon}. \quad (16.4)$$

Similarly, the required positional and velocity dependencies for the FB are determined based on [8]. The relationship between angle φ and ϱ as well as their velocities are written as:

$$\varrho(\varphi) = \arccos\left(\frac{d^2 + e^2 - c^2}{2de}\right) + \arccos\left(\frac{f^2 + e^2 - b^2}{2fe}\right), \quad (16.5)$$

$$\dot{\varrho}(\varrho, \varphi) = \frac{b \sin(\alpha)}{d \sin(\varphi + \alpha - \varrho)} \dot{\varphi}, \quad (16.6)$$

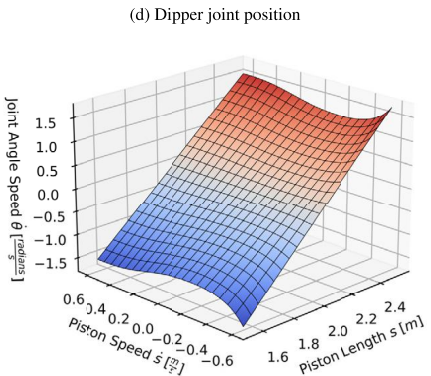
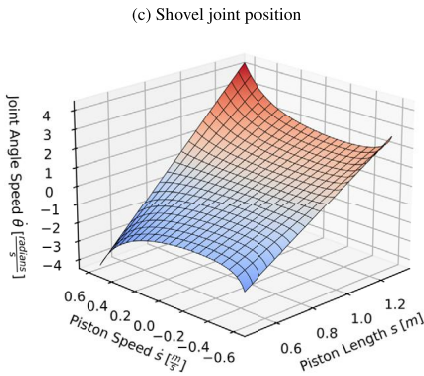
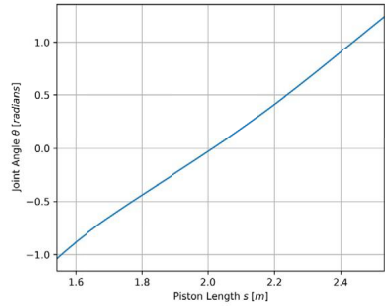
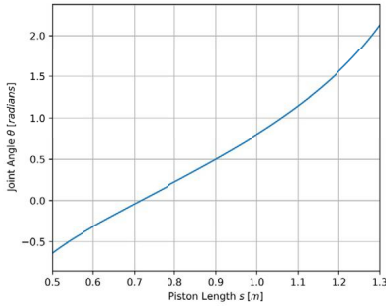
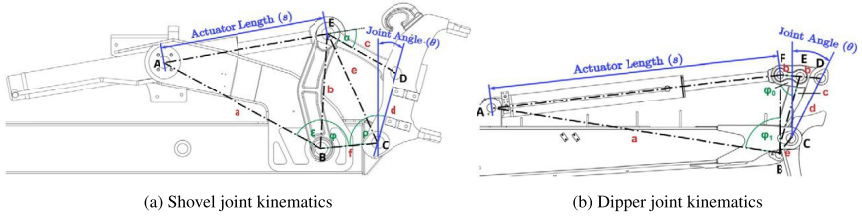


Figure 16.2 Kinematics of the shovel and dipper joints along with solutions for the parallel kinematics.

where

$$\alpha = 2\pi - \text{atan} \left(\frac{d \sin(\varrho) - b \sin(\varphi)}{f + d \cos(\varrho) - b \cos(\varphi)} \right) - \varphi \quad (16.7)$$

defines the *coupler angle*.

Combining the equations of FB and ISC by formulating a rule of the common line segment b as

$$\varphi - \varphi_{off} = -(\varepsilon - \varepsilon_{off}) \quad (16.8)$$

(with φ_{off} and ε_{off} being constant angular offsets of the mechanism) leads to the forward and inverse parallel kinematics of this shovel mechanism. For clarification, the index *isc* is appended to the ISC functions of actuator position $s(\varepsilon)$ and velocity $\dot{s}(\dot{\varepsilon}, \varepsilon, s)$ as described by (16.3) and (16.4).

$$s(\theta) = s_{isc}(-\varphi(\theta)), \quad (16.9)$$

$$\dot{s}(\dot{\theta}, \theta) = \dot{s}_{isc}(-\dot{\varphi}(\dot{\theta}, \theta), -\varphi(\theta), s_{isc}(-\varphi(\theta))), \quad (16.10)$$

$$\theta(s) = \varrho(-\varepsilon(s)), \quad (16.11)$$

$$\dot{\theta}(\dot{s}, s) = \dot{\varrho}(\dot{\varepsilon}(\dot{s}, s, \varepsilon(s)), -\varepsilon(s)). \quad (16.12)$$

The functions can be visualized as seen in Fig. 16.2c and Fig. 16.2e.

Dipper joint kinematics

Fig. 16.2b shows the dipper mechanism, its joint angle θ and its actuator length s . The mechanism cannot be attributed to any generalized mechanism. The function $s(\theta)$, which can be interpreted as part of the inverse kinematics, was determined analytically:

$$s(\theta) = \sqrt{a^2 + b^2 + c^2 - 2bc \cos(\varphi_0(\theta))} - Z, \quad (16.13)$$

where $Z = 2a\sqrt{b^2 + c^2 - 2bc \cos(\varphi_0(\theta))} \cos(\varphi_1(\theta))$ and a through c are constant lengths and $\varphi_0(\theta)$, $\varphi_1(\theta)$ being auxiliary angles dependent on θ . These angles can be derived using geometric relations (e.g., law of cosines). Fig. 16.2d shows the inverse kinematic function $s(\theta)$.

Since the forward kinematics could not be determined this way, function $s(\theta)$ was used to find an approximation: After generating 22781 values in 0.0001° steps of the input variable θ in the joints actual range of motion, its corresponding piston lengths s were calculated. Using the method of least squares, a polynomial of 15th degree was found to be a good estimate for the inverse $\theta(s)$ (for which the residual sum of squares was negligible):

$$\begin{aligned} \theta(s) \approx & -0.0215s^{15} + 0.256s^{14} - 0.998s^{13} + 0.530s^{12} \\ & + 4.718s^{11} - 2.211s^{10} - 27.221s^9 - 3.952s^8 \\ & + 156.505s^7 + 98.182s^6 - 922.31s^5 - 360.465s^4 \\ & + 6061.165s^3 - 10994.638s^2 + 8539.325s - 2565.228 \end{aligned}$$

Its first derivative with respect to time yields function $\dot{\theta}(\dot{s}, s)$, which is depicted in Fig. 16.2f.

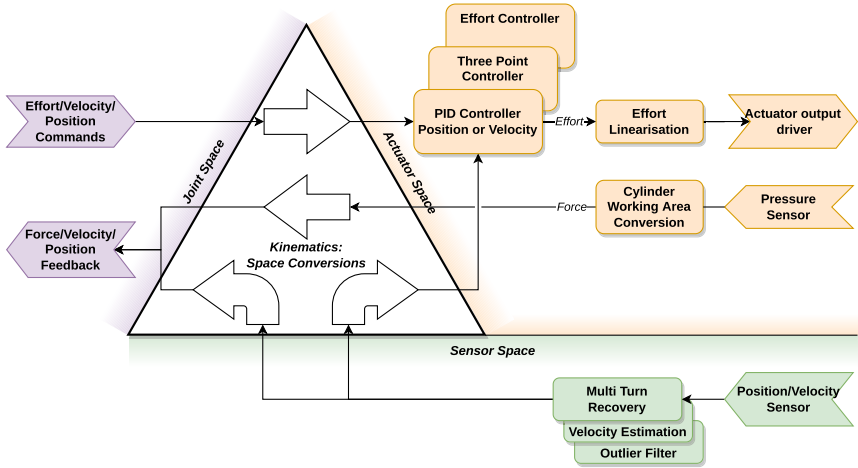


Figure 16.3 Low-level control structure.

16.3.2 Low-level control

The low-level control runs on the General Computing Module of the PLC added to the excavator. Fig. 16.3 shows the general structure of the low-level Control.

The primary function of the low-level control is to translate commands from the higher levels of control (effort, velocity, or position) into direct commands for the actuators. This is achieved through the utilization of sensor data, which serves as the basis for achieving the desired outcomes.

In order to utilize the relatively straightforward PID and three-point controllers, it is necessary to convert the input values from sensors and commands, as well as the output values to the actuators, to a common control space. Actuator space was selected as the common space utilized by the controllers. Additionally, the low-level control is responsible for the kinematic conversion between the control spaces of the actuators, sensors, and the actual joint. The higher levels of control operate exclusively within the respective joint spaces.

In addition to converting data from sensor to actuator or joint space, the sensors undergo minor preprocessing. In the case of continuous rotational sensors that do not register a full turn, the turn count is reconstructed. Outlier positions are identified and removed by imposing a maximum distance between positions and a maximum number of samples. In the event that velocity data is absent or of insufficient quality, it is estimated by first

Table 16.3 Joint controllers. Original controllers by built-in PLC are marked as “o”, retrofitted by added PLC as “x”, and not available as “-”.

Joints	Position PID	Velocity PID	Position Three-Point	Effort	Current
Cabin, Boom, Dipper, Telescope, Shovel	x	x	-	x	x
Tilt, Roto	x	x	-	o	o
Stabilizers	x	x	-	x	x
Swivel	-	-	x	x ^a	-
Steering	xo ^b	xo ^b	-	xo ^b	xo ^b
Wheel	-	x	-	o	-
Claw rotate	-	-	x	x ^a	-
Claw telescope	-	-	-	x ^a	-

^a Direction only.^b Selectable.

performing a discrete differentiation and then applying a low-pass filter to reconstruct a limited degree of additional resolution. The pressure sensors are connected to the actuators, and thus their pressures are in actuator space. After considering the hydraulic cylinder areas, a force is calculated, converted to joint space, and reported back.

Three controllers are implemented for position and velocity control, Effort (force) control is open-loop and passed directly to the actuators (after conversion to actuator space). Table 16.3 shows the different types of controllers used by each joint, as dictated by its sensor and actuator structure.

The actuators generally have a non-linear relationship between their input signal (e.g., current) and their output action (e.g., velocity). A simple interpolated look-up table is used to convert effort commands from the controllers. This look-up table can be as simple as a linear one-to-one mapping, add inversion of the control, dead zones, or a more complex shape.

16.3.3 Mid-level control

The mid-level control layer provided controllers for use with remote control, safety controllers and monitors, and interfacing between high-level and low-level controllers. The general structure is as shown in Fig. 16.4.

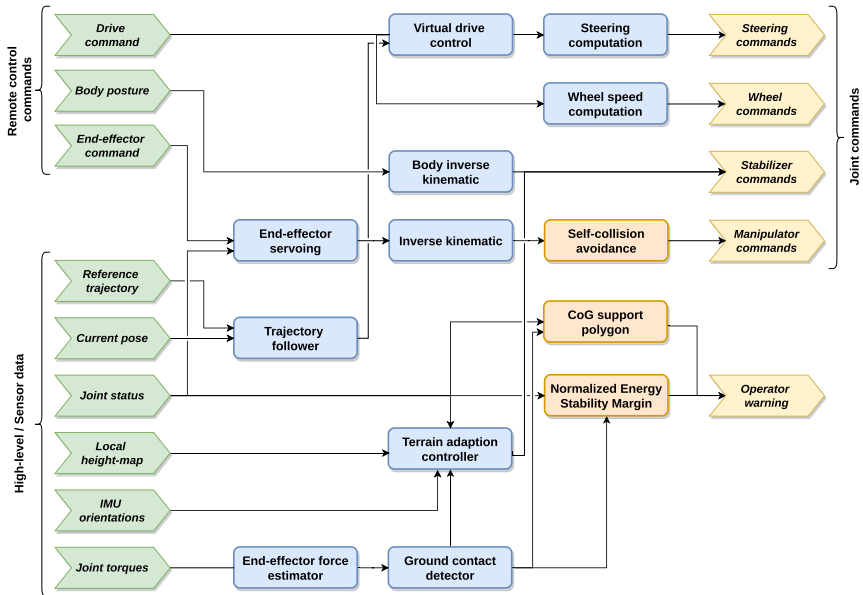


Figure 16.4 Mid-level control structure.

Remote control

In order to facilitate remote control operation, the commands pertaining to the drive, body posture, and end-effector are processed and transmitted to the low-level joint controllers. The drive command is a combination of the steering command and the velocity commands for the wheels. The command is then conveyed to the virtual drive control module, which translates the drive commands into a format that can be mapped to a bicycle kinematic model. The steering computation module performs the conversion of the reference steering angle to the desired joint angles of the robot steering joints. The wheel speed computation module is responsible for computing the wheel speeds based on the virtual drive speed.

The body posture command provides the desired velocities for the robot's body posture, including height, roll, pitch, and so forth. The body inverse kinematic module is responsible for transforming the command into the corresponding stabilizer joint velocities. This facilitates the operator's ability to control the leg heights.

The end-effector twist command for the manipulator will be routed through the servoing module ([9]), which computes the desired end-effector position. This position is then utilized by the inverse kinematic

module to compute the joint angles. These joint angles, provided they do not result in a self-collision, are transmitted to the low-level controllers.

Autonomy controllers

A set of controllers has been developed to facilitate the functioning of different high-level tasks. The trajectory follower, which takes in the reference trajectory and the current pose of the robot as inputs, issues commands to the virtual drive to steer the robot base along a desired path. Two controllers have been implemented: one based on model predictive control (MPC) ([10]) and a second based on pure pursuit ([11]).

The terrain adaptation controller is designed to autonomously adjust the active suspension system to accommodate uneven terrain by controlling the elevation of the legs. The controller's multiple objectives include maintaining stability, avoiding ground collisions, and maintaining ground contact of wheels. As outlined in [12], the basic version of the controller is capable of maintaining ground contact and the attitude of the robot body. A more sophisticated controller employing reinforcement learning was developed in [13] to automatically adapt to the terrain based on height maps, contact estimation, and orientation information. The ground contact detection employs an end-effector forces estimation, which is itself based on the measured joint torques. It should be noted that the end-effector forces are only partially observable, as not all joint torques are measured.

In the context of controlling a robotic platform with adaptive suspension, a primary factor that warrants consideration is that of the tip-over stability. In the field of robotics, a number of stability margins have been developed to address this issue. In their study, Garcia et al. [14] compared several stability margins and concluded that the Normalized Energy Stability Margin (NESM), developed by Hirose [15], is particularly well suited for robots navigating uneven terrain. The NESM is employed for the purpose of learning the terrain adaptation controller, as well as for the purpose of providing a warning to the operator.

16.3.4 High-level control

The high-level control module serves as the interface between the operator or mission control and the mid-level or low-level controllers. Fig. 16.5 provides an overview of the high-level control for ARTER.

The high-level component receives input from the mission control system and sensor data. The footprint changer module receives the desired

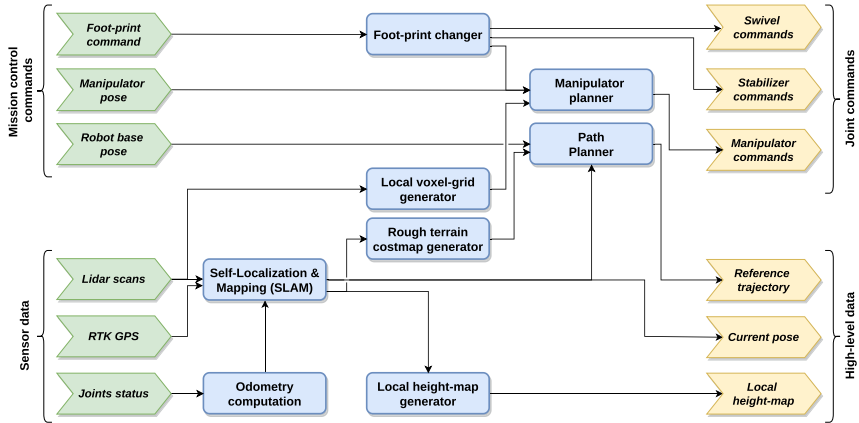


Figure 16.5 High-level control structure.

footprint as input and controls the manipulator and the swivel and stabilizer joints in the legs. The footprint changer module executes a set of predefined sequences to utilize the manipulator in the front side to lift the front wheels and then move the swivel joints to the desired angles. This process is repeated for the rear side as well.

The manipulator planner, as described in reference [9], accepts the desired manipulator pose as a command input and also utilizes the local OctoMap ([16]), which is a probabilistic 3D occupancy grid map. The OctoMap is generated through the integration of data obtained from three distinct LIDARs. The planned motion is then executed by transmitting the requisite commands to the manipulator.

The path planner assists in the planning of the trajectory of the robot base, taking a 2D pose (x , y , and yaw) on the map as input and generating the necessary commands to navigate to a different location. At present, the three-dimensional environment is represented as a grid map ([17]) and converted to the corresponding two-dimensional cost map by taking into account terrain features such as obstacles, roughness, slope, and so forth. Subsequently, the path is planned using the Smac planner by Macenski [18], which employs the Hybrid-A* algorithm. The output of the planner is the trajectory, which is then conveyed to the mid-level controller.

The self-localization and mapping (SLAM) module is responsible for generating a 3D map of the environment and estimating the pose of the robot within that map. This is implemented using the SLAM3D package ([19]). This module is principally based on LIDAR scans and the data

provided by the inertial navigation system which utilizes the real-time kinematic (RTK) GPS as an input. Additionally, the computed odometry of the vehicle, which is based on the joint data, is provided as input. The localization and map of the environment generated by SLAM are employed to create a local height map, which is then utilized by the mid-level controllers.

16.4 Conclusion and outlook

The ARTER system has been developed for use with excavators in environments that are unsuitable for human presence and which are characterized by challenging terrain. The retrofitted walking excavator with a wheel-on-leg design, similar to that of many outdoor mobile robots, exhibits a high degree of traversability. The use of a manipulator for locomotion serves to enhance its capabilities further.

The retrofitting of the original vehicle necessitated extensive modifications and additions to the hydraulic control system, sensors, electrical and electronic components, and software. Furthermore, the kinematics, including parallel kinematics, were modeled and implemented to develop controllers for the joints. A complex control structure is developed for executing remote control and autonomous operations.

16.4.1 Success stories

In the context of the ROBDEKON project, two distinct scenarios were delineated, each of which makes use of the robot's unique functional capabilities. The initial scenario entails the collection of waste barrels that have been discarded in a waste disposal site. The barrels are in a state of corrosion and damage, with the presence of chemicals that are harmful to humans. This underscores the necessity for the deployment of remotely operated robots. In this context, the attainment of a higher degree of autonomy is challenging due to the absence of structured elements and the difficulty in automatically detecting the barrels. This scenario was successfully demonstrated as part of the project.

The second scenario entails the collection of soil samples from predefined locations via the utilization of a custom-designed gripper and a lance that has been specifically conceptualized for soil sampling. In this scenario, a high degree of autonomy is employed. Given an approximation of the location of the lance, the robot is capable of autonomously planning the optimal route to reach that location. The lance magazine and each lance are

equipped with AprilTag markers ([20]) for the purpose of detecting their respective poses. The robot grasps a lance, removes it from the magazine, and autonomously proceeds to the target sampling location. The sample is obtained by inserting the lance into the ground. The retrieved sample is then moved and placed back in the magazine. These steps are repeated until the desired number of samples has been obtained. Once the samples have been analyzed for their chemical contents, a map of the environment is generated with respect to the soil contents.

16.4.2 Design limitations

Most of the joint controllers in the manipulator are controlled using hydraulic pilot control valves, which in turn controls the main valves. This setup introduces a lot of non-linearities and other undesired effects which cannot be completely removed by the current controllers. This results in inaccurate trajectory tracking of joints.

As far as control of joints are concerned, one of the main limitation is the lack of force control, as described by Hutter in [21]. The main challenge is the design of a custom hydraulic valve, which is complex and expensive. If such valves were available, the chassis can be force controlled, as described by Hutter in [22], or the manipulator can be controlled in impedance control mode.

The accuracy of the pose computation of the manipulator end-effector and wheels are limited by the sensor resolution and flexibility in the mechanical system. Even though the sensors are fairly accurate, due to the large dimensions of the links, a small inaccuracy in the joints will result in a few centimeters of inaccuracies in the end-effector. Additionally, there is flexibility of the mechanical components, which make the final pose computation of the end-effector off by a few centimeters.

16.4.3 Lessons learned

Operating heavy equipment in a vehicle is very different from operating it remotely. Many factors that are intuitive and straightforward become difficult during remote operation. To improve this experience, in addition to camera images, robot state estimation, contact force estimation, tilt stability calculations, and external 3D information such as local maps are required. Assistive features such as automatic ground adaptation can also reduce operator effort.

16.4.4 Future work

The ARTER system is actively developed in several projects for different scenarios. The controller designs discussed here may undergo major changes in the future.

The locomotion control of the robot is complex and requires advanced solutions like reinforcement learning to solve the problem. The terrain adaptation controller has already been developed using deep reinforcement learning. Controllers for other locomotion modes such as stepping and climbing are currently under development. A hierarchical reinforcement learning based controller is also being developed to select the appropriate controller based on the sensor data.

In this work, the kinematics are explicitly modeled, the controller linearized, and tuned by hand to make the end-effector move in the desired way. Research is being conducted to control the end-effector without these steps and to learn the controller in an end-to-end fashion. The input is the pose of the end effector and the control is the raw valve inputs.

References

- [1] F. Cordes, F. Kirchner, A. Babu, Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain, *Journal of Field Robotics* 35 (7) (2018) 1149–1181, <https://doi.org/10.1002/rob.21808>, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21808>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21808>.
- [2] B.H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, B. Cooper, Athlete: a cargo handling and manipulation robot for the moon, *Journal of Field Robotics* 24 (5) (2007) 421–434, <https://doi.org/10.1002/rob.20193>, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20193>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20193>.
- [3] W. Reid, F.J. Pérez-Grau, A.H. Göktoğan, S. Sukkarieh, Actively articulated suspension for a wheel-on-leg rover operating on a martian analog surface, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 5596–5602, <https://doi.org/10.1109/ICRA.2016.7487777>.
- [4] M. Bjelonic, C.D. Bellicoso, Y. de Viragh, D. Sako, F.D. Tresoldi, F. Jenelten, M. Hutter, Keep rollin’—whole-body motion control and planning for wheeled quadrupedal robots, *IEEE Robotics and Automation Letters* 4 (2) (2019) 2116–2123, <https://doi.org/10.1109/lra.2019.2899750>.
- [5] D. Jud, S. Kerscher, M. Wermelinger, E. Jelavic, P. Egli, P. Leemann, G. Hotz, M. Hutter, Heap - the autonomous walking excavator, *Automation in Construction* 129 (2021) 103783, <https://doi.org/10.1016/j.autcon.2021.103783>, <https://www.sciencedirect.com/science/article/pii/S092658052100234X>.
- [6] J. Petereit, J. Beyerer, T. Asfour, S. Gentes, B. Hein, U.D. Hanebeck, F. Kirchner, R. Dillmann, H.H. Götting, M. Weiser, M. Gustmann, T. Egloffstein, ROBDEKON: robotic systems for decontamination in hazardous environments, in: 2019 IEEE In-

- ternational Symposium on Safety, Security, and Rescue Robotics (SSRR), 2019, pp. 249–255, <https://doi.org/10.1109/SSRR.2019.8848969>.
- [7] R.L. Williams II, *Mechanism Kinematics & Dynamics: Mechanical Engineering*, Ohio University, 2019, <https://www.ohio.edu/mechanical-faculty/williams/html/PDF/Supplement3011.pdf>.
 - [8] J.M. McCarthy, G.S. Soh, *Geometric Design of Linkages*, 2nd edition, *Interdisciplinary Applied Mathematics*, vol. 11, Springer, New York, NY, 2011.
 - [9] D. Coleman, I. Sucan, S. Chitta, N. Correll, Reducing the barrier to entry of complex robotic software: a MoveIt! case study, April 2014.
 - [10] A. Babu, K. Yurtdas, C. Koch, M. Yüksel, Trajectory following using nonlinear model predictive control and 3D point-cloud-based localization for autonomous driving, <https://doi.org/10.1109/ECMR.2019.8870956>, 2019, pp. 1–6.
 - [11] E. Jelavic, D. Jud, P. Egli, M. Hutter, Towards autonomous robotic precision harvesting, arXiv preprint, arXiv:2104.10110, 2021.
 - [12] F. Cordes, A. Babu, F. Kirchner, Static force distribution and orientation control for a rover with an actively articulated suspension system, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017)*, 2017.
 - [13] A. Babu, F. Kirchner, Terrain adaption controller for a walking excavator robot using deep reinforcement learning, in: *2021 20th International Conference on Advanced Robotics (ICAR)*, 2021, pp. 64–70, <https://doi.org/10.1109/ICAR53236.2021.9659399>.
 - [14] E. Garcia, J. Estremera, P.G. De Santos, A classification of stability margins for walking robots, *Robotica* 20 (6) (2002) 595–606.
 - [15] S. Hirose, H. Tsukagoshi, K. Yoneda, Normalized energy stability margin: generalized stability criterion for walking vehicles, in: *Proc. Int. Conf. Climbing and Walking Robots*, 1998, pp. 71–76.
 - [16] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: an efficient probabilistic 3D mapping framework based on octrees, *Autonomous Robots Software* available at <https://octomap.github.io>, 2013, <https://doi.org/10.1007/s10514-012-9321-0>.
 - [17] P. Fankhauser, M. Hutter, A universal grid map library: implementation and use case for rough terrain navigation, in: A. Koubaa (Ed.), *Robot Operating System (ROS) – The Complete Reference*, vol. 1, Springer, 2016, Ch. 5, https://doi.org/10.1007/978-3-319-26054-9_5, <http://www.springer.com/de/book/9783319260525>.
 - [18] S. Macenski, F. Martin, R. White, J. Ginés Clavero, The marathon 2: a navigation system, in: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
 - [19] <https://github.com/dfki-ric/slam3d>.
 - [20] J. Wang, E. Olson, AprilTag 2: efficient and robust fiducial detection, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
 - [21] M. Hutter, P. Leemann, S. Stevsic, A. Michel, D. Jud, M. Hoepflinger, R. Siegwart, R. Figi, C. Caduff, M. Loher, S. Tagmann, Towards optimal force distribution for walking excavators, in: *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 295–301, <https://doi.org/10.1109/ICAR.2015.7251471>.
 - [22] M. Hutter, P. Leemann, G. Hottiger, R. Figi, S. Tagmann, G. Rey, G. Small, Force control for active chassis balancing, *IEEE/ASME Transactions on Mechatronics* 22 (2) (2017) 613–622, <https://doi.org/10.1109/TMECH.2016.2612722>.

This page intentionally left blank

PART 5

Software and outlook

This page intentionally left blank

CHAPTER 17

PHOBOS: creation and maintenance of complex robot models

Henning Wiedemann^a, Kai A. von Szadkowski^b, and Malte Langosz^b

^aWorking Group Robotics, University of Bremen, Bremen, Germany

^bRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

For most robotic applications, model representations of the respective systems are crucial for controlled operation. No matter whether for simulations or different model-based-control approaches, a lot of software tools require detailed information about the robot. As CAD (Computer-Aided Design) software is mostly dedicated to create mechanical or kinematic designs respectively, the resulting models lack many of the information required for such tasks. PHOBOS¹ is a tool to create, edit and annotate robot models with any additional information needed.

17.1 Model information

In the following, various applications for robot models will be discussed, with regard to the necessary information contained and the possible representations that facilitate the numerous calculations derived from these models.

17.1.1 Kinematic properties

The most common kinematic model formats describe a kinematic tree as a hierarchy of links and joints (e.g., URDF, SDF, see Section 17.2). Basic serial mechanisms can be described easily that way.

As soon as kinematics become more complex, e.g., for parallel mechanisms like a four-bar mechanism, where the movement of a joint is determined by the movement of other joints, some challenges emerge. Closing kinematic loops using a simple link-and-joint approach, defining

¹ The state reported here relates to PHOBOS version 2.0.0.

the dependent joints' mechanical constraints implicitly, would be a valid approach. However, depending on later usage of the model, calculating implicitly determined motion constraints may require higher computational effort. Hence, a more pragmatic approach is to explicitly provide further information: In the simplest case, this can be solved, e.g., by defining one joint as mimicking another joint's motion. For even more complex mechanisms, where one joint is not only dependent on one but multiple others, a precise description of the specific submechanism is required (see submechanism file description in HYRODYN [1,2]).

17.1.2 Geometric information

Another application where any basic representation will result in higher computational effort compared to more elaborate annotations is collision detection and avoidance. In formats like URDF or SDF, link geometry is described either as primitive shapes, such as boxes or spheres, or as meshes. While computing collisions of primitives is analytically simple and thus fast, computational effort for mesh collisions increases with mesh size and complexity. This computation time also grows exponentially as a function of the degrees of freedom. Exported from a CAD software, mesh geometry representations of designed parts are typically not optimized regarding the ratio of vertex numbers to level of detail. Consequently, reworking and enhancing the collision representation is essential. Even though we need precise geometry information for collision detection, calculating collisions in real-time is mandatory. Thus, in many cases it is advantageous or necessary to strike a balance between accuracy of representation and performance, yet typically less detail is required as for visual representations. Or in a nutshell: A simpler mesh with fewer vertices can adequately represent the geometry of a link. For collision prevention, it is not a problem for such a hull to be slightly larger than the mechanical parts, as long as the level of abstraction is chosen reasonably, considering the robot's application. Further computational simplification is possibly by annotating which links are inherently unable to collide. Regarding the visual geometry representation per mesh, a reduction of not visible vertices can also benefit performance for visual rendering.

17.1.3 Closing the gap between design and hardware

Rarely is a CAD model a 100% accurate representation of the final real robot; especially regarding the mass and inertial properties. Even if materials are specified correctly for all parts, the finished real robot will diverge

slightly in its physical properties, and it is difficult to include all minute details such as, e.g., cables. Hence, any exported CAD model is good for obtaining a basic kinematic representation with a first iteration of mass and inertia properties, but any such model needs to be enhanced to accurately represent the assembled system.

17.1.4 Annotating complex components

Every robot relies on actuators and sensors. Therefore, the model has to be able to provide information on motor parameters, as well as sensor types, placements and properties. Some robots might also include other elements such as mechanical interfaces that need to be represented in some form in a model.

17.1.5 Additional information

Apart from purely hardware-related annotations, often information such as simulation properties for specific simulators might be required. Often physical processes such as contact forces require specific parameters of a simulator to be set to accurately approximate the behavior of the real physical objects. Furthermore, especially when testing vision-based tasks in simulation, it is helpful to provide information on visual materials, shading, or lighting specific to the utilized rendering engine.

17.1.6 Modular approach for complex robots

Complex robots with multiple serial and parallel mechanisms often contain a large amount of links, joints and thus, the corresponding annotations. For such systems, it makes sense to take advantage of recurring structures (e.g., arms and legs of the humanoid depicted in Fig. 17.3). Consequently, assembling such robot models from smaller parts by loading them multiple times and if necessary mirror symmetric parts, helps to decrease redundant data and workload immensely. This applies especially if annotations do not have to be specified for each model version, but are inherited.

17.2 Model formats

There are multiple formats for modeling robots, each with their own specialization and particularities designed for a particular purpose, such as URDF [3], SDF [4], or SRDF [5], with URDF being the most common in robotics research due to its use in ROS. The authors of [6] review

common formats and discuss details of their specific advantages and limitations. Here, we focus on the format SMURF, of which a previous version has been reviewed by [6], too.

As the crucial part of PHOBOS is to support delivering compatible models for as many software tools as possible, the flexible SMURF format with URDF and SDF respectively as underlying kinematics representation suits this task well.

17.2.1 URDF & SDF

URDF, the Universal Robot Description Format,² is the principal robot description for ROS. It is an XML format that at its core models a kinematic tree as a hierarchy of *links* and *joints*. Links define coordinate frames in space and contain data on physics (mass, inertia tensor) and geometry of the mechanical parts they contain, with geometric information divided into visual and collision representations. A peculiarity of URDF is that links themselves contain no transformation data on where in space they are located with reference to one another. Instead, this data is contained in its joint elements: Each joint defines the transformation of its parent to its child link, as well as the constraints governing the motion between these links. Additional information on robotic hardware such as sensors is rather simplistic in URDF, even with current proposals for extensions³ mostly guided by data- or simulation-focused parameter sets, lacking, e.g., lens properties for cameras. Furthermore, URDF is not modular and does not allow for importing URDFs to define components. However, in the ROS community, the XML macro language (XACRO) serves a somewhat similar function (cf. [7]). Another extension of URDF is SRDF,⁴ the Semantic Robot Description Format, which allows to group subsets of the kinematic tree defined in a URDF file into semantic units, e.g., individual arms of a robot.

The Simulation Description Format (SDF), is another popular XML format for robot descriptions that was created for the Gazebo simulator.⁵ While there is some similarity to URDF when it comes to kinematic definitions, SDF is much more extensive, covering elements beyond a single robotic system, such as parameters for environment physics and lighting,

² <http://wiki.ros.org/urdf/XML/model>.

³ <http://wiki.ros.org/urdf/XML/sensor/proposals>.

⁴ <http://wiki.ros.org/srdf>.

⁵ <https://gazebosim.org/>.

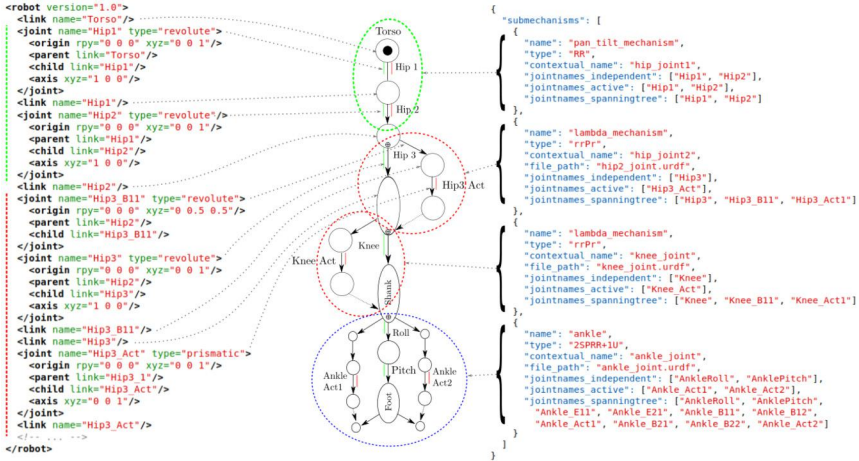


Figure 17.2 Example of a sparse URDF representation (left) and the submechanisms section in the SMURF files (right) of a serial-parallel kinematic tree. The loop-closing joints are displayed as dashes, as they are not part of the URDF, because plain URDF does not support loop closures.

This way, SMURF is able to hold all information available about a system. By using the converter delivered by PHOBOS model files in various formats⁶ can be generated and will hold all data their format supports. Making use of this, PHOBOS takes care that all information compatible with the underlying XML-format model file is placed there, too. This makes SMURF very versatile as it provides as much information as possible to software that can only parse the already present XML-file. Some of the software proposed in this book, like HYRODYN or the simulator MARS, are able to directly parse and make use of the complete SMURF models. (See Fig. 17.2.)

17.2.3 Scenes & assemblies

SMURF is complemented by two additional formats: SMURF-Assemblies (SMURFA) and SMURF-Scenes (SMURFS), which are similar but used for different purposes. Both are able to arrange multiple entities in space, with SMURFA describing the composition of a modular robotic system and SMURFS defining how multiple robots or objects are arranged in

⁶ Currently, only the conversions between SMURF, URDF, SDF are supported. The support of further formats is goal for future work. Contributions are appreciated.

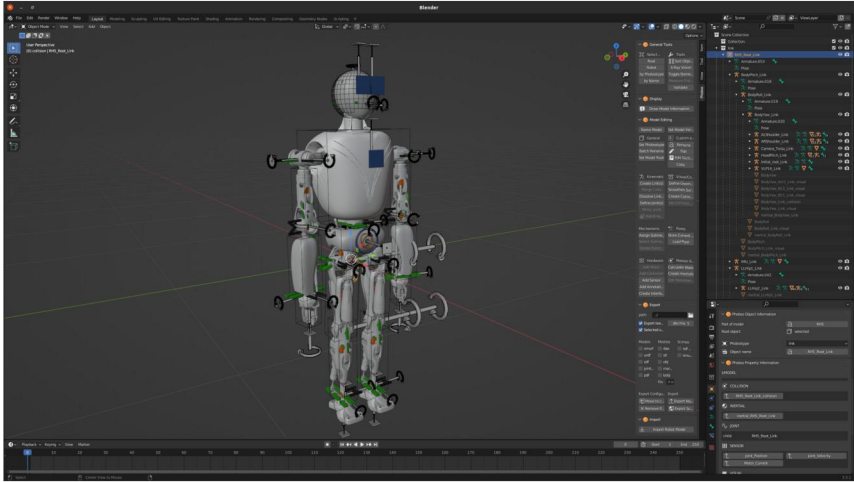


Figure 17.3 Screenshot of the Blender add-on of PHOBOS. Displaying the robot in the “3D View” in the center with its frames, joints, collision shape outlines, sensors, inertials, and motors. On the upper right side the “Outliner” panel displaying the kinematic tree structure and below the custom properties editor for model annotations.

a scene, along with the simulation parameters for the world. SMURFS/SMURFA are only mentioned for sake of completeness, but are not further discussed in detail.

17.3 Blender add-on

The PHOBOS Blender add-on described here and previously proposed by [10]⁷ integrates seamlessly with the GUI of the open-source 3D modeling tool Blender.⁸ It makes use of the “Outliner” panel to show all entities in the kinematic tree structure as well as grouped in collections by entity type. Using the custom properties of Blender objects, every entity can also be annotated with any custom information. In the “3D View”, the user can inspect and edit the complete model with all its entities in a WYSIWYG (“What You See Is What You Get”) environment, including collision shapes, visual representations, joint range of motion, center of mass of links, mechanical interfaces, sensors, motors driving joints, etc.

⁷ Talk at ROSCon 2015: <https://vimeo.com/142622220>.

⁸ <https://www.blender.org>, currently v3.3 LTS.

Listing 17.1: Loading, editing and exporting a robot model file via script using PHOBOS in Python.

```

1 import phobos
2 r = phobos.core.Robot(inputfile="model.smurf") # loading SDF/URDF works the same
3 root_link = r.get_root()
4 root_link.name = "new_root_link_name"
5 r.export_urdf("edited_model.urdf") # exporting SDF/SMURF works the same

```

Additionally, information about parallel mechanisms can be provided (cf. [11]). Fig. 17.3 shows an exemplary screenshot of a robotic arm.

In the backend, all this information is present in Blender’s internal representation but can be converted to the PHOBOS-API’s robot representation. This is in several ways advantageous: While being generic and compatible with any information that may in the future be added, it also helps developers to easily access the data without having to deal with Blender’s internal representation. For input/output, the model information is exported via the PHOBOS-API’s robot representation. This translation is also available in Blender’s Python console, allowing for API usage there.

17.4 API & tools

17.4.1 API

The PHOBOS package comes not only with the aforementioned Blender add-on, but is also a regular Python-package. This means it can be included and used in custom python scripts like any other Python-package (as shown in Listing 17.1) to have an easy interface for processing and editing robot models.

As the Python API is used in the Blender add-on, the import and export methods are inherently consistent with each other. To ensure compatibility with other software the SDF (cf. [4]) and URDF (cf. [3]) parsers comply with these respective format specifications.

The “Robot” class gives the user access to all elements of a robot (links, joints, sensors, submechanisms, and so on). After editing a robot, it can be exported again to the various formats (see Listing 17.1).

17.4.2 Command line tools

PHOBOS provides several useful scripts for common tasks that are accessible by the PHOBOS shell command.

- > `phobos assemble_smurfa`
Loads a SMURFS/SMURFA file (see Section 17.2) and exports a single URDF/SMURF of the assembly. This way, scenes can be displayed in software that is unable to load scenes or robots consisting of several parts.
- > `phobos check_hydrodyn`
Checks whether the model can be loaded in HYRODYN.
- > `phobos check_meshes`
Checks whether all meshes are available.
- > `phobos convert`
Converts the given input robot file to SDF/URDF/SMURF depending on the specified output file. If a PDF is given as output file, PHOBOS creates a tree view of the robots kinematic with detailed joint information including to which submechanisms they belong.
- > `phobos preprocess_cad_export`
Preprocesses CAD-to-URDF exported URDF models to use them with the pipeline. E.g., optimizes meshes, ensures correctness of URDF. (Cf. Section 17.5.2.)
- > `phobos smurfs_in_pybullet`
Provides a scene loader to the pyBullet simulator. Therefore loads a SMURFS file and adds it either to a new or already running pyBullet instance.
- > `phobos run_pipeline [--process, --test, --deploy]`
Process simulation models automatically as explained in Section 17.5.
- > `phobos test_model`
Test the latest model using a CI-pipeline (cf. Section 17.5.3) according to the corresponding configuration file. In contrast to the `run_pipeline` this can be applied to any model, not only those that have been processed by the pipeline.

17.5 Continuous integration

When developing complex robots, changes to the robot occur in each development cycle. There are iterations regarding the design, changes by integration and later on changes to the representation for different purposes. Especially when developing software and hardware in parallel, during all this time a model of the robot is needed to be as close to the reality as possible at each respective stage. This model is not only needed for one

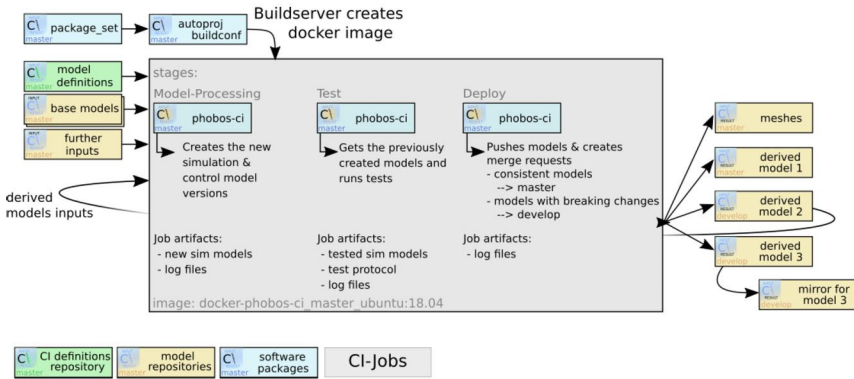


Figure 17.4 Integration schematic of a PHOBOS-CI setup.

purpose, but for several, each with different and sometimes conflicting requirements. Hence, multiple models have to be maintained.

To keep all those model versions in-sync and up-to-date with the current development state, a continuous integration (CI) pipeline tool is proposed in the following.

17.5.1 CI-pipeline-setup

As already mentioned, PHOBOS-CI can be integrated as a CI-pipeline for GIT-repositories. This way, it will run whenever there are changes either in the configuration files or the base models (CAD exported model). The basic schematic of how PHOBOS is integrated is shown in Fig. 17.4.

For the setup mentioned in the other chapters, a Docker image including an Autoproj⁹ bootstrap that has PHOBOS installed, is created by a buildserver on a regular basis.

The PHOBOS-CI pipeline works on three set of repositories: *input* repositories which contain the CAD exports (e.g., submechanism exported from a complex robot), *output* repositories, to which the models are stored after testing and processing (see Section 17.5.4) and a *definitions* repository, which contains configuration files.

In the definitions repository, the user can place configuration files defining the different model versions to maintain. There are two types of config-

⁹ Autoproj is a build system that manages several repositories and there dependencies according to a basic set of packages defined in a “buildconf”. See <https://www.rock-robotics.org/documentation/autoproj/index.html>.

uration files: First, for each model version, a file that defines that version. Second, a general pipeline file, which holds some general properties and the list of model versions to process.

A pipeline run consists of three jobs: “Process” (cf. Section 17.5.2), “Test” (cf. Section 17.5.3) and “Deploy” (cf. Section 17.5.4). The “Process” job iterates through the definition files to create the different model versions accordingly using the data from the input repository. The “Test” job then picks up the created models from the previous job’s artifacts, and the “Deploy” job finally pushes the models to the output repositories.

This setup and usage has been tested and validated using the GitLab-CI¹⁰ for the RH5 Humanoid and Recupera-Reha Exoskeleton.

An example for the definition of one of these jobs is displayed in Listing 17.2.

17.5.2 Processing workflow

Before uploading CAD-exported models to the input repositories, pre-processing them might be a good idea. As mentioned in Section 17.1.2, optimizing the exported meshes not only saves memory but also computation time in the software that uses the models later on. Doing so can reduce mesh file size down to 10–20%. This can be done either by using the preprocessing tool of PHOBOS (see Section 17.4.2) or manually by using the PHOBOS Blender add-on, with the latter enabling a more detailed optimization. Once the preprocessed CAD-exported models have been uploaded, they can serve as base models to create model versions.

For each model version a definition file has to be present. Each model version file lists from which base models this model version is derived. The source can either be an input repository, another model version (that is processed before) or a combination of both. If depending on more than one base, a definition is included on how these base parts are assembled. This allows parts to be reused so that they have to be designed and exported from CAD only once (cf. Section 17.1.6). Renaming, changing transformations, even mirroring of parts is possible in this step. When mirroring right-handed coordinate frames are ensured in such a way that the joint motion is symmetric as well. If the user needs another frame orientation, this can be simply specified in the following. Also, sub-parts of a base part can be used for assembly. During the assembling process, the annotations of the parts are inherited to the new model, making them reusable there.

¹⁰ For further information on GitLab-CI, refer to <https://docs.gitlab.com/ee/ci/>.

Listing 17.2: GitLab-CI configuration exemplary for the processing job. “Test” and “Deploy” simply use the `–test` resp. `–deploy` option of the `phobos run_pipeline` command.

```

1 image: ${URL_OF_DOCKER_REGISTRY}/docker-phobos-ci_master_ubuntu:20.04
2
3 stages:
4   - model-processing
5   - test
6   - deploy
7
8 variables:
9   DOCKER_WORKSPACE: "/opt/workspace"
10  WORKING_DIR: "/opt/workspace/${CI_ROOT_PATH}"
11  DEFINITIONS_DIR: "/opt/workspace/${CI_ROOT_PATH}/${CI_MODEL_DEFINITIONS}"
12
13 before_script:
14   # Before every job some initial commands are executed to allow
15   # ssh connection for pulling and pushing to the different
16   # repositories during the jobs
17
18 model-processing:
19   stage: model-processing
20   script:
21     - cd ${DEFINITIONS_DIR}
22     - phobos run_pipeline --process --logfile "public/process.log" "pipeline.yml"
23 after_script:
24   # deleting unnecessary meshes before staging the artifacts
25   - rm -rf "${CI_PROJECT_DIR}/temp/temp_*/repo"
26   - rm -rf "${CI_PROJECT_DIR}/temp/temp_*/combined_model/mesh*"
27 artifacts:
28   when: always
29   paths:
30     - public/
31     - temp/
32
33 # here are similar definitions for test and deploy placed

```

Once the assembly is done, various other operations can be performed on the model version by the pipeline. Apart from adding and transforming frames/links, renaming entities and changing collision geometries, the model can be annotated or information can be overridden (e.g., one version of the output model needs the hardware joint limits, another the software joint limits). More elaborate operations can be applied, too. These include:

- From the given basic submechanism and joint definitions the complete submechanism definition for HYRODYN is generated.
- The KCCD (Kinematic Continuous Collision Detection Library)¹¹ model for collision avoidance can be generated automatically.

¹¹ A library that models bodies by a convex hull that is wrapped around a certain number of points in a specific distance. Compare [9] and [12].

- If not given from CAD, the center of mass positions for each link can be estimated from the collision geometries.
- To be fully customizable, the processing job also provides an option to run custom Python scripts on the newly generated model.

17.5.3 Test for model integrity & consistency

During robot development, especially in larger projects, many dependent software packages are maintained. As models are a vital and deeply integrated part in robotic applications, changes of the models breaking functionality in unforeseen ways can lead to damage to hardware and significant expenditure of time for debugging.

To prevent such issues, a test routine is included with PHOBOS-CI to check each new model iteration for consistency. The user can define a list of tests to check for changes considered problematic, including definitions of tolerances for numeric values such as mass or transformations.

The checks distinguish in tests that compare the new model with the previous approved version and tests that check for self-related properties. Two mandatory checks are always performed to guarantee *file structure consistency*, ensuring that no files have been (re-)moved (as this might lead to file-not-found-issues in code using the model), and general *kinematic integrity* (e.g., all links are attached by a joint, only one root, etc.). Additionally, the user may add optional test operations to compare with the former model version to ensure *link mass consistency*, *link transformations consistency*, and *torque consistency* (robot's joint torques are in the current pose consistent with the previous version) with the previous version. On the self-related side the user can check whether *link mass symmetry*, *link transformation symmetry* (whether links are symmetrically located left and right). and *torque symmetry* (right and left joints are symmetric regarding their torque values) are given. Also, two further tests check whether the new model is loadable in PYBULLET and HYRODYN. Finally, the *Change pose* operation can be specified in the list of tests, which puts the robot's joints to the given configuration. All tests after this entry in the test procedure will then be run with the newly specified robot pose. This way, e.g., link transformation symmetry can be used to check if the robot is also symmetric in other poses than the zero/default configuration.

17.5.4 Model deployment

While processing and testing can be run locally, the deploy job is designed for CI-pipeline integration. The deploy job of PHOBOS-CI receives the

test results, and according to them will push the processed models to their repositories. If any test has failed—which means there are breaking changes—PHOBOS-CI will push the new model version to a separate branch and create a merge request. A human can then check on this merge request, decide whether the breaking changes are wanted and everything else is ready for these changes. If so, the changes can simply be merged. If not, the new model can be adapted on the pipeline’s input/definition side until it is correct and ready. Only if all tests for a model version have succeeded, the new model is directly pushed to the main branch.

17.6 Conclusion & outlook

PHOBOS delivers easy-to-use tools to enhance URDF models from a CAD export to fully usable robot models for simulation and control. With the Blender add-on, model annotations can be made and edited in a 3D environment. Once a model is prepared, all dependent models can be maintained using the PHOBOS-CI. With the command line tools the user is equipped with easy and fast utilities for model inspection and maintenance.

PHOBOS and SMURF are open-source¹² and under continuous development. This development aims to further deliver support and compatibility with even more modeling requirements and applications. While already compatible with URDF and SDF and supports almost all these particularities, SRDF support is under development, too.

Furthermore, increasingly extensive functionalities are planned to be integrated. This includes GUI editable configuration files for model versions maintained by the CI-utility. Supporting different levels of annotations in one model (e.g., joint hard-limits and soft-limits, detail levels of collision representations) and dealing with reconfigurable modular robots are further points for future development.

Blender add-ons operate on Python, therefore the implementation of PHOBOS happened in Python, too. Nevertheless, a translation of the software to C++ is another goal in further development; a C++ implementation with Python bindings would contribute to a notable increase in processing speed.

Despite all these future goals to get even better, PHOBOS already provides an extensive set of tools and utilities for successful creation of highly

¹² PHOBOS & SMURF: <https://www.github.com/dfki-ric/phobos>, SMURF-Parser: https://github.com/rock-simulation/smurf_parser. Contributions are welcome.

functional robot models all in one software solution and has found a growing community in the ROS (Robot Operating System) and ROCK (Robot Construction Kit) world due to its high compatibility and flexible extensibility.

References

- [1] S. Kumar, K.A. von Szadkowski, A. Mueller, F. Kirchner, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, *Journal of Mechanisms and Robotics* 12 (2) (2020).
- [2] R. Kumar, S. Kumar, A. Mueller, F. Kirchner, Modular and hybrid numerical-analytical approach – a case study on improving computational efficiency for series-parallel hybrid robots, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 3476–3483, <https://doi.org/10.1109/IROS47612.2022.9981474>.
- [3] I. Sucas, J. Kay, C. Lalancette, S. Loretz, Unified robot description format v1.13.2, <http://wiki.ros.org/urdf>, Sep 2022.
- [4] N. Koenig, S. Peters, et al., Simulation description format v1.9, <http://sdformat.org/spec>.
- [5] I. Sucas, Semantic robot description format v1.10, <http://wiki.ros.org/srdf>, Jul 2022.
- [6] M. Ivanou, S. Mikhel, S. Savin, Robot description formats and approaches, in: 2021 International Conference “Nonlinearity, Information and Robotics” (NIR), IEEE, 2021, pp. 1–5.
- [7] N. Albergo, V. Rathi, J.-P. Ore, Understanding Xacro misunderstandings, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE, 2022, pp. 6247–6252.
- [8] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
- [9] D. Mronga, T. Knobloch, J. de Gea Fernández, F. Kirchner, A constraint-based approach for human–robot collision avoidance, *Advanced Robotics* 34 (5) (2020) 265–281.
- [10] K. von Szadkowski, M. Langosz, Phobos: a blender plugin for creating robot simulation models, Tech. Rep. 1406, DFKI, September 2015.
- [11] S. Kumar, K.A. von Szadkowski, A. Mller, F. Kirchner, HyRoDyn: a modular software framework for solving analytical kinematics and dynamics of series-parallel hybrid robots, in: International Conference on Intelligent Robots and Systems, 2018.
- [12] H. Täubig, U. Frese, A new library for real-time continuous collision detection, in: ROBOTIK 2012; 7th German Conference on Robotics, VDE, 2012, pp. 1–5.

This page intentionally left blank

CHAPTER 18

HyRoDyn: Hybrid Robot Dynamics

Shivesh Kumar^a, Rohit Kumar^a, Mareike Förster^a, Andreas Müller^b,
and Frank Kirchner^{a,c}

^aRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

^bInstitute of Robotics, Johannes Kepler University, Linz, Austria

^cWorking Group Robotics, University of Bremen, Bremen, Germany

The software library HyRoDyn stands for Hybrid Robot Dynamics. It implements modular methods for solving kinematics and dynamics of highly complex series-parallel hybrid robotic systems in minimal coordinates. It combines analytical and numerical methods to resolve loop closure constraints in an error free manner keeping a balance between generality and computational efficiency. The content of this chapter is based on [1] and [2].

The chapter is organized as the following: Section 18.2 presents the robot description format for HyRoDyn and Section 18.3 describes a visual editing tool Phobos for generating such description files. Section 18.4 presents some software implementation details of HyRoDyn and Section 18.5 presents the integration of HyRoDyn in a robot middleware operating system called RoCK. Section 18.6 presents the Python interface to HyRoDyn software and the automated robot analysis tooling based on HyRoDyn.

18.1 Motivation

Building a robot is a highly complex process that requires cross-domain expertise. Robots are becoming so complex that it is kind of impossible for a single person to manage the process of robot development. We restrict the scope of robot development to its conceptualization, construction, kinematic, dynamic modeling and position-velocity-torque control. These are the absolute minimum steps that are needed in the development process of most robots before more complex behaviors are realized with the system. To aid the further explanation, three roles are identified:

- **Designer:** Designer is the person responsible from initial design conceptualization to implementing the mechatronic aspects in the final

prototype. A designer is an expert in CAD software and may also know some CAE tools (e.g., ADAMS MSC, RecurDyn) for immediate analysis and simulation of their design.

- **Kinematician:** Kinematician or geometer is a domain expert in the field of geometric and kinematic analysis of mechanisms. Such a person is expected to have strong foundations in mathematics in particular in the area of geometry and mechanics.
- **Control Engineer:** Control Engineer is the person responsible for implementing low, mid or high level controllers inside a robot. They are considered to be experts in control theory and have a good understanding of applied mechanics.

The main motivation behind the HyRoDyn software architecture is to provide a holistic treatment for kinematic and dynamic modeling of series-parallel hybrid robotic systems, while involving the three above defined roles into the process. In [3], a systematic survey on the state of the art in series-parallel hybrid robots is presented. Designers find it increasingly useful to utilize parallel kinematics-based submechanism modules in their design to achieve a good payload-to-weight ratio, stiffness properties and dynamic properties. While they can perform a basic simulation of their design in CAD or CAE software, they often turn to domain level experts to optimize their design further. The presence of closed loops in robots significantly increases the complexity of the kinematics and dynamics problems associated with multi-body systems (MBS). Hence, most multi-body dynamics libraries or software packages support only serial or tree type mechanisms and provide analytical formulations for solving forward kinematics, forward and inverse dynamics. Inverse kinematics is usually solved through numerical techniques. Further, it has been noted that a limited number of tools that do provide the possibility of modeling closed-loop systems deal with loop closure constraints numerically. This allows for a general treatment of mechanisms, but only leads to a limited *local* kinematics and dynamics analysis of these systems without deeper insights that are typically required in the design and analysis of complex mechanisms. Further, the numerical approaches may suffer from inaccuracies and computational inefficiency. Hence, a big portion of research done by the kinematics community is to provide comprehensive geometric/kinematic analysis of mechanisms of specific class or type [4,5] (e.g., see Chapters 3–5). This kind of analysis is usually a one-time effort and very useful in the design optimization, e.g., removing singularities from the feasible workspace, optimizing force and velocity transmission of the parallel joint modules in

different applications (e.g., wrist, ankle or torso design). Typically, this kind of feedback is given by the geometrician/kinematician to the mechanical designer to optimize their design. This information is also useful for the control engineers to model the kinematics and dynamics to realize a position, velocity or torque control in the robot. However, there is no effort yet to make this kinematic analysis reusable in the context of design and modeling closed-loop systems and the mechanisms that can be derived from their compositions. There is, however, a very practical need to do so because of the growing popularity of series-parallel hybrid designs in robotics. It is becoming increasingly desirable to analyze and control these robots accurately and efficiently. In [6], analytical and modular methods for solving the kinematics and dynamics of series-parallel hybrid robots are presented based on the concept of loop closure functions. The main idea behind HyRoDyn is to store these LCFs in configurable submechanism libraries to form a PKM software database. This leads to an analytical and modular software workbench that optimizes the above workflow and make the process efficient and reusable. Later, HyRoDyn was extended to include numerical loop closure for submechanisms for which analytical solutions are not yet known [7]. Hence, we have a good balance between exploitation of domain-specific knowledge (closed-form solutions for PKM modules) and generality (unknown mechanisms can be resolved numerically) in this software. It fits into the overall picture of x-RoCK project series [8] which attempts to streamline and simplify the robot development process. In the following, HyRoDyn developer and user workflows are discussed.

18.1.1 Developer workflow

An overview of HyRoDyn developer workflow is demonstrated in Fig. 18.1. The designer usually starts with a high-level design specification, which is often overly simplistic in nature. An example can be to design a 32 DOF humanoid robot with height 1.60–1.80 m and weight 60–70 kg which can carry 5 kg payload in each arm. The designer performs a simulation study to deduce further design criteria which includes required range of motion in each joint, torque requirements. They may start with a walking simulation of a simple stick figure humanoid (where links are simplified as point masses) to get an estimate of torque requirements in the system for the desired height and weight ranges. Once a rough estimate of range of motion, torque requirements is at hand, the designer starts with the mechanism as well as actuator design. The designer often takes inspiration from nature and tries to abstract the biological system

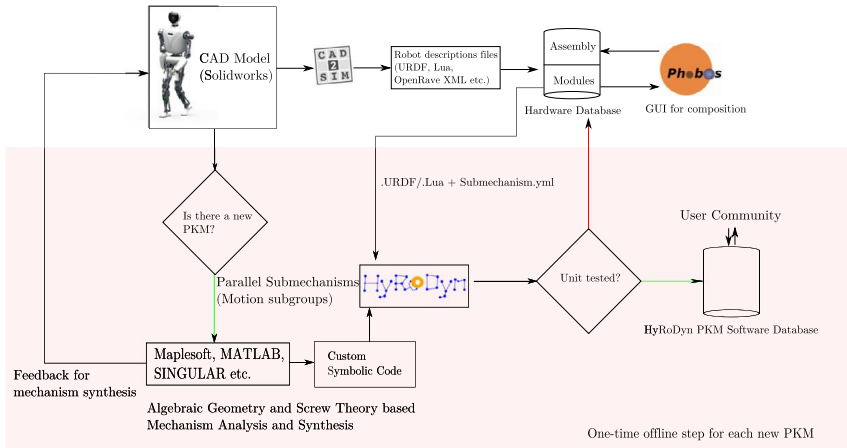


Figure 18.1 HyRoDyn Developer Workflow (adapted from [1]).

with well-studied kinematic joint types (revolute, universal, spherical, etc.). For example, a humanoid leg might be constructed as a serial mechanism of type spherical-revolute-universal (SRU) or spherical-revolute-spherical (SRS). The kinematic joints in this design may further be realized using already existing closed loop mechanisms or PKM modules in order to optimize mass-inertia properties of links, dynamic performance, stiffness properties, etc. In the process of doing so, they may invent a new parallel mechanism, the preliminary analysis of which can be conducted in typical CAD software. However, in order to further optimize the design, a complete kinematic analysis is desirable.

At this step, the designer communicates with the kinematician to perform a detailed analysis. Based on the topological description of this new mechanism, the kinematician formulates the constraint equations of the mechanism and analyzes the geometric conditions under which these equations are valid.¹ The kinematician should try to keep these constraint equations as generic as possible by maintaining a balanced trade-off between the principle solvability of the equations and computational efficiency in solving them. They may use modern computer algebra tools (e.g., MATLAB®, Maple, Mathematica, SINGULAR, etc.) for analyzing and solving these

¹ While the topological description carries indicative information about the mobility of the mechanism, the true mobility of the mechanism depends on the configuration and choice of geometric parameters.

constraint equations. Various key insights like maximum number of solutions to forward/inverse kinematics, singularity curves of the mechanism, velocity/force transmission can be derived. This feedback can be provided to the designer for improving the mechanism design. The symbolic solution of these constraint equations can be exported in the form of efficient C-code and transferred into loop closure function of the submechanism library in HyRoDyn's PKM software database. More details about writing a submechanism library will be provided later in this chapter. The PKM software database in HyRoDyn will grow as developers around the world can contribute to and benefit from the existing submechanism libraries.

The control engineer, working with the designer and kinematician would export the CAD model of the submechanism into robot description formats (e.g., URDF) using tools like CAD2SIM or Solidworks2URDF exporter. This hardware model can be unit-tested with the HyRoDyn library. At this stage, they can already realize a physical prototype of the mechanism and test the model for control purposes. Due to the modular approach here, it becomes easier to debug the software and improve the mechanism's model. On successful testing, the model can be stored in the x-RoCK hardware database.

18.1.2 User workflow

Once, the submechanism design has been optimized, the designer repeats the same process for designing other submechanism modules and composes the overall model from it. For example, a humanoid leg of type SRU can be realized with a composition of RR module, 1-RRPR for hip flexion-extension, 1-RRPR for knee joint and 2-SPRR+1U module for the ankle joint. Preliminary simulation of the overall CAD model can be performed by the designer and the design can be optimized.

The kinematician at this stage checks if its feasible to a derive closed form solution for the abstracted serial SRU chain. If yes, such a model can be provided to the control engineer.

The control engineer exports the robot description of all the submechanism models (including source and sink transforms) and test it using HyRoDyn. Then, a Blender-based visual editor called Phobos is used to compose the complete robot model and its description is exported. This provides a user-friendly way to compose highly complex models. Once the overall model has been tested, it is added to the xRoCK hardware database. These models can be used to implement position, velocity or torque control in the robot and can be used in conjugation with more complex control

architectures like whole-body control. The forward algorithms in HyRoDyn can be used for simulation purposes and the inverse algorithms can be used for analysis and control. The overall user workflow is depicted in Fig. 18.2.

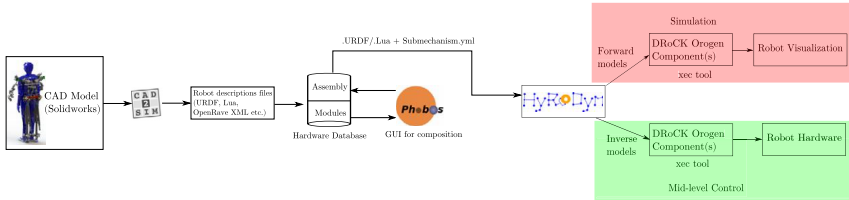


Figure 18.2 HyRoDyn User Workflow (adapted from [1]).

18.2 SMURF: robot description for HyRoDyn

To overcome the limitations of URDF, which mainly deals with geometric and physical parameters of a robot, but is arguably the most widely-used format in the scientific robotics community, the Supplementable Mostly Universal Robot description Format (SMURF) format has been developed at DFKI-RIC, augmenting URDF by annotating data with reference to its links and joints [10]. To accommodate the definition of the kind of modular mechanisms as described in this thesis, SMURF was updated to allow the specification of sub-mechanisms mapped on the spanning tree defined in a URDF file, thus preserving the possibility to define an explicit spanning tree on a looped graph, while still adding the information relevant to identifying the nature of loop closure constraints in the mechanical system.

18.2.1 Parallel submechanisms with known analytic solutions

The submechanisms definition is exported in the form of a YAML file as:

```
...
- name: <SUBMECHANISM_NAME>
  type: <TYPE>
  file_path: <PATH_TO_SUBMECHANISM_URDF>
  jointnames_independent: [<J1>,...,<JM>]
  jointnames_spanningtree: [<J1>,...,<JN>]
  jointnames_active: [<J1>,...,<JP>]
...
```

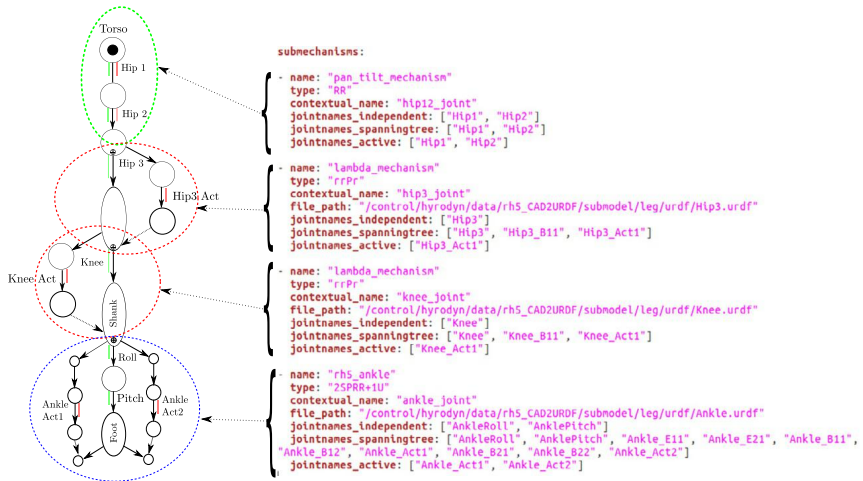


Figure 18.3 Example of YAML based submechanism description of RH5 [9] leg.

This YAML file basically contains the list of submechanisms that constitute the overall spanning tree of the robot. The parallel mechanisms are identified by a type (e.g., 1-RRPR, 2-SPRR+1U, 2SPU+1U, 6-UPS, etc.) and vectors of names of independent joints, active joints and spanning tree joints are defined. The spanning tree joint names are listed respecting the modular graph enumeration scheme. Further, it is required to provide a file path to the submechanism's URDF here for the parallel submechanism modules. Overall, to define a series-parallel hybrid robot completely, one needs to have the full URDF file of the robot itself, along with a YAML based submechanism description file which contains information about the modular composition (see Fig. 18.3 for an example of RH5 leg).

18.2.2 Generic parallel submechanism

In a further development of the user interface, the submechanism YAML file is extended to provide support for the numerical approach. The definition of loop constraints is required for the numerical approach. The key value that is used to switch between the numerical and analytical approach is through the node name *type* in the submechanism YAML file. The numerical approach can be used for a closed-loop system by changing the node name to *type: NUMERICAL*. This directs the software to find the explicit constraints numerically.

The definition of loop constraints in the YAML file includes various parameters, which include the name of the cut joint frame, predecessor body of the cut joint, successor body of the cut joint, and constraints. For each loop constraint, the user can input the name, constraint axis, or the direction in which the constraints need to be put. Note that the constraint axis should be defined with respect to the local frame of the cut joint and should be defined as a spatial vector. The developments in the submechanism YAML file can be seen below.

```

submechanisms :
- name: < SUBMECHANISM_NAME >
  type: < SUBMECHANISM_NAME / NUMERICAL >
  contextual_name :
  file_path: < PATH_TO_SUBMECHANISM_URDF >
  jointnames: < ALL JOINTS IN THE URDF FILE >
  jointnames_active: [ <J1 >, ..., <JP > ]
  jointnames_independent: [ <J1 >, ..., <JM > ]
  jointnames_spanningtree: [ <J1 >, ..., <JN > ]
  loop_constraints :
- cut_joint: < CUT_JOINT_FRAME >
  predecessor_body: < PREDECESSOR_BODY_NAME >
  successor_body: < SUCCESSOR_BODY_NAME >
  constraint_axes :
- name: < CONSTRAINT_NAME >
  axis: [< 6D_SPATIAL_VECTOR >]
  baumgarte_stabilization_parameter: < VALUE >

```

18.2.3 Example: RH5 Manus

A case study of reduced version of the RH5 Manus robot [11] is considered. The reduced version of the robot is shown in Fig. 18.4.

The whole system consists of 8 submechanisms, which are:

1. Torso Body: It is a parallel kinematic chain of the type 2SPU+1U. It is a multi-loop mechanism. Here, the independent or input joint is the universal joint and the actuated joint is left and right prismatic actuator. This can be seen in Fig. 18.5a.
2. Body Yaw: It is a serial chain with single revolute joint.
3. Left shoulder: It is a serial chain with 3 revolute joints.
4. Left elbow: It is a parallel kinematic chain of the type RRPR or lambda mechanism. The independent joint is the revolute joint denoted in green and the actuated joint is the prismatic joint denoted in

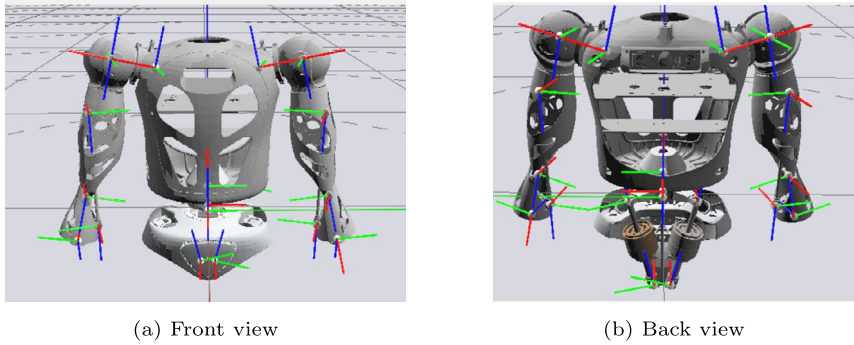


Figure 18.4 Reduced version of RH5 Manus [2].

red. This can be visualized in Fig. 18.5b. The cut joint considered here is revolute joint, defining the cut joint frame as $ALElbowAct$. The position constraints exist in y and z direction after referring to the URDF file. The predecessor and successor body are $ALElbow_Link$ and $ALElbowAct_Link$ respectively.

5. Left wrist roll: It is a serial chain with single revolute joint.
6. Right shoulder: It is a serial chain with 3 revolute joints.
7. Right elbow: It is also a parallel kinematic chain of the type RRPR or lambda mechanism as described before in Fig. 18.5b.
8. Right wrist roll: It is a serial chain with single revolute joint.

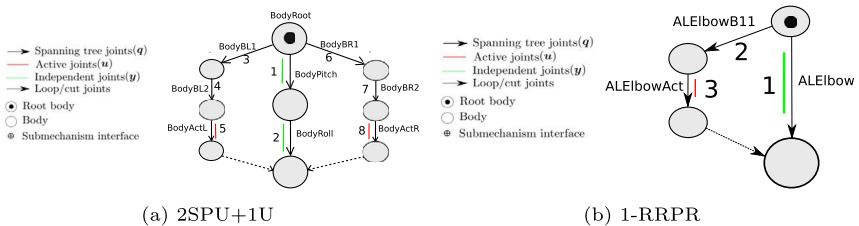


Figure 18.5 Topological graphs of torso and wrist mechanisms [2].

Fig. 18.6 gives the details about the topological graph of the whole system and the respective definitions of the submechanisms in YAML file.

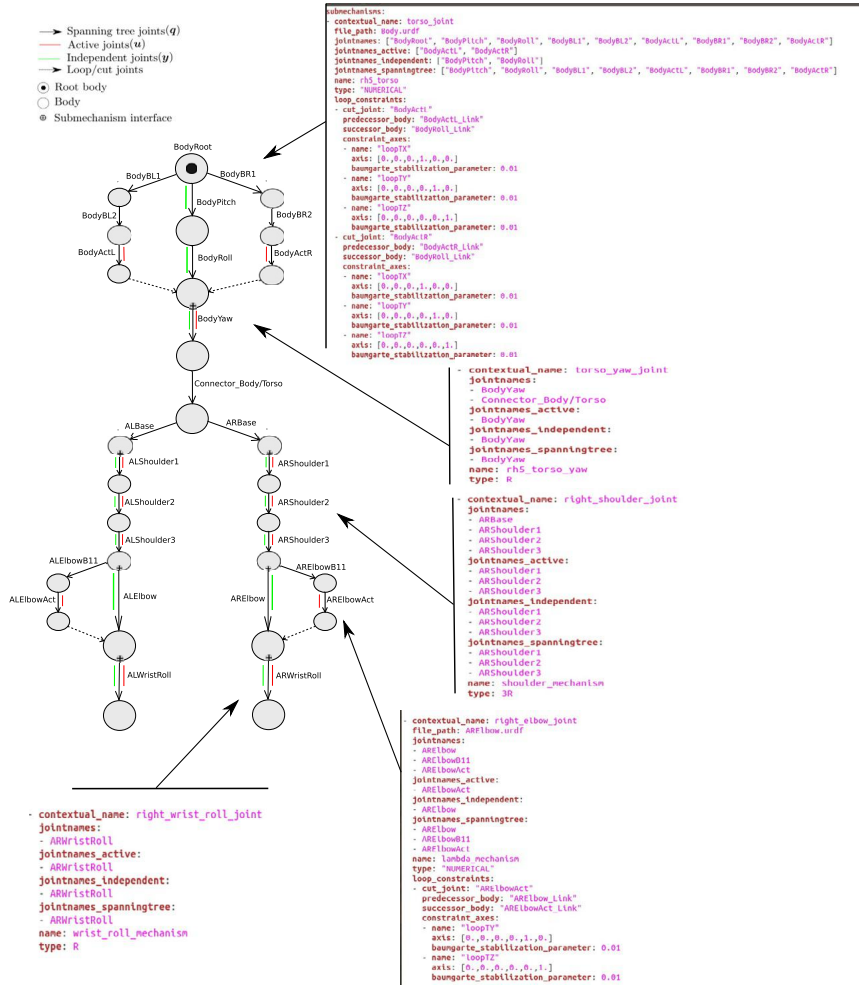


Figure 18.6 Spanning tree with submechanism definitions [2].

18.3 Phobos: visual editor for HyRoDyn

Phobos [10] is an open source visual editor for robots based on Blender developed by DFKI-RIC and is capable of exporting both URDF and the accompanying SMURF models. It is based on what-you-see-is-what-you-get (WYSIWYG) philosophy. Phobos was extended for the purpose of the methods presented here, allowing the export of sub-mechanism definitions as described above as part of the SMURF representation of a robot. The sub-

mechanisms recognized by HyRoDyn can be defined in a generic fashion in YAML files, which are parsed by Phobos and fed into an interactive operator that allows the assignment of the different joints defined for a mechanism to joints in the visual model, visualizing the components of the mechanism in the process. The relevant information for the mechanism definition is stored in the objects representing the mechanical joints and exported to SMURF together with the rest of the robot data, i.e., the kinematic model as a URDF, motor and sensor information, etc. This data can then be processed by HyRoDyn.

Due to the handling of objects in Blender in a similar spanning tree as URDF, the use of Phobos enables both a top-down or bottom-up approach to design and describe such modular series-parallel hybrid robotic systems. Since both approaches contain a number of hurdles for human designers, the use of a visual editor makes this process more reliable and simple.

18.3.1 Top-down modeling

In the top-down approach, an already existing URDF representation of a robot can be annotated with sub-mechanism definitions and the submechanism URDFs can be exported. Fig. 18.7 shows an example of top-down modeling using Phobos. A URDF of the RH5 humanoid [9] upper leg is imported into Phobos and three submechanisms namely Knee, Hip3 and Hip12 are annotated on the model. A SMURF export at this stage provides the YAML based submechanism description and individual URDFs for each annotated submechanism. For the example shown in Fig. 18.7, the submechanism description will include the first three blocks of the description provided in Fig. 18.3. The blue cone at the end of the kinematic chain in Fig. 18.7 demonstrates the sink link.

18.3.2 Bottom-up modeling

In the bottom-up approach, URDF representations of pre-defined submechanisms are assembled into one complex model. This is a powerful way of modeling complex systems as symmetry in the mechanical design can be exploited to replicate complex substructures without having to remodel them. This is demonstrated with the help of an example composition in Fig. 18.8. Fig. 18.8a shows the already annotated ankle submechanism of the humanoid leg. It is imported in the Phobos environment, which already contains the upper leg assembly (see Fig. 18.8b for an exploded view). Using the source and sink links on these submechanisms, they are

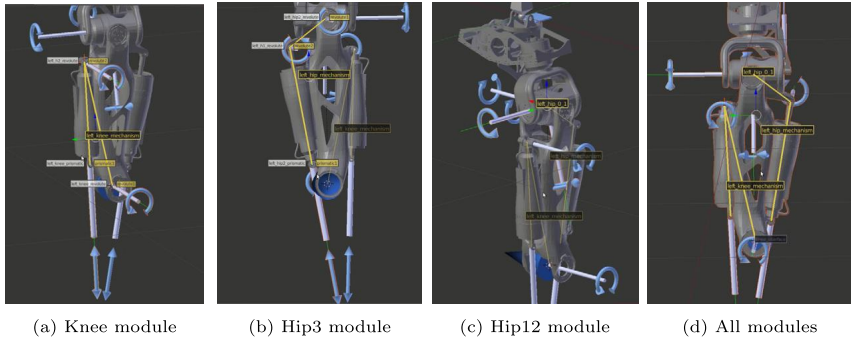


Figure 18.7 Top-Down modeling using Phobos [1].

connected to form the single leg assembly of RH5 humanoid [9], as shown in Fig. 18.8c. Since the two legs of the RH5 humanoid are symmetric in design, the leg assembly is replicated twice in Phobos. Then, the batch rename feature is used to rename left leg attributes to right leg attributes. After this step, the second leg joint in the right leg is adjusted to regain the symmetry which prepares lower body model of RH5 humanoid, as shown in Fig. 18.8d.

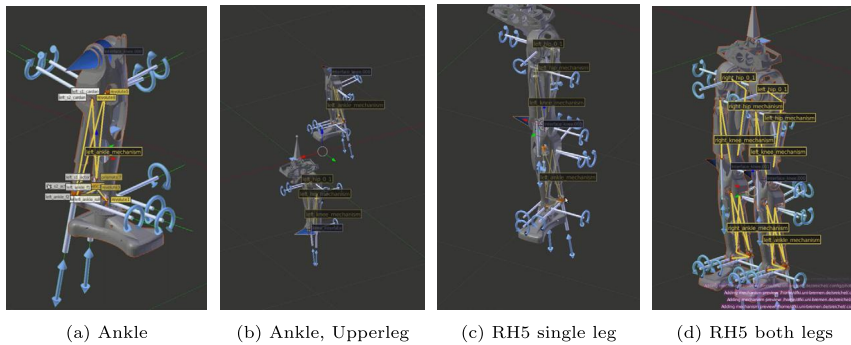


Figure 18.8 Bottom-Up modeling using Phobos [1].

18.4 HyRoDyn software library

Hybrid Robot Dynamics (HyRoDyn) is a software library that implements the modular and analytical formulations for series-parallel hybrid

robotic systems. It is implemented in C++ and utilizes the implementation of multi-body dynamics algorithms for tree type systems (based on Featherstone [12]) from the Rigid Body Dynamics Library (RBDL) [13]. Additionally, it depends upon the linear algebra package Eigen 3 [14] and `yaml-cpp`² for parsing the modular submechanism definitions provided in SMURF model. The functions implemented in HyRoDyn have a model-based interface that separates the models from the algorithms. The library is completely independent of any middleware software used in robotics community.

18.4.1 Submechanism libraries

The most crucial aspect of HyRoDyn is its submechanism libraries, which contain the symbolic expressions for the loop closure function of a particular type of the submechanism. Each submechanism library is a C++ class that inherits from an *abstract* class called `ExplicitLoopConstraintSet`. The abstract class defines the matching member variables and functions that the derived submechanism class must have. Fig. 18.9 shows the relationship between the abstract class `ExplicitLoopConstraintSet` and child submechanism classes and the generic numerical loop constraint class with the help of a class diagram. Hence, an object of the abstract class `ExplicitLoopConstraintSet` is basically an abstract mechanism, which defaults to a serial mechanism, because the loop closure function for a serial chain defaults to identity function.

18.4.1.1 Known submechanisms with analytical solutions

A submechanism class is specific to a mechanism type which must have a member variables as geometric parameters of the mechanism, member functions as loop closure functions, and a constructor that loads the robot model and extracts the geometric parameters of the robot. For example, see the 1-RRPR and 2-SPRR+1U classes in Fig. 18.9. The class structure is completely agnostic to any specific method (solution to trigonometric equations, algebraic approach, etc.) for solving the loop closure constraints inside the class. While the general guideline is to derive analytical expressions for LCFs, one may also use (hybrid)-numerical methods inside them [4].

² <https://github.com/jbeder/yaml-cpp>.

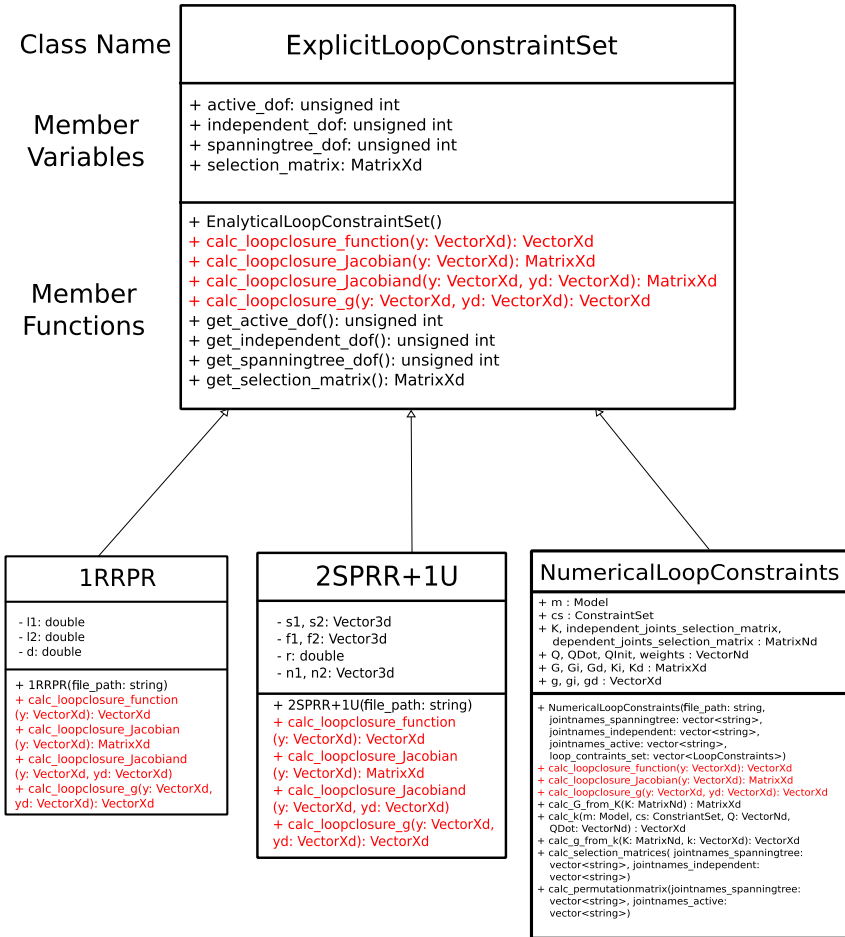


Figure 18.9 Class diagram demonstrating the relationship between the abstract class `ExplicitLoopConstraintSet`, submechanism classes and generic numerical loop constraint class [2].

18.4.1.2 Generic submechanisms with numerical solutions

To improve the generality of HyRoDyn software, a `NumericalLoopConstraints` class is also provided, which can solve any general PKM submechanism module. In this way, HyRoDyn can exploit the analytical solutions to known submechanism types in a robot and use numerical solutions for the other submechanisms where the analytical solutions are not available yet [7]. The constructor of the `NumericalLoopConstraints` class takes arguments that

include the file path of the URDF, various joint names, and loop constraint set. The loop constraint set is passed through a YAML parser from the submechanism YAML file. The constructor is responsible to attach the loop constraints to the model. As discussed in the previous chapter, loop constraints require parameters like body id's, cut joint frames, constraint axis, and relative transformations. Although, most of the information is provided in the submechanism YAML file, relative transformations between cut joint frame and predecessor body, and relative transformation between cut joint frame and successor body are required to be computed in the constructor of the class. This feature is user-friendly, as the transformation matrices can be cumbersome to write. The relative transformations can be found as

$${}^0\mathbf{T}_{p(k)}{}^{p(k)}\mathbf{T}_k = {}^0\mathbf{T}_k, \quad (18.1)$$

$${}^{p(k)}\mathbf{T}_k = ({}^0\mathbf{T}_{p(k)}^{-1}){}^0\mathbf{T}_k, \quad (18.2)$$

$${}^0\mathbf{T}_{s(k)}{}^{s(k)}\mathbf{T}_k = {}^0\mathbf{T}_k, \quad (18.3)$$

$${}^{s(k)}\mathbf{T}_k = ({}^0\mathbf{T}_{s(k)}^{-1}){}^0\mathbf{T}_k, \quad (18.4)$$

where ${}^{p(k)}\mathbf{T}_k \in SE(3)$ is the relative transformation between the predecessor body and the cut joint frame. Similarly, ${}^{s(k)}\mathbf{T}_k \in SE(3)$ is the relative transformation between the successor body and cut joint frame.

When the user specifies cut joint frame, predecessor body, and successor body, the transformations ${}^0\mathbf{T}_{p(k)}$, ${}^0\mathbf{T}_{s(k)}$, and ${}^0\mathbf{T}_k$ are calculated. Here, k is represents the cut joint frame, $p(k)$ denotes the predecessor body, and $s(k)$ denotes the successor body. For a defined model, it is fairly easy to compute the transformation from base to any frame using RBDL.

The constructor is also responsible for calculating the selection matrices \mathbf{Q}_i and \mathbf{Q}_d . \mathbf{Q}_i is the independent joints selection matrix of size $m \times n$, and \mathbf{Q}_d is the dependent joint selection matrix of size $(n - m) \times n$.

Example 18.1. The C++ code snippet presented in Listing 18.1 shows the ease of use of HyRoDyn library. First, an object `rh5` of the `RobotModel_HyRoDyn` class is created (Line 7). By defining the file paths to the URDF and submechanism description of the RH5 humanoid leg (Lines 9 and 10), a robot model is loaded (Line 12). The Hip3 and Knee joint angles are set to 0.1 radians and 0.2 radians respectively (Lines 14 and 15) which constitutes a configuration in independent joint space $\boldsymbol{\gamma}$ of the robot. One could set the independent joint velocities and accelerations by setting the variables `yd` and `ydd` (they default to zero). Three different member functions of this class are demonstrated in this example.

Listing 18.1: C++ code for solving system state, inverse and forward dynamics using HyRoDyn.

```
#include <iostream>
#include <hyrodyn/robot_model_hyrodyn.hpp>

int main(int argc, char** argv)
{
    // Create an object of RobotModel_HyRoDyn class
    hyrodyn::RobotModel_HyRoDyn rh5;
    // Define robot description file paths
    string filepath_urdf = "leg.urdf";
    string filepath_submechanisms = "submechanisms_leg.yml";
    // Load robot model
    rh5.load_robotmodel(filepath_urdf, filepath_submechanisms);
    // Set independent joint angles
    rh5.y(2) = 0.1; // Hip3 joint
    rh5.y(3) = 0.2; // Knee joint
    // Compute the system state of the robot
    rh5.calculate_system_state();
    cout << "System_State_Output:" << rh5.Q.transpose() << endl;
    // Compute inverse dynamics of the robot
    rh5.calculate_inverse_dynamics();
    cout << "Inverse_Dynamics_Output:" << endl <<
    rh5.Tau_actuated.transpose() << endl;
    // Compute forward dynamics of the robot
    rh5.calculate_forward_dynamics();
    cout<<"Forward_Dynamics_Output:" << endl <<
    rh5.ydd.transpose() << endl;
    return 0;
}
```

Listing 18.2: Output of C++ code in Listing 18.1.

```
System State Output:
0 0 0 0.1 0.120279 -0.00973191 0.2 -0.0393719 0.0129114 0 0 0
0 0 0 0 0 0 0
Inverse Dynamics Output:
-1.2033 -0.0178554 -49.4034 36.2226 -1.48666 -1.48665
Forward Dynamics output:
-9.3213e-17 2.2155e-16 6.3693e-16 -7.4643e-16 -5.0238e-16 1.7903e-14
```

- The complete system state of the robot can be computed by calling the member function `calculate_system_state()`. The output of this function is stored in the member variables `Q`, `QDot`, `QDDot` (see Line 18–19 for the position output of the spanning tree).
- The inverse dynamics of the robot can be computed by calling the member function `calculate_inverse_dynamics()`. The output of this function is stored in the member variable `Tau_actuated` (see Line 21–22 for the output actuator torques).
- The forward dynamics of the robot can be computed by calling the member function `calculate_forward_dynamics()`. The output of this function is stored in the member variable `Tau_actuated` (see Line 25–26 for the output independent joint accelerations).

Table 18.1 Robotic systems tested with HyRoDyn.

Robotic system	Spanning tree DOF (n)	Independent DOF (m)	Active DOF (p)	Independent loops (c)
UR5	6	6	6	0
Recupera	18	5	5	2
RH5 leg	18	6	6	4
RH5 both legs	36	12	12	8
RH5 full	71	29	29	14

The output of this simple program can be found in Listing 18.2. Notice in Line 2 that the position state of the spanning tree is zero for all indices, except for indices belonging to the Hip3 and Knee submechanism modules. The velocity and acceleration state of the spanning tree are zero vectors, because there was no input velocities and accelerations to the model and hence not shown here. The inverse dynamics output is shown in Line 4, which corresponds to the gravity torques in this configuration, since the velocity and acceleration are set to zero. The forward dynamics output shown in Line 6 also corresponds to an almost zero vector, since the output of inverse dynamics is the input to forward dynamics function.

18.4.2 Computational performance

The computational performance of HyRoDyn is evaluated by testing it with different robot models (e.g., UR5, Recupera Wheelchair system, RH5 humanoid) of varying complexity for different algorithms (SYSSTATE, IDYN, FDYN), presented in this thesis. Active, independent and spanning-tree DOF for these systems as well as number of independent closed loops present in them are specified in Table 18.1. The computational performance is measured in terms of CPU time for 100,000 calls to these methods for randomized input in independent joint space, respecting the joint limits. Fig. 18.10 shows the average CPU time needed for using these methods for the different robotic systems. These tests were performed on a standard laptop with Intel Core i7 CPU @ 2.8 GHz.

18.5 Integration in middleware

A middleware is a software framework that enables distributed processes to exchange data on heterogeneous platforms. This software bus uses an object map to offer a simple and coherent interface for accessing objects and to guarantee data transmission. Probably, the most frequently classical used

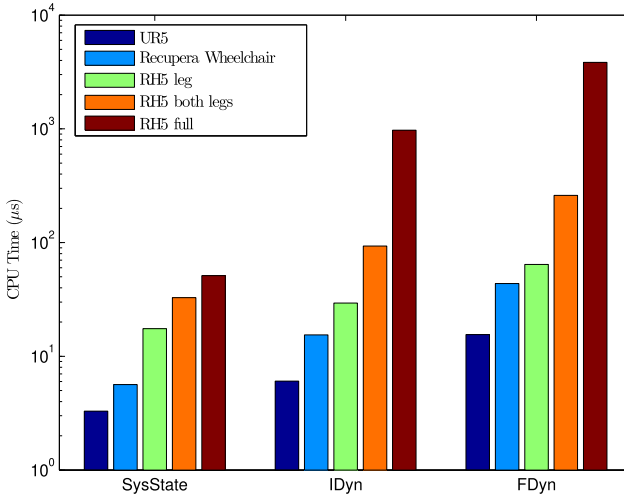


Figure 18.10 Computational performance of HyRoDyn to solve the system state, inverse dynamics and forward dynamics for different robotic systems described in Table 18.1 [1].

robot middlewares are the Robot Operating System (ROS), Yet Another Robot Platform (YARP), Robot Construction Kit (RoCK), OpenRTM, etc.

HyRoDyn is implemented as an Orogen component in the Robot Construction Kit (RoCK) [15] which is based on the component model of the Orocos Real Time Toolkit (RTT) and the object request broker (ORB) implementation, omniORB. Components encapsulate different functionalities or tasks, run independently and provide input and output for other components. The configuration can be applied to each component individually. HyRoDyn-rogen component implements various tasks which helps the end user exploit different mappings required for implementing robot behavior. The inverse task as shown in Fig. 18.11a takes as input the motion trajectories defined in the independent joint space ($\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$) and computes the actuator trajectories ($\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, \boldsymbol{\tau}_u$). This is crucial for abstraction of the robot in independent joint space, so that higher level of control can be done by treating the robot as a tree type system. The forward task as shown in Fig. 18.11b takes as input the actuator status ($\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}$) and computes the independent joint status ($\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$) of the robot. Both tasks compute the full system state of the spanning tree ($\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$) and make it available on an output port which can be used for robot visualization. The

inverse kinematics task shown in Fig. 18.11c provides mapping from task space ($\mathbf{T} \in SE(3)$, $\mathbf{V} \in se(3)$, $\dot{\mathbf{V}} \in \mathbb{R}^6$) to independent joint space ($\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$), while respecting the loop closure in the robot. Additionally, it is possible to constrain the COM ($\mathbf{r}_{\text{com}} \in \mathbb{R}^3$) of the robot, which is useful for floating base robots, e.g., humanoids. Finally, the controller task implements inverse dynamics controller in actuation space, as shown in Fig. 18.11d. The input-output interface of various tasks in HyRoDyn-rogen component is shown in Fig. 18.11. The component can be configured by a SMURF file (URDF and YAML based submechanism file) and provides a bi-directional mapping between the independent joint space and actuator space of the robot. These tasks can be used for both real-time simulation and control of series-parallel hybrid robots. The component-based architecture of RoCK framework makes it easy to reuse the same component in different robotics applications.

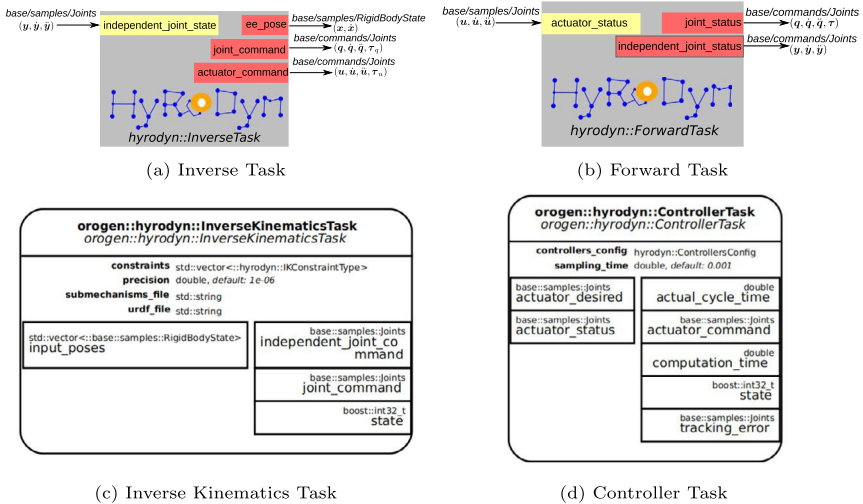


Figure 18.11 HyRoDyn orogen components in RoCK.

18.5.1 Interfacing with other components

An advantage of component-based software architecture is that it allows interfacing with other components to exchange data so that complex applications can be designed in a modular way. HyRoDyn orogen component can be interfaced with other components by following the guidelines below.

- To keep the configuration requirement of the component minimal, the output of the component is decided based on its input.
 - If only position is provided, HyRoDyn only computes the position and static forces/torques (gravity terms).
 - If position and velocity are provided as inputs, HyRoDyn computes output position+velocity with required forces/torques (coriolis-centrifugal + gravity terms).
 - If position, velocity and acceleration are provided as inputs, HyRoDyn computes output position, velocity, acceleration with required forces/torques (mass-inertial + coriolis-centrifugal + gravity terms).
 - If effort is provided on the input port, it is assumed that inverse dynamics of the abstracted mechanism is being solved outside the component and hence, the component simply transforms these torques into the actuation space. This feature can be used if model order reduction is desired for inverse dynamics computations.
- HyRoDyn component takes a very strict approach towards plant modeling in mechanics domain. The input and output ports should strictly correspond to the joints available in the spanning tree. It does not bypass any joint status/commands if they are not available in the urdf file and submechanism file.
- Floating base robots, such as humanoids, are modeled using a six DOF free flyer joint, which should be defined explicitly in both YAML based submechanism description file and the robot's URDF. A combination of six one DOF joints is added to the urdf in the following sequence: X, Y, Z, RX, RY, RZ. This becomes the part of both independent joint space input and joint space command output. The actuator space remains the same. HyRoDyn expects the floating base coordinates (from other components or Inertial Measurement Unit).
- The input and output ports operate on SI units, i.e., for rotary joints: position is measured in rad, velocity in rad/s, torque in N m, etc., and for linear joints: position is measured in m, velocity in m/s, force in N, etc. Torque (N m) to current (A) conversion or position (rad/m) to motor ticks conversion must be done outside this component.

18.5.2 Examples

HyRoDyn component can be used in various application designs. In the following three examples, we discuss its application in gravity component control, task space control and independent joint space control of a robot.

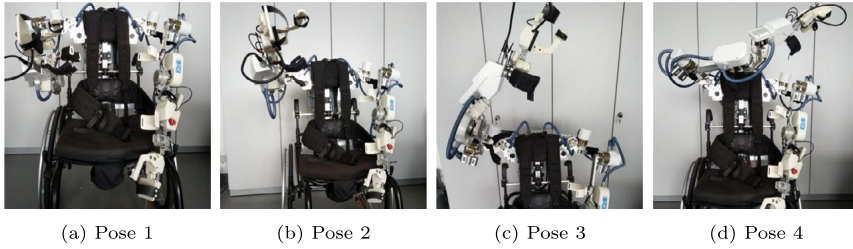


Figure 18.13 Gravity compensation for four different poses [18] (Licensed under [CC BY 4.0](#)).

Table 18.2 Commanded torque, measured torque and MAE for 4 different poses shown in Fig. 18.13 [18] (Licensed under [CC BY 4.0](#)).

Pose	Commanded Torque (Nm)	Measured Torque (Nm)	$\ MAE\ $
1	$(-3.72, 0.85, -5.02, -2.99)^T$	$(-3.78, 0.86, -4.84, -2.99)^T$	0.2631
2	$(-4.70, 1.42, -9.62, -3.24)^T$	$(-4.59, 1.42, -9.51, -3.24)^T$	0.1536
3	$(-1.59, -1.91, -5.37, -0.41)^T$	$(-1.56, -1.90, -5.48, -0.42)^T$	0.1162
4	$(5.40, -3.86, -8.24, -2.09)^T$	$(5.27, -3.85, -8.31, -2.09)^T$	0.1435

Example 18.3. HyRoDyn orogen component can be used for whole-body control (WBC) applications where the chosen solver only has to work for a tree-type system. Fig. 18.14 shows its application in WBC of the RH5 humanoid using the RoCK framework. WBC component takes as input a list of task space trajectories and outputs the independent joint space velocities for the robot by exploiting the redundancy of the Jacobian. It is then fed to an interpolator component which generates the position, velocity and acceleration command for the robot. HyRoDyn component solves the joint-space inverse dynamics and generates the actuator space commands, which are then fed to the NDLCOM device drivers and further sent to the low level actuation space controllers implemented on FPGA stacks in each joint.

Two set points, i.e., foot up position and foot down position, are chosen in the task space of the robot. Using whole-body control, the independent joint positions needed to reach these points are computed for the abstracted serial robot. Then, these waypoints in independent joint space are fed to an interpolator, which provides smooth trajectories $(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}})$ for up and down movement of the leg. Fig. 18.16 shows the task space and independent joint space trajectories for the RH5 leg to produce this movement. These trajectories are then used to compute the actuator trajectories $(\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, \boldsymbol{\tau}_u)$, using inverse kinematics and inverse dynamics algorithms as presented earlier.

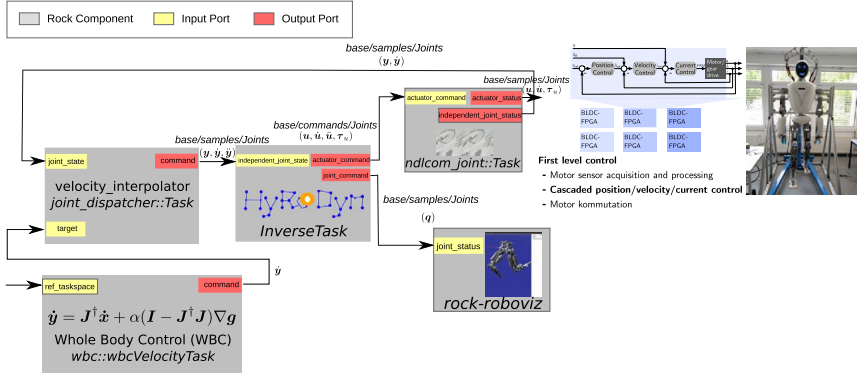


Figure 18.14 Application of HyRoDyn orogen component in the whole-body control of RH5 humanoid [1].



Figure 18.15 Animation of up-down movement with RH5 leg [1].

Fig. 18.17 shows the actuator position, velocity, acceleration and torque profile of the robot. Further, the full spanning tree state (q, \dot{q}, \ddot{q}) is computed during this process, which can be used for robot visualization (see Fig. 18.15).

Example 18.4. Feed-forward control scheme consists of feed-forward of the nonlinear dynamic model of robot and a linear servo feedback. It is the simplest form of the non-linear controller which can be employed for motion control of a robot exploiting its dynamic model. The control law can be formulated by the following equation:

$$\tau = \tau_u + K_p(u^{ref} - u) + K_d(\dot{u}^{ref} - \dot{u}), \quad (18.5)$$

where τ_u is the vector of actuator forces computed by inverse dynamic model, $K_p = \text{diag}(K_p^1, K_p^2, \dots, K_p^p)$ and $K_d = \text{diag}(K_d^1, K_d^2, \dots, K_d^p)$ are the diagonal matrices for proportional and derivative gains respectively. The inverse task (Fig. 18.11a) and controller task (Fig. 18.11d) in HyRoDyn can be connected to achieve feed-forward motion control of the RH5 humanoid leg. The input motion is specified in the independent joint space

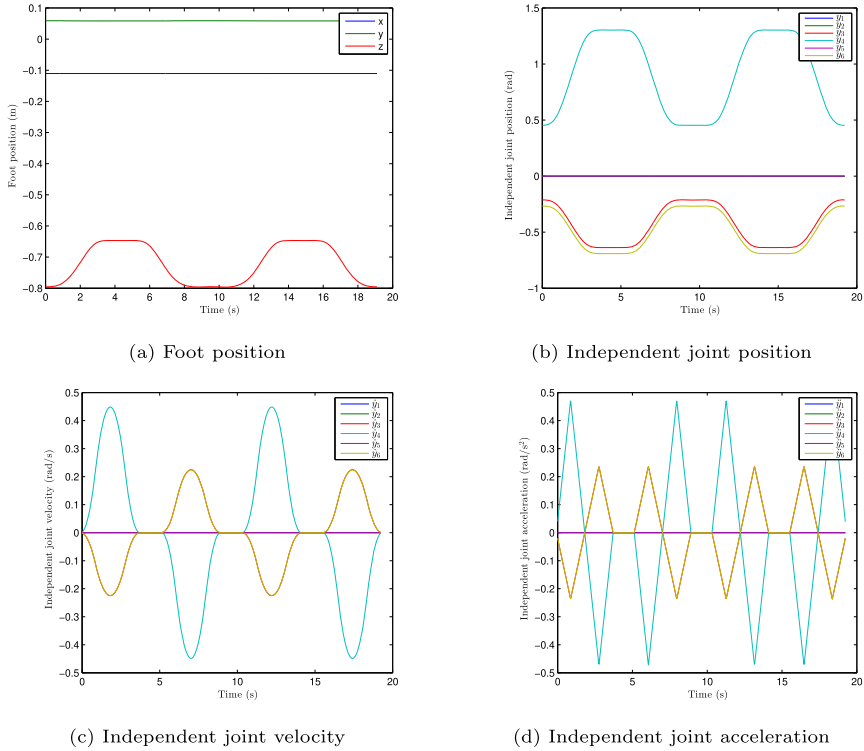


Figure 18.16 Task space and independent joint space trajectories of RH5 leg for up-down movement [1].

using an position interpolator between two desired set points. For the sake of simplicity, only one joint, i.e., LLHip3 is moved from a position of -0.91 rad to -0.22 rad. The interpolator produces a smooth reference trajectory $(\mathbf{y}_3^{ref}, \dot{\mathbf{y}}_3^{ref}, \ddot{\mathbf{y}}_3^{ref})$ between these two set points back and forth. Inverse kinematics and dynamics problems are solved using HyRoDyn and the reference commands for the actuators $(\mathbf{u}^{ref}, \dot{\mathbf{u}}^{ref}, \boldsymbol{\tau}_u^{ref})$ are generated. These actuator commands are then fed to the feed-forward controller implemented on the FPGA of the BLDC joints in the humanoid platform. This controller utilizes the commanded actuator forces as a feed-forward term and commanded actuator position and velocity (along with the measured position and velocity, i.e., \mathbf{u} & $\dot{\mathbf{u}}$) for computing the linear servo feedback. Fig. 18.18 shows the trajectory tracking in both actuation and independent joint space. In particular, Fig. 18.18a shows the motion tracking in the independent joint space, i.e., it compares the reference position val-

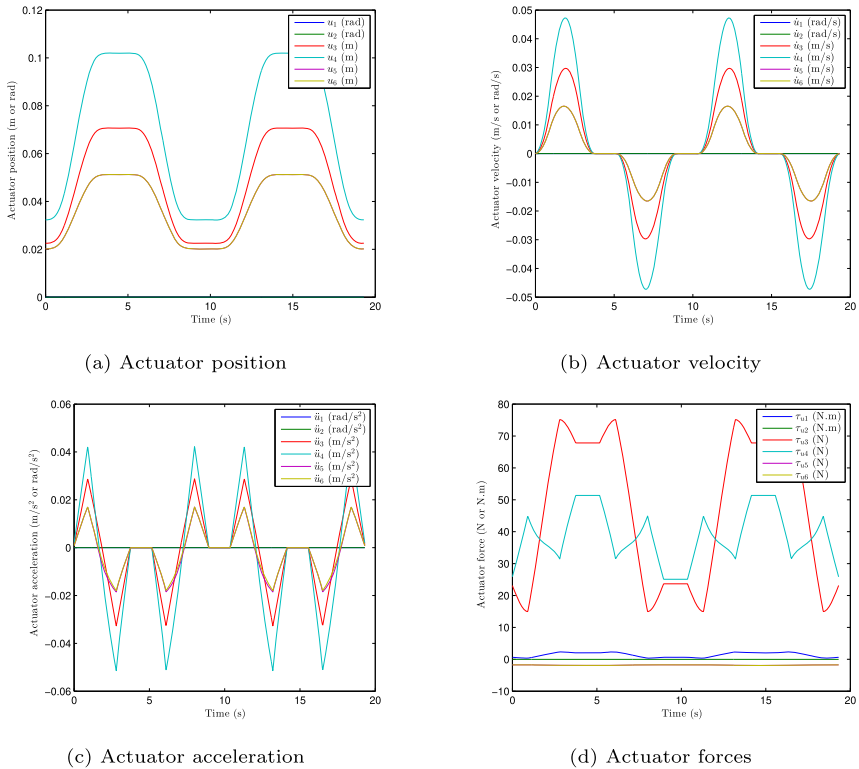


Figure 18.17 Actuator state of RH5 leg for up-down movement [1].

ues (\mathbf{y}_3^{ref}) against absolute position encoder measurements in the concerned joint (\mathbf{y}_3). Similarly, Fig. 18.18b and Fig. 18.18c show the actuator position and velocity tracking. Finally, Fig. 18.18d compares the commanded actuator force in LLHip3 and LLKnee joints against the direct force measurements available from the actuators.

18.6 Automated robot analysis using HyRoDynPy

A Python based interface to HyRoDyn library, called HyRoDynPy, has also been developed to make it easy for the users to use HyRoDyn software with the user-friendly programming language Python. When a new robot is developed, various different types of analyses need to be performed. This includes the analyzes of configuration space, independent joint space, actuator space, work space, etc. The quality of velocity or force transmission of

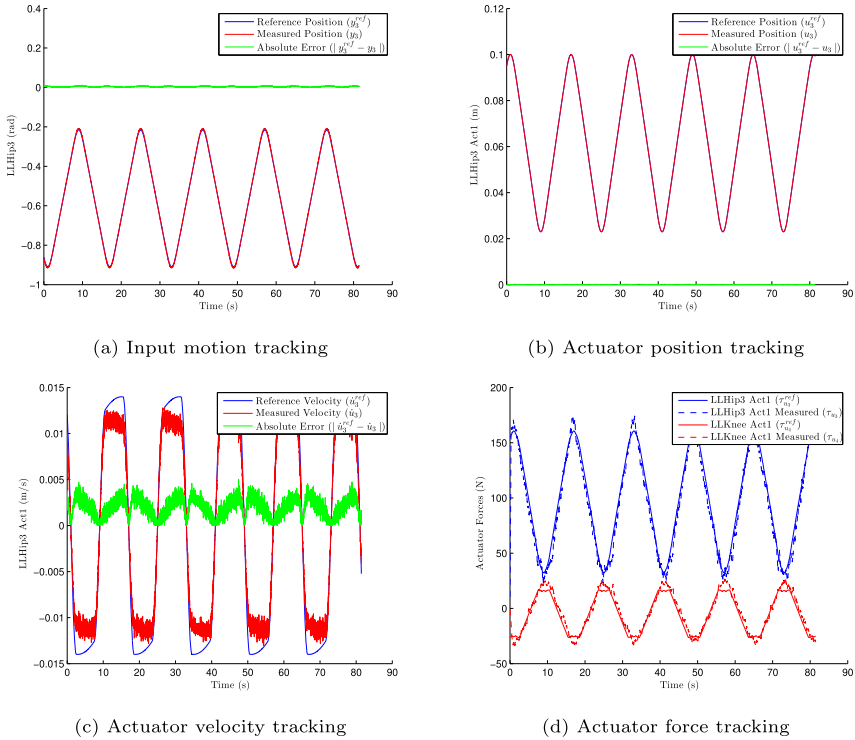


Figure 18.18 Feed-forward control of RH5 humanoid using HyRoDyn [1].

a robot can be measured by plotting the inverse of condition number of the kinematic Jacobian matrix (\mathbf{J}) over the robot's workspace. The inverse of condition number of the Jacobian is calculated with $c(\mathbf{J}) = \frac{1}{\|\mathbf{J}\| \|\mathbf{J}^{-1}\|}$ where $\|\cdot\|$ represents the Euclidean norm of the matrix. Its value is bounded between 0 and 1, which signify the worst and best conditioning. Fig. 18.19 shows the plot of inverse of condition number of Jacobian over the workspace of RH5 leg and Recupera right arm [17,18] systems. It is also possible to compute various slices of configuration space and equip them with such quality measures. This is very relevant for the analysis of various parallel submechanism modules in a robot. Further, the end user might be interested in checking the payload capacity of the robot at various points in the robot workspace. It is also possible to swing all the joints of the robot between their minimum and maximum joint limits in either choice of coordinates (independent joint space, actuation space or task space) using this tool which gives the end user a quick impression in how the robot

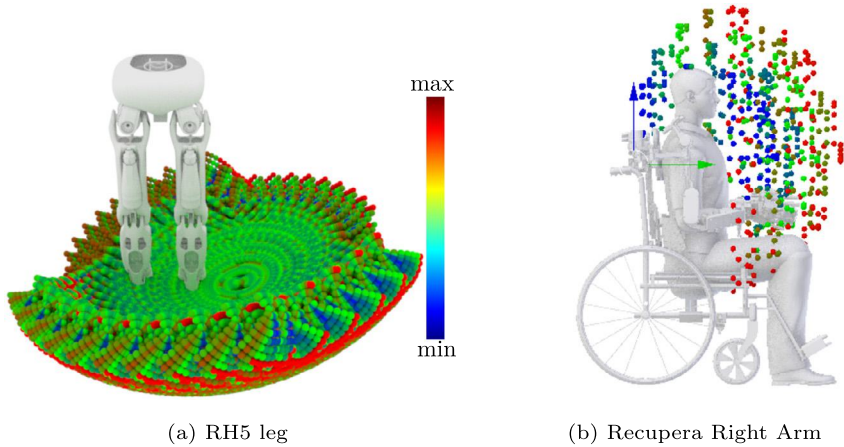


Figure 18.19 Workspace quality measure using HyRoDyn [1].

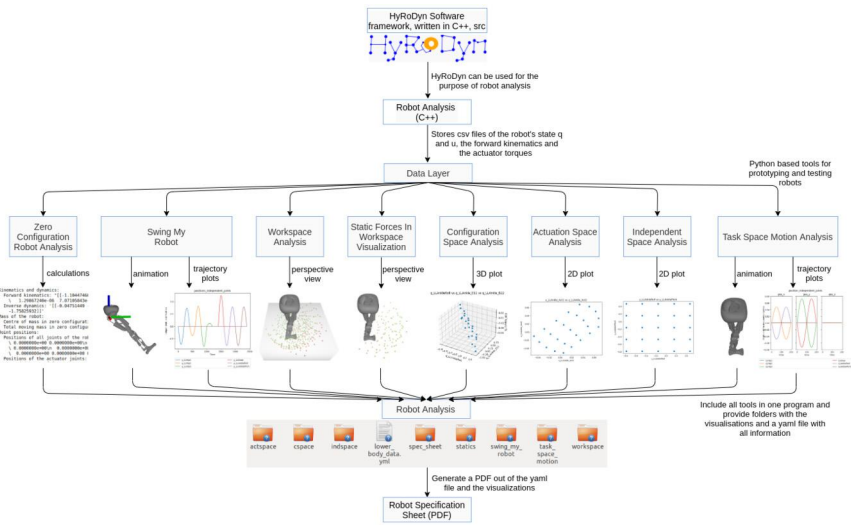


Figure 18.20 Automated Robot Analysis using HyRoDynPy.

can move in different ways. An overview of different robot analysis tools is provided in Fig. 18.20. Additionally, with a single command line tool, all these different analyzes can be performed and the relevant information is written to a human readable PDF file which makes it easy for robot designers to analyze their robot design without having any background in programming.

18.7 Conclusion

This chapter presents the architecture of HyRoDyn software framework for the simulation and control of the series-parallel hybrid robots. Different aspects of this framework, such as robot description format, visual editing of the robot models, HyRoDyn software library and its integration in RoCK middleware are discussed. Lastly, automated robot analysis using the Python interface of HyRoDyn, called HyRoDynPy, was discussed. Overall, this chapter demonstrates the usability of HyRoDyn software for different types of robotics engineers or researchers.

References

- [1] S. Kumar, Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots, Ph.D. thesis, Universität Bremen, 2019.
- [2] R. Kumar, A hybrid numerical and analytical approach towards resolving loop closure constraints in rigid body dynamics, <https://doi.org/10.13140/RG.2.2.14004.99207>, Aug 2021.
- [3] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Mechatronics* 68 (2020) 102367.
- [4] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, *Journal of Intelligent & Robotic Systems* 94 (2019) 303–325, <https://doi.org/10.1007/s10846-018-0792-x>.
- [5] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: J. Lenarcic, V. Parenti-Castelli (Eds.), *Advances in Robot Kinematics 2018*, Springer International Publishing, Cham, 2019, pp. 431–439.
- [6] S. Kumar, A. Mueller, An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots, in: Vol. 5A: 43rd Mechanisms and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2019, v05AT07A054, <https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2019/59230/V05AT07A054/6453674/v05at07a054-detc2019-97115.pdf>, <https://doi.org/10.1115/DETC2019-97115>.
- [7] R. Kumar, S. Kumar, A. Müller, F. Kirchner, Modular and hybrid numerical-analytical approach – a case study on improving computational efficiency for series-parallel hybrid robots, in: 2022 IEEE/R SJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 3476–3483, <https://doi.org/10.1109/IROS47612.2022.9981474>.
- [8] D-RoCK, Models, methods and tools for the model based software development of robots, <https://robotik.dfki-bremen.de/en/research/projects/d-rock.html>, 2018. (Accessed 5 October 2018).
- [9] J. Esser, S. Kumar, H. Peters, V. Bargsten, J. de Gea Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid robot, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), 2021, pp. 400–407, <https://doi.org/10.1109/HUMANOIDS47582.2021.9555770>.

- [10] K.A. von Szadkowski, M. Langosz, Phobos: 3D robot modelling made easy, Tech. Rep. 1406, DFKI, September 2015.
- [11] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 01–07, <https://doi.org/10.1109/ICRA46639.2022.9811843>.
- [12] R. Featherstone, Rigid Body Dynamics Algorithm, Springer, 2008.
- [13] M.L. Felis, RBDL: an efficient rigid-body dynamics library using recursive algorithms, *Autonomous Robots* 41 (2) (2017) 495–511.
- [14] G. Guennebaud, B. Jacob, et al., Eigen v3, <http://eigen.tuxfamily.org>, 2010.
- [15] S. Joyeux, Rock: the robot construction kit, <https://www.rock-robotics.org/stable/>, 2010. (Accessed 5 October 2018).
- [16] S. Kumar, M. Simnofske, B. Bongardt, A. Müller, F. Kirchner, Integrating mimic joints into dynamics algorithms: exemplified by the hybrid Recupera exoskeleton, in: Proceedings of the Advances in Robotics, AIR '17, ACM, New York, NY, USA, 2017, pp. 27:1–27:6, <https://doi.org/10.1145/3132446.3134891>, <http://doi.acm.org/10.1145/3132446.3134891>.
- [17] S. Kumar, M. Simnofske, B. Bongardt, A. Mueller, F. Kirchner, Integrating mimic joints into dynamics algorithms – exemplified by the hybrid Recupera exoskeleton, in: Advances in Robotics (AIR-2017), June 28–July 2, New Delhi, India, ACM-ICPS, 2017.
- [18] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the Recupera exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019), <https://doi.org/10.3390/app9040626>, <http://www.mdpi.com/2076-3417/9/4/626>.

This page intentionally left blank

CHAPTER 19

Design of a flexible bio-inspired robot for inspection of pipelines

Swaminath Venkateswaran^a and Damien Chablat^b

^aLéonard de Vinci Pôle Universitaire, Research Center, 92 916 Paris La Défense, Paris, France

^bCentre National de la Recherche Scientifique (CNRS), Laboratoire des Sciences du Numérique de Nantes (LS2N), UMR CNRS 6004, Nantes, France

19.1 Introduction

Piping inspection robots have created a major attraction in the scientific research community. These robots can carry out an inspection or an intervention in areas that are difficult for humans to access. They also reduce the chances of hazards or risks to human life as these robots can potentially work within an irradiated or polluted environment. With the automation of control systems, inspection robots can accomplish the desired task within a short period as well as with better accuracy. Generally, the locomotion principles of these robots can employ either mechanical systems or inspire their motion from nature, otherwise referred to as Bio-inspired techniques. Kasim et al. [1] proposed a distinction between these two categories, where the mechanical systems can use wheels and pulleys [2], telescopic systems [3], impact modules [4], or natural peristalsis [5] to accomplish the locomotion. On the other hand, the bio-inspired systems can mimic their locomotions from animals such as earthworms [6], snakes [7], millipedes [8], lizards [9], or an octopus [10]. Currently, several piping inspection robots have been designed and developed by many researchers. These robots exploit either the mechanical [11] or bio-inspired approach [12,13] for the locomotion. However, the design and development of an inspection robot for pipeline diameters less than 150 mm remains to be a big challenge. Park et al. [14] proposed different kinds of obstacles that a piping inspection can generally encounter and it is shown below in Fig. 19.1.

This chapter will focus on arriving at the design of a piping inspection robot that can operate in a pipeline having a diameter of 150 mm or less and at the same time, a design which is capable of overcoming bends or intersections (Figs. 19.1b to 19.1d). Also, the design will be capable of adapting its structure when it encounters a diameter change inside a

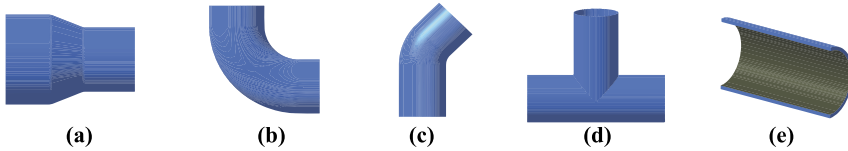


Figure 19.1 Possible profiles encountered by an inspection robot inside an industrial pipeline: (a) varying diameter; (b) curvature; (c) inclination; (d) branching; and (e) uneven surface walls.

pipeline (Fig. 19.1a). The chapter is organized as follows. Section 19.2 focuses on the initial design of a rigid caterpillar-type piping inspection robot. Section 19.3 focuses on the design and prototyping of a tensegrity mechanism that will be incorporated onto the robot studied in Section 19.2. Section 19.4 deals with the identification of optimal design parameters of the flexible robot in order to work inside straight or curved pipe profiles. Subsequently, the static modeling of the robot assembly inside a horizontal pipeline is proposed in Section 19.5. The chapter then ends with conclusions and future perspectives.

19.2 Rigid bio-inspired piping inspection robot—an overview

As a part of a project with AREVA, a novel bio-inspired piping inspection robot that mimics the locomotion of a caterpillar was proposed and developed by Henry et al. [15] and Chablat et al. [16]. This robot is capable of addressing the issue of the design of inspection robots for pipelines having diameters less than 150 mm. Using electrical actuators with leg mechanisms and spindle drive units, this robot accomplishes the motion of a caterpillar and can work inside 40–94 mm diameter pipelines. Three architectures were studied for optimization viz: slot-follower, four-bar, and six-bar mechanisms in [15] for the choice of leg mechanism for this robot. By following a heuristic optimization [17] technique in MATLAB[®], the slot-follower mechanism was chosen for the robot [15]. A detailed force analysis was then performed on the robot during clamping and locomotion conditions. The static force model was analyzed using Coulomb's law during the clamping phases of the leg modules with the pipeline walls [16]. For the dynamic force model, the forces on the central actuator were identified under locomotion using the Newton–Euler recursive algorithm [18]. Upon comparison, it was observed that the static forces were higher than

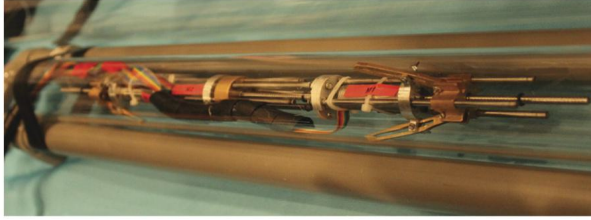


Figure 19.2 Prototype of the caterpillar-type piping inspection robot developed at LS2N, France.

the dynamic forces. However, in the numerical analysis, it was not possible to validate electromechanical factors and frictional coefficients. Using DC-motors, spindle drives, leg mechanisms, and standard fasteners, the entire robot was realized at LS2N, France. Experiments were then carried out on this prototype, where the peak results of the static force models were set as threshold limits for ensuring tight clamping with the pipeline walls. The real-time forces induced on the actuators under locomotion were then interpreted using a Force-control algorithm with the help of a BeagleBone (BB) black micro-computer inside a 74 mm diameter straight pipe. The robot was able to overcome the issue of working inside pipelines having diameters less than 150 mm that can have variable diameters. One major disadvantage of this prototype is that it is a rigid model and it cannot be employed for curved pipe profiles such as represented in Figs. 19.1b to 19.1d. The spindle drive unit used in each module of the robot was also currently oversized, as it has a screw length of 102 mm. This excess screw length restricted the robot to overcome a pipeline having bends or curvatures if the robot was made flexible. Thus, in the following section, a solution will be proposed which can assist this prototype to work inside curved pipe profiles. The prototype of the caterpillar-type rigid piping inspection robot is shown in Fig. 19.2.

19.3 Design, analysis, and synthesis of a tensegrity mechanism

In order to overcome pipe bends and junctions, the design of the caterpillar-type inspection robot discussed in the previous section was modified. An articulation unit can be incorporated between the modules to overcome this issue. Generally, articulation units can be classified into active and passive systems. In the literature, there exist several robots

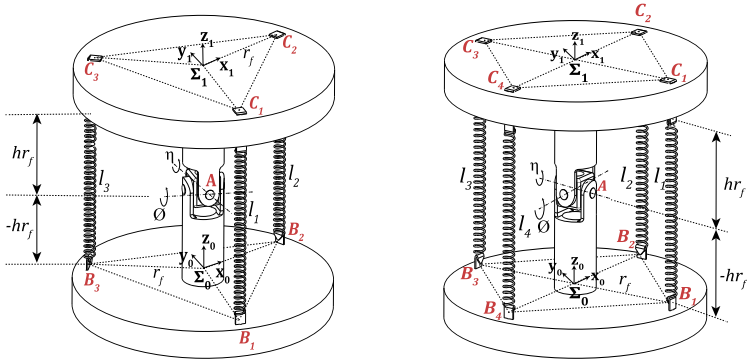


Figure 19.3 The 3-SPS-U (left) and 4-SPS-U (right) tensegrity mechanisms at their home-poses.

that incorporate one of these solutions, such as the robot THES-I [19], which proposed an actuated universal joint, and the robot of Kwon et al. [13], which employed a passive compression spring. Most of these robots employed articulation units that are either passive or active, but not a combination of both. Two types of tensegrity mechanisms that employ a passive universal joint and either three or four tension springs were proposed as articulation units for the rigid piping inspection robot. While encountering a pipe bend at 90°, both mechanisms can work in a passive mode. In the event of a junction or a T-union, cables that pass through the springs of each mechanism are actuated to tilt in a certain direction. These mechanisms were designed taking into account three parameters, namely [20]: Passive compliance (autonomous mode), active compliance (actuation), and tilt limits.

19.3.1 Identification of optimal design parameters

The tensegrity mechanisms were analyzed by correlating it to parallel manipulators of type 3-SPS-U and 4-SPS-U. Here S indicates the spherical joint, U indicates the universal joint, and P indicates the actuated prismatic joints or the springs. The representation of the tensegrity mechanisms at their home poses and the correlation to a parallel manipulator is shown in Fig. 19.3.

The fixed coordinate frame of the base is represented by Σ_0 with the origin at B_0 . The spring mounting points on the fixed base are represented by B_i ($i = 1, 2, 3$) and form the imaginary equilateral triangle of the manipulator base, whose median is r_f for the 3-SPS-U mechanism. The

base mounting points B_i ($i = 1, 2, 3, 4$) forms an imaginary square for the 4-SPS-U mechanism. The diagonal length for the square is $2r_f$. The vector coordinates for the base mounting points are given by:

$$\mathbf{b}_i = [r_f \cos\left(\frac{i2\pi}{j}\right), r_f \sin\left(\frac{i2\pi}{j}\right), -r_f h]^T, \quad (19.1)$$

where $j = 3$, $i = 0$ to 2 for 3-SPS-U and $j = 4$, $i = 0$ to 3 for 4-SPS-U.

The moving coordinate frame of the end-effector is represented by \sum_1 with its origin at C_0 . The spring mounting points of the end-effector is represented by C_i ($i = 1, 2, 3$) and C_i ($i = 1, 2, 3, 4$) for the 3-SPS-U and 4-SPS-U mechanism respectively. To estimate the vector coordinates of the end-effector mounting points, the XY Euler angle matrix is employed with respect to the central point A of the universal joint. The vector coordinates for the end-effector mounting points are given by:

$$\mathbf{c}_i = \mathbf{E} [r_f \cos\left(\frac{i2\pi}{j}\right), r_f \sin\left(\frac{i2\pi}{j}\right), r_f h]^T \quad \text{with } \mathbf{E} = \mathbf{R}_x(\eta)\mathbf{R}_y(\phi), \quad (19.2)$$

where $j = 3$, $i = 0$ to 2 for 3-SPS-U and $j = 4$, $i = 0$ to 3 for 4-SPS-U.

In Eq. (19.2), $\mathbf{E} \in SE(3)$ represents the spatial transformation matrix obtained from the Euler angles of the universal joint, and it is used to identify the end-effector coordinates as indicated in Eq. (19.2). The inverse kinematic problem (IKP) for the mechanism is simpler and it determines the length of springs between the base and end-effector at home pose and working conditions. The equation is given by:

$$l_i = \sqrt{(b_{ix} - c_{ix})^2 + (b_{iy} - c_{iy})^2 + (b_{iz} - c_{iz})^2}, \quad (19.3)$$

with $i = 1$ to 3 for 3-SPS-U & $i = 1$ to 4 for 4-SPS-U.

In line with the flange units employed in the piping inspection robot, the value of r_f was taken as 11 mm. However, the value of parameter h plays an essential role in the stability of the mechanism under static modes. According to Lagrange, for a moving system, the equation of motion [21] is given by:

$$\boldsymbol{\tau} = \frac{d}{dt} \left(\frac{\partial T}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial T}{\partial \mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}}, \quad \text{where } \mathbf{q} = [\eta, \phi]^T. \quad (19.4)$$

Under static conditions, the total potential energy of the tensegrity mechanisms is given by:

$$U_{c1} = \sum_{i=1}^3 F_i l_i \quad U_{s1} = \sum_{i=1}^3 \frac{1}{2} k (l_i - l_0)^2 \quad \text{for 3-SPS-U,} \quad (19.5)$$

$$U_{c2} = \sum_{i=1}^4 F_i l_i \quad U_{s2} = \sum_{i=1}^4 \frac{1}{2} k (l_i - l_0)^2 \quad \text{for 4-SPS-U,} \quad (19.6)$$

$$U_{3\text{-SPS-U}} = U_{c1} + U_{s1}, \quad U_{4\text{-SPS-U}} = U_{c2} + U_{s2}. \quad (19.7)$$

In Eqs. (19.5) and (19.6), U_c and U_s represents the potential energy contributed by the cables and the springs. The applied forces along the cables were considered as F_1 to F_3 and F_1 to F_4 for the 3-SPS-U and 4-SPS-U mechanisms respectively. The total potential energy for both mechanisms is given by $U_{3\text{-SPS-U}}$ and $U_{4\text{-SPS-U}}$ in Eq. (19.7). For the stability analysis, the springs were considered massless and their free lengths l_0 were set to 0 mm for the analysis. A meta-heuristic optimization approach using genetic algorithm in MATLAB was followed to identify the parameter h and spring stiffness k for the two tensegrity mechanisms at their home poses where the tilt angles are zero. As the mechanism depends on the two pose variables η and ϕ , it was necessary to estimate the determinant of the Hessian matrix and the value of the second-order derivative of the potential energy with respect to one of the tilt angles. For a stable design configuration, both parameters must be greater than zero. As the tensegrity mechanism is proposed to be integrated with the piping inspection robot, a preloading of around 6 N exists due to the robot weight. Thus, a preloading of 2 N and 1.5 N along each spring for the 3-SPS-U and 4-SPS-U mechanisms were considered for the analysis. The optimization problem is stated as:

$$\text{Maximize: } f_{11}(\mathbf{x})$$

$$\text{subject to constraints: } g_1 : \det(\mathbf{H}) \geq 0, \quad g_2 : f_{11}(\mathbf{x}) \geq 0, \quad \text{where } \mathbf{x} = [h, k]^T.$$

In the above definition, \mathbf{H} is the Hessian matrix and $f_{11}(\mathbf{x})$ is the second-order derivative of the potential energy with respect to one of the tilt angles.

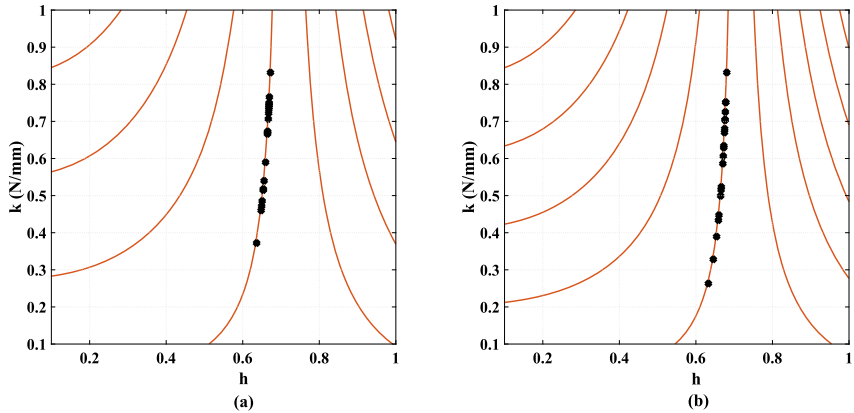


Figure 19.4 Solutions for optimization problem obtained by Genetic Algorithm for the (a) 3-SPS-U and (b) 4-SPS-U mechanisms.

Their equations are of the form:

$$\mathbf{H}_i = \begin{bmatrix} \frac{\partial^2 U_j}{\partial \eta^2} & \frac{\partial^2 U_j}{\partial \eta \partial \phi} \\ \frac{\partial^2 U_j}{\partial \eta \partial \phi} & \frac{\partial^2 U_j}{\partial \phi^2} \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}, \quad \det(\mathbf{H}_i) = f_{11}f_{22} - f_{12}^2, \quad (19.8)$$

where for $i = [1, 2]$, $j = [3\text{-SPS-U}, 4\text{-SPS-U}]$.

The objective function aims to maximize the derivative f_{11} such that it remains positive along with the determinant of the Hessian matrix throughout the optimization process. The results of solutions obtained from the optimization problem are represented in Fig. 19.4.

The solutions obtained by the genetic algorithm in MATLAB are depicted as black-colored scatter points over the contour of the objective function in Fig. 19.4. It could also be observed that optimum values of h are distributed between 0.6 and 0.7. For each value of h , an optimum value for spring stiffness k was obtained. Based on a spring stiffness of 0.75 N/mm that was available at LS2N, the corresponding value of h was chosen from the results, which were around 0.649 and 0.663 for each mechanism. For ease of calculations and prototyping, a common value of $h = 0.6$ was taken for both mechanisms. The stability plot for both mechanisms with the optimized design parameters under the presence of a preload is represented below in Fig. 19.5.

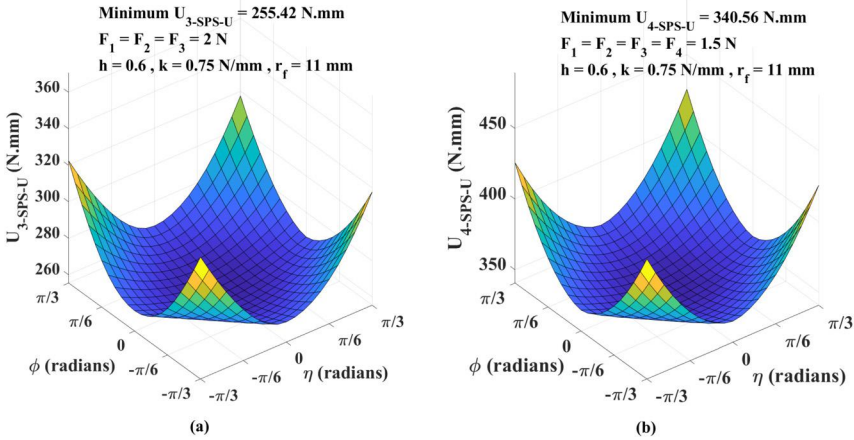


Figure 19.5 Plot of total potential energy versus the tilt angles η and ϕ for $h = 0.6$ at the home-pose condition for the (a) 3-SPS-U and (b) 4-SPS-U mechanisms.

From Fig. 19.5, a stable configuration could be observed for both mechanisms under the presence of a preload. These design parameters will be useful to analyze the singularity and tilt limits of both architectures.

19.3.2 Singularity and workspace analysis

With the determination of stable design parameters, the next step focuses on the analysis of singularities, which provides information on the tilt limits of both architectures. This analysis also enables us to identify the best architecture that can be accommodated for the piping inspection robot. For a parallel manipulator, the singularity equation is given by the well-known equation [22]:

$$\mathbf{A}\mathbf{t} + \mathbf{B}\dot{\boldsymbol{\rho}} = 0, \tag{19.9}$$

where \mathbf{t} represents the angular velocity vector

and $\dot{\boldsymbol{\rho}} = [\dot{l}_1, \dot{l}_2, \dot{l}_3]^T$ represents the joint velocity vector.

Parallel singularities occurs when the determinant of the direct kinematics matrix \mathbf{A} of Eq. (19.9) vanishes. This phenomenon can be observed when the end-effector platform aligns itself with one of the springs of the mechanism. There exist no serial singularities for the mechanism as the determinant of the inverse kinematics matrix \mathbf{B} does not vanish as the length

of the prismatic springs cannot be zero. Since the matrix \mathbf{A} does not correspond to an $n \times n$ square matrix, parallel singularities were analyzed by subdividing the manipulator into three and four sets of 2-SPS-U architectures for the 3-SPS-U and 4-SPS-U mechanisms [23]. By using the Groebner base elimination technique, the joint limits equations that provided solutions to the IKP were generated using the SIROPA library in Maple [24]. The singularity equations were generated for both mechanisms to determine their workspace limits [20]. The joint limits and singularity equations were then employed to determine the singularity-free workspace for the mechanisms. By using the cylindrical algebraic decomposition (CAD) technique, the real solutions were generated for the problem [25] using SIROPA library in Maple. For isolating the aspects around home-pose, the singularity and joint limits equations were transformed as inequalities [26] in Maple. For generating the traces of workspace and singularities, the joint limits were set as $[l_{min}, l_{max}] = [5, 30]$ mm for both mechanisms. The lower limit l_{min} contributes mainly to the boundaries of the aspects centered at $\eta = \phi = 0$ radians.

19.3.2.1 Analysis of the 3-SPS-U architecture

The workspace and singularity curves for the 3-SPS-U mechanism are represented below in Fig. 19.6a.

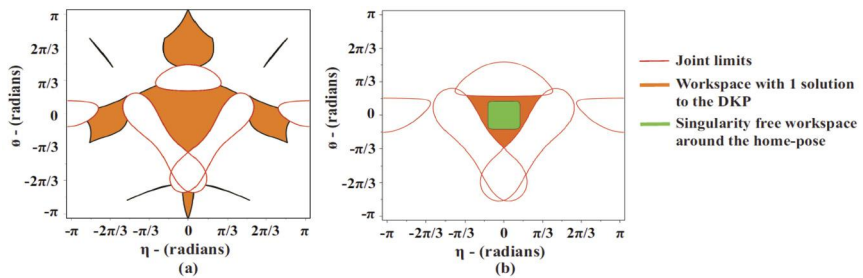


Figure 19.6 Representation of the (a) Feasible workspace with joint limits at $l_{min} = 5$ mm and (b) singularity-free workspace with joint limits at $l_{min} = 6.5$ mm for the 3-SPS-U mechanism.

The orange regions indicate the workspace for the mechanism with singularity boundaries. The joint limits for the three springs are represented by red lines and they appear superimposed on the plots. Parallel singularities can be observed for the mechanisms at the corners of the orange regions, especially at $\eta = \phi = 0$ radians. The singularity-free workspace for

the mechanism is bounded by a triangular region which has to be extracted to determine the minimum limit for the springs. One edge of triangle is extracted at $[\eta, \phi] = [0, -\pi/3]$ radians. At this position, one or two of the springs reach their minimum position with no singularities. A value of 6.5 mm is estimated for the lengths l_2 and l_3 at this position. This is the minimum limit to which the spring can go to avoid parallel singularities. For determining the remaining edges of the triangle, the minimum limits for the other length pairs ($l_1 - l_2$ & $l_1 - l_3$) are fixed as 13.5 mm. The values of η are taken as $\pm\pi/3$ radians from the workspace obtained in Fig. 19.6a. A value of 0.67 radians is found for ϕ at these positions. With the modified lower limit for the springs, the workspace for the 3-SPS-U tensegrity mechanism is recalculated and plotted as shown in Fig. 19.6b. It could be seen that a singularity-free workspace in the form of a Reuleaux triangle is obtained. The minimum square zone within this triangle is bounded by the limits $[\eta, \phi] \in [-2\pi/15, 2\pi/15]$ radians. Thus, in order to avoid singularities during operating conditions, the 3-SPS-U mechanism can tilt up to $\pm 2\pi/15$ radians.

19.3.2.2 Analysis of the 4-SPS-U architecture

The workspace and singularity surfaces were generated for the 4-SPS-U architecture and are shown below in Fig. 19.7a.

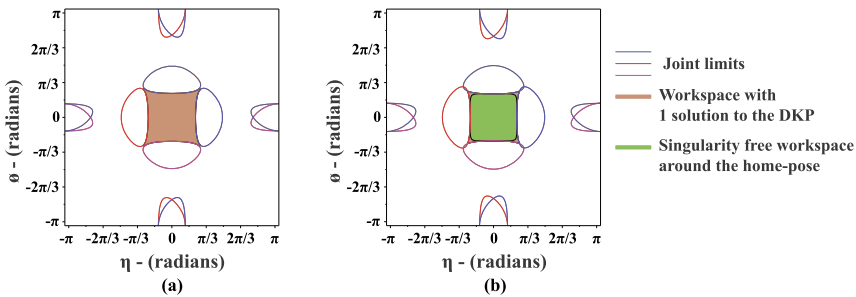


Figure 19.7 Representation of the (a) Feasible workspace with joint limits at $l_{min} = 5$ mm and (b) singularity-free workspace with joint limits at $l_{min} = 5.3$ mm for the 4-SPS-U mechanism.

In Fig. 19.7a, the joint limits are represented by the colored lines, and the workspace is represented by the brown-colored region. Compared to the 3-SPS-U architecture, a wider workspace with singularity regions is observed for the 4-SPS-U mechanism. Based on the analysis, parallel

singularities occur when both η and $\phi \in [\pm\pi/3, \pm\pi/3]$ radians. At these boundaries, two of the four legs reach their minimum length. At η equal to 0 radians, ϕ attains maximum tilt values of $\pm\pi/4$ radians. Using the joint limits and the minimum spring length value at the singularity boundary, the maximum tilt limits with singularity-free workspace were estimated for $[\eta, \phi]$ at $[0, \pi/4]$ radians. A tilt limit of $\pm 5\pi/18$ radians was obtained and the minimum spring length was estimated to be 5.3 mm. The recalculated workspace was in the form of a square for the 4-SPS-U mechanism and it is represented in Fig. 19.7b. The singularity-free workspace for this mechanism is superimposed on this square, where the tilt limits are given by $\pm 5\pi/18$ radians.

19.3.2.3 Discussions

From the workspace analysis, it could be observed that the 4-SPS-U mechanism generates a higher tilt limit when compared to the 3-SPS-U mechanism. The articulation unit for the bio-inspired robot must be able to overcome pipe bends at $\pi/2$ radians in a passive mode. With the 3-SPS-U mechanism, it will be difficult to address this problem as there are possibilities that the mechanism might reach singular poses within a narrow tilt limit range. On the other hand, the 4-SPS-U mechanism offers higher tilt limits with a singularity-free workspace. The possibilities of reaching singular positions by this mechanism are comparatively less than the 3-SPS-U mechanism. The tilt limit issues for the 3-SPS-U mechanism can be addressed by using stacked modules. Based on the results of the algorithm, it can be concluded that the 4-SPS-U architecture is suitable for addressing passive and active compliance issues. The narrow tilt range of 3-SPS-U architecture restricts its application for the issue of active compliance.

19.3.3 Prototyping and control of the tensegrity mechanism

In order to demonstrate the validation of the issue of active compliance, a prototype of the 3-SPS-U tensegrity mechanism was realized at LS2N using springs (stiffness of 0.75 N/mm), universal joint and ABS platforms. A scaled-up model was constructed with $r_f = 56.7$ mm. As the mechanism is similar to a parallel robot, using the IKP solutions, the Jacobian matrices were extracted from the direct-kinematics matrices. Using a BB black microcomputer, a closed-loop PID controller law was developed to tilt the mechanism. DC-motors coupled with a high-reduction gear trains were employed to have a static model. Encoders were coupled to these motors to transmit angular position data to the user. A circular trajectory was

demonstrated within the singularity-free workspace zone of the tensegrity mechanism (Fig. 19.6b).

19.3.3.1 Trajectory planning and control

For the experiments, the IKP was solved, where the tilt angles η and ϕ were passed as inputs to the control loop. The pose variables and the articular variables for the mechanism are given by $\mathbf{q} = [\eta, \phi]$ and $\rho = [l_1, l_2, l_3]$ [23]. The simpler form of the IKP is provided in Eq. (19.3). The desired angular positions of the mechanism (θ_{di}) can be calculated with respect to the IKP and the measured angular positions can be calculated as a function of the motor parameters (encoder data E , gear reduction ratio G , and counts per revolution C). The corresponding equations are given by:

$$\theta_{di} = \frac{(l_i - l_{home})}{r}, \quad \theta_{mi} = \frac{E_i \pi}{2CG}, \quad \text{with } i = 1, 2, 3. \quad (19.10)$$

In Eq. (19.10), the value of l_{home} is 68 mm for $r_f = 56.7$ mm and $h = 0.6$. During the change of tilt angle from home-pose to the desired position, the Cartesian velocities of the prismatic joints can be calculated using the equation [27]:

$$\begin{bmatrix} \dot{l}_1 \\ \dot{l}_2 \\ \dot{l}_3 \end{bmatrix} = \mathbf{J}_c \begin{bmatrix} \dot{\eta} \\ \dot{\phi} \end{bmatrix}, \quad \text{with } \mathbf{J}_c = \begin{bmatrix} \frac{\partial l_1}{\partial \eta} & \frac{\partial l_1}{\partial \phi} \\ \frac{\partial l_2}{\partial \eta} & \frac{\partial l_2}{\partial \phi} \\ \frac{\partial l_3}{\partial \eta} & \frac{\partial l_3}{\partial \phi} \end{bmatrix}. \quad (19.11)$$

In Eq. (19.11), \mathbf{J}_c represents the direct kinematics matrix (\mathbf{A}) of the 3-SPS-U tensegrity mechanism. For estimation of intermediate positions from the home-pose to the final pose, the fifth-degree polynomial equation of Khalil [28] is employed for displacement, velocity, and acceleration profiles. The values of tilt angles to perform a circular trajectory are given by:

$$\mathbf{P}(t) = \begin{bmatrix} \eta \\ \phi \end{bmatrix} = \begin{bmatrix} R \sin(r_0 + (r_1 - r_0)s(t)) \\ R \cos(r_0 + (r_1 - r_0)s(t)) \end{bmatrix}. \quad (19.12)$$

In Eq. (19.12), R corresponds to the radius of the circular trajectory, which is equal to the tilt angle chosen within the singularity-free workspace. The values r_0 and r_1 maps the initial and final points of circle and it goes from 0 to -2π to perform a counter-clockwise (CCW) circular trajectory. For the control of the tensegrity mechanism, a force control algorithm was

employed to reach the desired position for input tilt angles. Through the application of motor torques and current, the mechanism was made to tilt to reach the desired position. A closed-loop feedback system was employed, which tries to minimize the errors between the desired and measured position. The classical relation for the PID control scheme, which provides the motor torque Γ for the experimental setup, is given by the equation:

$$\Gamma = J(\ddot{\theta}_i + K_P(\theta_{di} - \theta_{mi}) + K_D(\dot{\theta}_{di} - \dot{\theta}_{mi}) + K_I \int_0^t (\theta_{di} - \theta_{mi})), \quad i = 1, 2, 3. \tag{19.13}$$

In Eq. (19.13), Γ is the regulated torque obtained at the gearbox output shaft after PID correction. For computing the torques induced on each motor, Eq. (19.13) was multiplied by the gear reduction ratio G of the planetary gearbox. The inertia of motor-gearbox assembly J was taken as $4.1e - 7 \text{ kg.m}^2$ from the catalog of motor. The values for the PID terms of Eq. (19.13) are calculated using the motor dielectric constants and resistance. The constants are given by:

$$\omega = \frac{k_t k_e}{R J} = 14 \text{ rad/s}, \quad K_P = 3\omega^2 = 588, \quad K_D = 3\omega = 42, \quad K_I = \omega^3 = 2744. \tag{19.14}$$

An overview of the experimental setup and the PID controller block is shown below in Fig. 19.8.

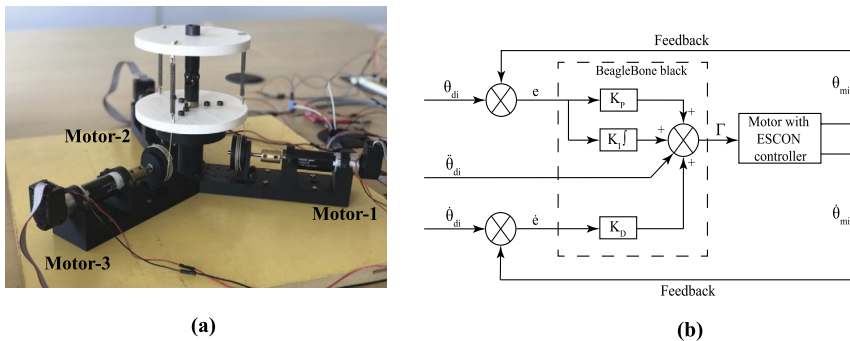


Figure 19.8 (a) Experimental setup of the tensegrity mechanism and (b) PID controller block employed in the program.

The force control algorithm was executed in C-program with the help of a BB black microcomputer.

19.3.3.2 Results of experiments

With the interfacing being done between the prototype test bench and micro-controllers, the experiments were carried out for two orientations of the tensegrity mechanism: Vertical and Horizontal. The vertical orientation is the natural pose of the mechanism as depicted in Fig. 19.8a. This orientation is also inline with the vertical orientation of the piping inspection robot. The horizontal orientation is the posture when the tensegrity mechanism is coupled with the bio-inspired robot and the assembly is moving inside a horizontal pipeline. For experiments, the sampling frequency of the control loop was set to 1000 Hz. Before the circular trajectory was performed, a linear tilt was initiated to tilt the mechanism along one of the springs. This operation was simpler compared to the equations of circular trajectory and can be understood in detail in [29]. For smoother operations, sleep routines were introduced in the control law. Thus, the total time to perform the experiments was around 124 s. The initial linear tilt was performed from 0 to $\pi/10$ radians. The circular trajectory was created from this tilt position in the counter-clockwise direction. During the experimental cycle, two of the three prismatic springs extended to a maximum length of 84.9 mm from the home position. The springs connected to each motor also reached a minimum length of 50 mm during the circular trajectory. From the obtained results, the error was calculated between the measured and desired angular positions of the pulley after gear reduction using the equation:

$$Error = 180 \frac{(\theta_{mi} - \theta_{di})}{\pi}. \quad (19.15)$$

Due to the frequency issues associated with the BB black microcomputer, higher noises were observed in the plots. In order to reduce these errors, the frequency of the results was measured in MATLAB and the Savitzky–Golay filtering method [30] was applied to reduce the noise. This filtering technique uses the least-squares method to smooth signals without distorting it. The plot for the error data after filtering for both orientations of the mechanism is represented below in Fig. 19.9.

It could be observed that the global error range lies between -0.01° to 0.02° for vertical orientation and -0.046° to 0.06° for horizontal orientation. The difference between measured and desired data was very much smaller and this proved the effectiveness of the PID control law. At each instance of the experiment, the torque induced on each motor to perform cable pull or push was also evaluated. The representation of motor torques

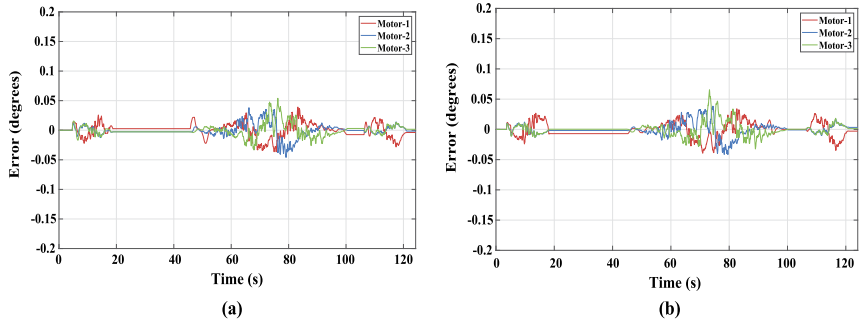


Figure 19.9 Joint position errors along the circular trajectory for the (a) vertical and (b) horizontal orientations of the mechanism.

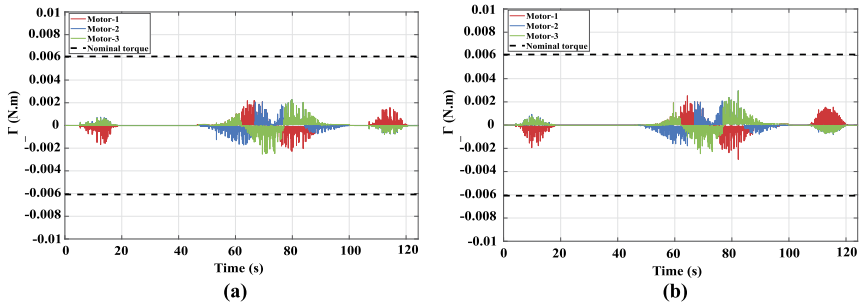


Figure 19.10 Motor torques under operation along the circular trajectory for the (a) vertical and (b) horizontal orientations of the mechanism.

after application of Savitzky–Golay filter for both orientations is provided in Fig. 19.10.

For the vertical orientation of the mechanism, the torque values were between -0.0021 N.m and 0.0023 N.m. However, for the horizontal orientation of the mechanism, the torque values were between -0.0025 N.m and 0.0027 N.m. This is because, the self-weight of the mechanism created an influence on the motor torques. On a global scale, these effects were comparatively smaller. With the addition of an external loading, significant differences can be observed between both orientations. Also, it must be noted that the operating torques were around one-third of the nominal motor torques and these values were well within the operating range of the DC-motor.

19.3.3.3 Discussions

The experiments with the circular trajectory allows to correlate with the theory of Tilt and Torsion [31]. Initially, the mechanism was made to tilt along a given direction. Followed by that, the azimuth angle was chosen from 0 to -2π radians for the given tilt position to have a complete circular trajectory. The corresponding $X - Y$ Euler angle equations for this trajectory were determined and passed as inputs to the control loop to execute the circular path. Owing to the higher simulation time and incorporation of intermittent sleep periods of around 20 s in the control loop, the initiation time for the circular trajectory was much longer. This phenomenon could be observed in the plots of errors and torques between 25–50 s. During this period, the motor torques were also equal to zero. When the mechanism is coupled with the bio-inspired robot, the circular trajectory can be incorporated in the control law of the entire robot assembly. Based on the pipeline profile encountered, the circular trajectory can be performed to align the axis of the robot with that of the pipeline after overcoming a bend or a junction.

19.4 Optimal design of the bio-inspired robot

When the tensegrity mechanism proposed in the earlier section was assembled with the bio-inspired piping inspection robot, the assembly proved to be oversized to pass through a 90° pipe bend or a junction. The main factors were associated with the sizing of the spindle drive screw and the slot-follower leg mechanism. In order to overcome this issue, two optimization problems were defined and solved. The first optimization problem aims to identify the size of motor units without the presence of leg mechanisms, while the second problem aims to determine the size of the leg mechanism for the results of first optimization problem.

19.4.1 First optimization problem

The first optimization problem aims to determine the sizing of motor modules of the inspection robot inside the test bench without the presence of leg mechanisms. The optimization problem was tested for pipeline diameters ranging from 70 to 160 mm. From the results of optimization, a suitable motor-spindle combination unit was identified from the catalog of Maxon. Apart from the motor sizing, the existing bio-inspired robot had issues associated with velocities and cable management. Based on results obtained

from the optimization problem, the Maxon motors were identified in such a way that the spindle drive has a lower gear reduction ratio to have higher velocity. The robot was modeled as multi-body planar blocks with a tensegrity mechanism between each module in MATLAB. Each module consisted a Maxon brushless DC-motor coupled with a spindle drive. The robot geometry considered for the optimization problem is represented below in Fig. 19.11.

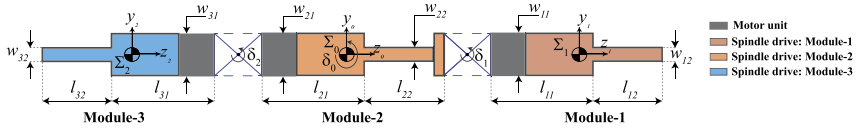


Figure 19.11 Representation of the robot assembly with various design parameters for the first optimization problem.

In Fig. 19.11, $[l_{k1}, w_{k1}]$ represents the length and width of motor and spindle drive unit (without the screw) for module k , whereas $[l_{k2}, w_{k2}]$ represents the length and width of output screw drive of spindle unit for module k . Here, k indicates a module and it assumes values from 1 to 3. The spindle screw length was one of the main factors that hindered the existing bio-inspired robot to overcome a pipe bend. This is the reason why the screw unit and motor-spindle unit were considered as separate design variables. The dimensions of the tensegrity mechanism were taken from the results of Section 19.3. The entire robot assembly, depicted in Fig. 19.11, was simulated inside a test pipe bench constructed in MATLAB through discretization equations. At each position of the robot, the constraints were evaluated to ensure that there exists no collision between each module and the pipeline walls [32]. The first optimization problem is then defined as follows:

$$\text{maximize: } \sum_{i=1}^n \sum_{k=1}^3 (l_{k1}w_{k1} + l_{k2}w_{k2})$$

subject to the constraints: $g_1, g_2, g_3, g_4, g_5, g_6,$

where k indicates the module, $i = 1..n$ indicates the discretized positions.

The six inequality constraints ensured that there exists no collision between inner and outer walls of the pipeline on each module. Using the *fmincon* function in MATLAB, the optimization problem was solved for 70 to 160 mm diameter pipelines. Finally, a motor-spindle drive combination

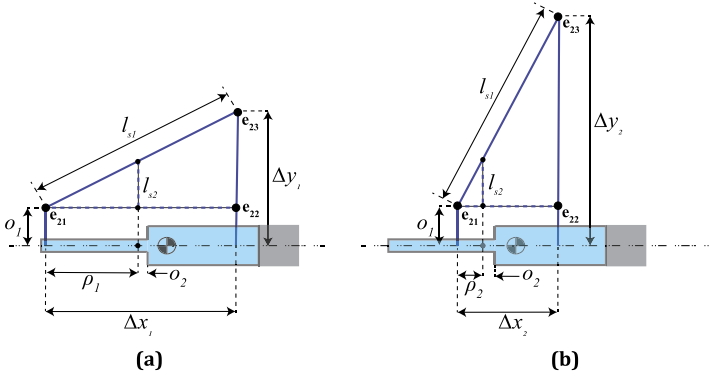


Figure 19.12 Representation of the design parameters of the leg mechanism during (a) declamped and (b) clamped phases.

with $[l_1, w_1, l_2, w_2] = [51.5, 16, 45, 5]$ mm was chosen [32,33] based on dimension and velocity factors. This motor-spindle drive combination holds good for pipeline range of 100–120 mm diameters.

19.4.2 Second optimization problem

Using the optimized dimensions of the motor-spindle unit, the dimensions of the leg mechanism were then identified. Unlike in [15], a geometric approach was followed to determine the dimensions. The design parameters of the leg mechanism for the second optimization problem are represented in Fig. 19.12.

The second optimization problem, which is used to estimate the size of the leg mechanism is defined by:

$$\text{maximize } \sum_{i=1}^n \sum_{k=1}^2 (\|\mathbf{e}_{k1} - \mathbf{e}_{k3}\| \|\mathbf{e}_{k3} - \mathbf{e}_{k2}\|)$$

subject to constraints: $g_1, g_2, g_3, g_4, h_1, h_2,$

where k indicates front & rear modules, i indicates the discretized positions.

For the second optimization problem, the central module was not taken into account as it does not accommodate the leg mechanisms. Similar to the first optimization problem, the inequality constraints g_1 to g_4 ensured that there are no collisions between leg mechanisms and pipeline walls. The two additional equality constraints h_1 and h_2 ensured that the stroke length ρ_1 remains at 32 mm and the parameter l_{s1} remains constant during clamp-

ing and declamping modes. Upon solving the problem, the dimensions of the leg mechanism were found to be $l_{s1} = 57$ mm and $l_{s2} = 9.7$ mm. By assembling the tensegrity mechanism, the DC-motor with spindle drive unit identified from the catalog of Maxon, the optimized slot-follower leg mechanism and standard fasteners such as circlips, bolts, etc., the entire robot was realized in a CAD software. The flexible robot resembles an “Elephant trunk” during a passive mode and it is demonstrated below in Fig. 19.13.

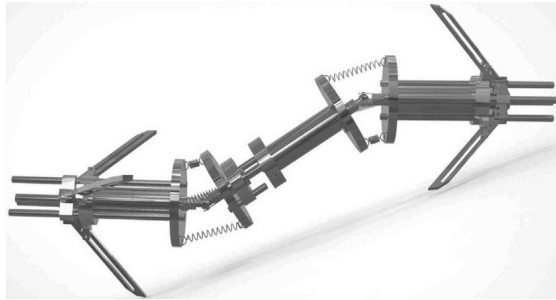


Figure 19.13 The correlation of the flexible robot assembly to an “Elephant trunk”.

19.5 Static force modeling of the flexible robot assembly

The static force modeling goes back to the models proposed in [16], however with the presence of the tensegrity modules in the assembly. When one set of leg mechanisms (left) are clamped with the pipeline walls, the self-weight of the robot imposes forces and moments at the CG position of the assembly. However, the tensegrity mechanism experiences a deflection due to the weight of the assembly. While working inside a closed environment, the robot assembly can assume any orientation about the z-axis. From previous studies and analysis, the tensegrity mechanism undergoes the maximum deformation when one of the spring is at a distance of r_f from the origin, as demonstrated below in Fig. 19.14a.

The tensegrity mechanism, which is deflected by the robot assembly, is represented in Fig. 19.14b. In Fig. 19.14b, w is the weight of the entire robot assembly, which can be extracted using the digital model of the prototype, and ϕ is the deflection of the mechanism. This analysis is currently under study and for ensuring a static stability, the optimal design parameters

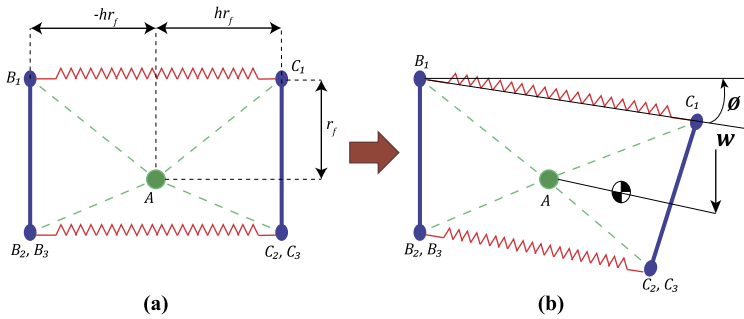


Figure 19.14 The tensegrity mechanism at (a) home-pose and (b) load of the robot assembly.

of the spring such as the stiffness, number of windings and coil diameter are determined. This study will then be extended by considering stacked tensegrity modules. The assembly sequence of this setup resembles to that of a series-parallel hybrid manipulator and the digital model of the robot to be studied in the future is represented below in Fig. 19.15.

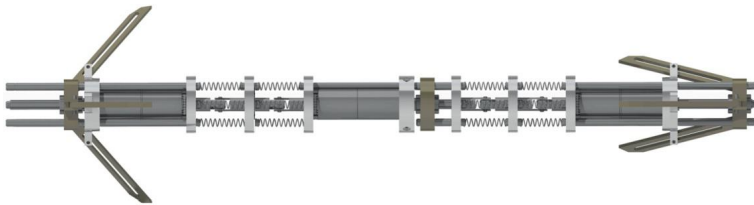


Figure 19.15 Representation of the flexible robot assembly with stacked tensegrity modules.

19.6 Conclusions and future works

This chapter mainly focused on the design of a flexible, bio-inspired piping inspection robot. The operational context was first defined wherein the robot must be capable of working inside pipelines having diameters less than 150 mm subject to varying diameters as well as profiles. In-line with the context, a rigid, bio-inspired, caterpillar-type robot proposed by Henry et al. [15] was taken into study. Numerical force models were created for this robot and experiments were carried out on the prototype using microcontrollers inside a test pipe. A comparison was then made between

the simulations and the experiments. The robot was then converted into a flexible system by the incorporation of a tensegrity mechanism, which can potentially operate in passive and active modes. Two types of tensegrity mechanisms were studied. The optimal design parameters were identified based on stability and preloading parameters for these mechanisms. Followed by that, singularity and workspace analyses were performed and it was found that the 4-SPS-U architecture was an ideal architecture that can address both active and passive compliance issues. Experiments were then carried out using a simple closed-loop PID control algorithm for the 3-SPS-U tensegrity mechanism in vertical and horizontal orientations. For given input tilt angles, the IKP was solved for the mechanisms for a circular trajectory. An optimization approach was then carried out to identify the sizing of motors and the leg mechanisms for the flexible robot to work through 90° pipe bends. From the results of optimization, the digital model of the optimized robot assembly was realized in CAD software to demonstrate an “Elephant trunk” type model.

Currently, the static force modeling of the flexible robot assembly is being carried out. An overview of the analysis was discussed in Section 19.5 with the 3-SPS-U architecture. Further analysis will be carried out by considering stacked 3-SPS-U mechanisms. The proposal of the stacked two-stage 3-SPS-U mechanism can offer similar tilt limits as the 4-SPS-U. Initially, and to meet the space constraints, the tensegrity mechanisms will be passive. To operate all the modules, a change in control architecture is necessary. Controllers such as EtherCAT or CANOpen is proposed to be employed as they allow greater flexibility in mounting and simplify the wiring between the motors and the control unit. The results of the static force model will help in identifying the appropriate actuator for the tensegrity modules. At present, DC motors are employed for actuation, however, alternative solutions such as the shape memory systems are also being considered.

References

- [1] I. Kassim, L. Phee, W.S. Ng, F. Gong, P. Dario, C.A. Mosse, Locomotion techniques for robotic colonoscopy, *IEEE Engineering in Medicine and Biology Magazine* 25 (3) (2006) 49–56.
- [2] F. Cosentino, E. Tumino, G.R. Passoni, E. Morandi, A. Capria, Functional evaluation of the endotics system, a new disposable self-propelled robotic colonoscope: in vitro tests and clinical trial, *The International Journal of Artificial Organs* 32 (8) (2009) 517–527.
- [3] S. Masuda, Apparatus for feeding a flexible tube through a conduit, UK patent 1 (6) (1978).

- [4] S. Kim, H. Min, H. Lim, Y. Lim, J. Park, B. Kim, Magnetic impact actuator for robotic endoscope, in: *Intelligent Microsystem Symposium, 2001*, pp. 45–49.
- [5] G.J. Iddan, D. Sturlesi, In vivo video camera system, US Patent 5,604,531 (Feb. 18 1997).
- [6] M. Drapier, V. Steenbrugge, B. Successeurs, Perfectionnements aux cathétères médicaux, France patent 1 (1961) 15.
- [7] K. Ikuta, M. Tsukamoto, S. Hirose, Shape memory alloy servo actuator system with electric resistance feedback and application for active endoscope, in: *Proceedings. 1988 IEEE International Conference on Robotics and Automation, IEEE, 1988*, pp. 427–430.
- [8] M. Utsugi, Tubular medical instrument having a flexible sheath driven by a plurality of cuffs, US Patent 4,148,307 (Apr. 10 1979).
- [9] M.R. Treat, W.S. Trimmer, Self-propelled endoscope using pressure driven linear actuators, US Patent 5,595,565 (Jan. 21 1997).
- [10] I. Ginsburgh, J.A. Carlson III, G.L. Taylor, H. Saghatchi, Method and apparatus for fluid propelled borescopes, US Patent 4,735,501 (Apr. 5 1988).
- [11] A. Nayak, S. Pradhan, Design of a new in-pipe inspection robot, *Procedia Engineering* 97 (2014) 2081–2091.
- [12] Y. Zhang, M. Zhang, H. Sun, Q. Jia, Design and motion analysis of a flexible squirm pipe robot, in: *2010 International Conference on Intelligent System Design and Engineering Application, vol. 1, IEEE, 2010*, pp. 527–531.
- [13] Y.-S. Kwon, H. Lim, E.-J. Jung, B.-J. Yi, Design and motion planning of a two-moduled indoor pipeline inspection robot, in: *2008 IEEE International Conference on Robotics and Automation, IEEE, 2008*, pp. 3998–4004.
- [14] J. Park, D. Hyun, W.-H. Cho, T.-H. Kim, H.-S. Yang, Normal-force control for an in-pipe robot according to the inclination of pipelines, *IEEE Transactions on Industrial Electronics* 58 (12) (2010) 5304–5310.
- [15] R. Henry, D. Chablat, M. Porez, F. Boyer, D. Kanaan, Multi-objective design optimization of the leg mechanism for a piping inspection robot, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 46360, American Society of Mechanical Engineers, 2014*, p. V05AT08A001.
- [16] D. Chablat, S. Venkateswaran, F. Boyer, Mechanical design optimization of a piping inspection robot, *Procedia CIRP* 70 (2018) 307–312.
- [17] R.V. Rao, G. Waghmare, A new optimization algorithm for solving complex constrained design optimization problems, *Engineering Optimization* 49 (1) (2017) 60–83.
- [18] D. Chablat, S. Venkateswaran, F. Boyer, Dynamic model of a bio-inspired robot for piping inspection, in: *ROMANSY 22—Robot Design, Dynamics and Control, Springer, 2019*, pp. 42–51.
- [19] S. Hirose, H. Ohno, T. Mitsui, K. Suyama, Design of in-pipe inspection vehicles for ϕ 25, ϕ 50, ϕ 150 pipes, in: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), vol. 3, IEEE, 1999*, pp. 2309–2314.
- [20] S. Venkateswaran, D. Chablat, A new inspection robot for pipelines with bends and junctions, in: *IFTToMM World Congress on Mechanism and Machine Science, Springer, 2019*, pp. 33–42.
- [21] L. Meirovitch, *Fundamentals of Vibrations*, Waveland Press, 2010.
- [22] C. Gosselin, J. Angeles, et al., Singularity analysis of closed-loop kinematic chains, *IEEE Transactions on Robotics and Automation* 6 (3) (1990) 281–290.
- [23] S. Venkateswaran, D. Chablat, Singularity and workspace analysis of 3-SPS-U and 4-SPS-U tensegrity mechanisms, in: *International Symposium on Advances in Robot Kinematics, Springer, 2020*, pp. 226–233.

- [24] R. Jha, D. Chablat, L. Baron, F. Rouillier, G. Moroz, Workspace, joint space and singularities of a family of delta-like robot, *Mechanism and Machine Theory* 127 (2018) 73–95.
- [25] D. Chablat, E. Ottaviano, G. Moroz, A comparative study of 4-cable planar manipulators based on cylindrical algebraic decomposition, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 54839, 2011, pp. 1253–1262.
- [26] D. Chablat, P. Wenger, Working modes and aspects in fully parallel manipulators, in: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 3, IEEE, 1998, pp. 1964–1969.
- [27] S. Caro, D. Chablat, P. Lemoine, P. Wenger, Kinematic analysis and trajectory planning of the orthoglide 5-axis, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 57144, American Society of Mechanical Engineers, 2015, p. V05CT08A004.
- [28] W. Khalil, E. Dombre, *Modeling Identification and Control of Robots*, CRC Press, 2002.
- [29] S. Venkateswaran, D. Chablat, Trajectory planning for a 3-SPS-U tensegrity mechanism, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 85451, American Society of Mechanical Engineers, 2021, p. V08BT08A004.
- [30] Savitzky–Golay filtering, <https://fr.mathworks.com/>. (Accessed 28 August 2020).
- [31] S. Venkateswaran, M. Furet, D. Chablat, P. Wenger, Design and analysis of a tensegrity mechanism for a bio-inspired robot, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 59230, American Society of Mechanical Engineers, 2019, p. V05AT07A026.
- [32] S. Venkateswaran, D. Chablat, P. Hamon, Design of a piping inspection robot by optimization approach, in: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 83990, American Society of Mechanical Engineers, 2020, p. V010T10A018.
- [33] S. Venkateswaran, D. Chablat, P. Hamon, An optimal design of a flexible piping inspection robot, *Journal of Mechanisms and Robotics* 13 (3) (2021).

This page intentionally left blank

CHAPTER 20

Optimization of parallel mechanisms with joint limits and collision constraints

Durgesh H. Salunkhe^a, Shivesh Kumar^b, and Damien Chablat^a

^aNantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, Nantes, France

^bRobotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

Owing to their advantages, parallel kinematic manipulators (PKM) are employed as sub-mechanism modules in various fields such as humanoid robots (THOR [1], LOLA [2], Charlie [3]), exoskeletons [4,5], haptic interfaces [6], surgeries [7], and industrial applications [8,9]. An extensive survey on PKM with classification based on degrees of freedom and their applications is presented in [10]. PKMs are also widely used in high-speed industrial assembly lines, like the DELTA + 1 DOF wrist robot [11]. Another significant application of PKMs is in machining tasks such as milling operations and high-speed machining tasks [12–14].

Given the broad range of applications, PKM designs must cater to user needs and adhere to constraints associated with different processes. These needs may involve robot mobility, workspace size, movement precision, dynamic performance, and stiffness. Numerous performance indices have been established to address these requirements, which can be applied to optimization problems. Some workspace and kinematic performance indices include the Jacobian matrix conditioning, velocity amplification factors [15–17], regular workspace shapes [18], and safe working zones [19].

Various optimization methods have been proposed for mechanism synthesis in the past. Some employ the mathematical formulation of the objective function to implement gradient descent methods [20], while others use numerical approaches and evolutionary algorithms when the objective function is not available in closed form or gradient-based algorithms cannot be used. Some of these algorithms include Differential Evolution (DE) [21], Genetic Algorithms (GA) [22], Branch and Prune [23], Interval-based analysis [18], and Non-dominated Sorting Genetic Algorithm II (NSGA-II) [24–26] for multi-objective optimization. These methods are typically

computationally expensive, with efficiency heavily reliant on population size.

A recent development in mechanism design optimization involves co-optimization with motion trajectories [27]. This approach employs efficient algorithms to explore implicitly defined manifolds, leveraging the advantages of representing the problem as an implicit function for faster and more efficient convergence.

Local search methods can decrease the computational cost of mechanism optimization. The Nelder–Mead algorithm, a geometric-based search for the next best solution, is well-suited for mechanism optimization, as it allows easy optimization of link lengths. To prevent convergence to a local optimum, different methodologies combine local optimization methods with global searches [28–31].

This chapter introduces an accelerated, general algorithm for PKM design optimization that is flexible concerning objective function definition and adaptable to constraints. The method optimizes the design for the maximum safe working zone while considering the physical stroke of the prismatic actuator. A fast local search algorithm, the Nelder–Mead algorithm, combined with a global search procedure, enables quicker progress towards a global optimum, even for mechanisms with computationally expensive objective functions.

20.1 Design considerations in PKM optimization

In the parallel kinematic mechanism design, the following choices have to be made:

1. Architecture of the manipulator (e.g., $3R\underline{R}R$ (Revolute–Revolute(acted)–Revolute), $3R\underline{P}R$ (Revolute–Prismatic(acted)–Revolute), etc.)
2. Type of joints: different combinations of joints to achieve the same degrees of freedom (dof) (e.g., $\underline{U}PS$ (Universal–Prismatic(acted)–Spherical), $\underline{R}US$, $\underline{R}RPS$)
3. Pose of the joints: where and how to place a particular joint's frame?

Making a particular choice is non-trivial, especially because of its effect on the workspace, the direct and inverse kinematic model, and the size of the mechanism. Another interesting challenge is that the same architecture can perform different tasks with either kinematic or dynamic constraints and thus have to be optimized accordingly. The following subsections elaborate

on the common objective functions and constraints involved in mechanism optimization to motivate the choice of the algorithm.

20.1.1 Objective function

It is important to evaluate the quality of the motion performed while designing a manipulator with kinematic characteristics. The quality indices widely used in the past are the conditioning number [15] and the manipulability ellipsoid [16]. The feasible workspace and the global quality of the manipulator are directly related in the presented case and thus can be implemented together with appropriate weights.

20.1.1.1 Manipulator workspace

If the workspace involves only orientation or translation, Regular Dextrous Workspace (RDW) is an objective function representing an n -dimensional sphere within the n -dimensional output space. The necessary workspace is not considered a constraint, but rather, the algorithm aims to achieve the largest feasible workspace within the desired RDW (RDW_d) [18]. Concurrently, the notion of *safe working zone* for parallel manipulators has been presented in [19], defining a feasible workspace as one devoid of singularities, internal link collisions, and adhering to passive joint limits. The feasible set (\mathcal{F}) concept in this text pertains to the collection of all points in the discretized output space (\mathcal{K}) such that:

1. They correspond to non-singular configurations.
2. Adhere to passive joint limitations.
3. Ensure no internal collisions between actuators and the moving platform in all postures.

20.1.1.2 Quality of the manipulator

To measure the motion quality, the conditioning number (κ) was introduced in [15]. It signifies the asymptotic worst-case relative change in the output for a relative change in the input, evaluating the output sensitivity to input changes. The geometrical interpretation of κ relates to the ellipsoid's eccentricity proportionality, providing information about the ease of movement in a specific direction from the current end effector pose. When the κ equals 1, it corresponds to a sphere and the *isotropic configuration*. The κ value ranges from 1 to ∞ , and its inverse, κ^{-1} , is used for bounded values and is given by (20.1), where σ represents the Jacobian matrix, \mathbf{J} , singular

values.

$$\kappa^{-1} = \frac{\sigma_{min}}{\sigma_{max}}, \quad \kappa^{-1} \in [0, 1]. \quad (20.1)$$

The Jacobian matrix's dimensional non-homogeneity affects the conditioning number and is unsuitable for manipulators whose workspace is not a subset of either \mathbb{R}^3 or $SO(3)$ [32]. This issue is vital to consider when implementing the proposed optimization methodology for a general manipulator. The manipulators shown in Section 20.3 have only rotational degrees of freedom (DOF), so the inverse conditioning number is chosen as the quality index. A *global conditioning index* (κ_g^{-1}) (GCI), the mean quality index (κ^{-1}) over the RDW, is defined as follows,

$$\kappa_g^{-1} := \frac{\sum_{RDW_d} \kappa^{-1}}{1}. \quad (20.2)$$

20.1.2 Constraints

PKM's most common constraints to implement feasible workspace include:

- Non-singular constraint;
- Passive and active joint limits;
- Internal collision constraints;
- Feasible actuator range.

Among these constraints, the first three are self-explanatory and are not explained further. The constraint regarding feasible actuator range is pertinent in optimizing PKM with prismatic actuators and is discussed in detail to emphasize its importance.

20.1.2.1 Feasible actuator range

The active joint ranges are an essential constraint during PKM design. This constraint is particularly relevant to mechanisms with prismatic joints as actuators. The goal is to constrain the actuator selection to maximize the points in $\mathcal{F} \cap RDW_d$. Typically, a prismatic joint is represented as a constraint with a specific minimum and maximum range and with a constraint on the ratio between the length in the fully actuated state and its default length:

$$\rho_{min} \leq \rho \leq \rho_{max}, \quad (20.3)$$

$$\rho_{max} \leq \text{stroke} \cdot \rho_{min}, \quad \text{stroke} \in [1, 2]. \quad (20.4)$$

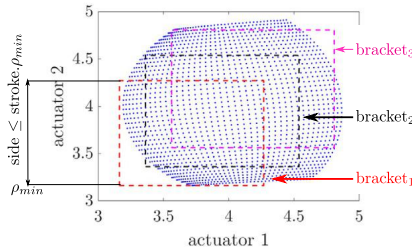


Figure 20.1 Different search brackets within the actuator space (input space Z). The dots correspond to the pair of lengths of actuators for a configuration in RDW.

Eq. (20.4) originates from the physical structure of general prismatic joints. If the actuator's unextended length is ρ_{min} , then it is impractical for typical prismatic joints to extend beyond their original length ($\rho_{max} < 2 \cdot \rho_{min}$). The novelty in expressing the actuator range in the current work is that we do not have a static value as a limit, as mentioned in Eq. (20.3), i.e., we express the constraint solely in terms of the stroke ratio defined in Eq. (20.4). This allows us to select the optimal actuator ranges to maximize the feasible workspace without imposing constraints on the prismatic joint's minimum or maximum size. This is demonstrated in Figs. 20.1 and 20.2, which introduce an example of a 2 dof 2UPS-1U orientation mechanism from [7]. The points in the dotted space in Fig. 20.1 correspond to actuator lengths in a feasible configuration. The objective is to search for an optimized bracket, $[\rho_{min}, \rho_{max}]$, i.e., a bracket encompassing as many blue points as possible, with the constraint that the square's side does not exceed a given proportion relative to its minimum value.

An algorithm presented in [33] (Algorithm 1) details the method used to obtain the optimized bracket for the actuators. After discretizing the RDW_d , we acquire the set of all valid points belonging to \mathcal{F} . Upon calculating the actuator length values at each point, the minimum ρ_{min} and maximum ρ_{max} value for the actuator is determined. The algorithm input is an $n \times 3$ matrix for the n valid points, with columns corresponding to the actuator lengths and the evaluation at that point. If the ratio of the maximum value to the minimum value of the actuator length respects the stroke ratio, the algorithm returns the actuator range without modification. Otherwise, a bracket of $[\rho_{min}, \text{stroke} \cdot \rho_{min}]$ is generated, and the actuator length values for each point in the set of valid points are checked against

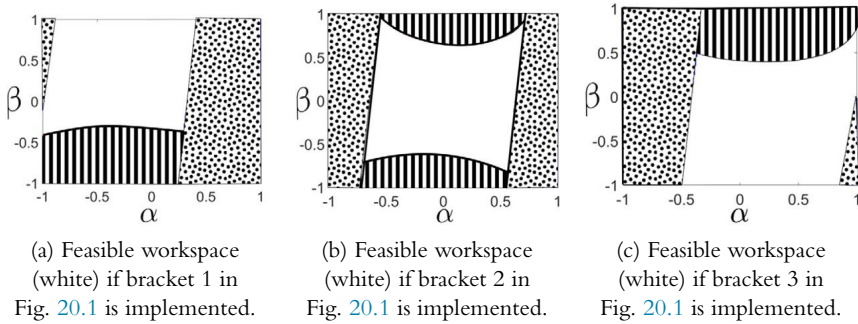


Figure 20.2 Comparison of feasible workspace (white space) within the RDW_d for different search brackets and a specific mechanism (2UPS-1U). The striped and dotted part represent the violation due to actuator lengths of first and second leg, respectively.

the bracket, and the number of points satisfying the bracket is stored. This process repeats by incrementing the ρ from ρ_{min} to $stroke.\rho_{min}$.

20.1.2.2 Implementation of constraints and evaluation function

The process of implementing constraints and evaluating the performance of a set of parameters is explained in Algorithm 20.1. The optimization space is first discretized to evaluate the parameters, and each point is evaluated for compliance with the constraints. Some constraints are strictly enforced, meaning that if any point in the RDW_d violates them, the set of parameters is considered invalid. In the current algorithm, the singularity constraint is strict. If the singularity curve intersects with even one point of RDW_d , the evaluation of the given parameters is negative. If the RDW_d is singularity-free, each point is evaluated for compliance with other constraints, such as passive joint limits and collision constraints. If a point satisfies all the constraints, it is rewarded with the corresponding κ^{-1} value. If any constraint (except singularity) is violated, the point in RDW_d is given a 0 value.

As each point in the discretized workspace is evaluated, the final evaluation is the cumulative value of κ^{-1} over the workspace where all constraints are satisfied. The rewarding strategy can be customized per the designer's requirements, and appropriate weightage can be assigned to the constraints to achieve an optimized design for a specific need. The algorithm demonstrates modularity with the constraints, where each constraint is independent. The flexibility to activate, deactivate, or add constraints without changing the algorithm is particularly useful for mechanism de-

sign. The designer can experiment with various constraints to understand their effects on the final feasible workspace. Each constraint can be designed to reward or penalize a specific set of parameters, allowing for a mix of strict and non-strict constraints in the optimization. The designer can also identify which constraint hinders the optimization and requires modification.

In summary, evaluating a given set of parameters involves discretizing the optimization space, assessing each point for compliance with the constraints, and rewarding the points that satisfy all constraints. The algorithm is modular and flexible and allows the designer to experiment with various constraints to achieve an optimized design for specific needs.

20.2 Proposed algorithm for mechanism optimization

In this section, we present the complete optimization method. As discussed in previous sections, the goal is to develop an algorithm capable of managing non-smooth objective functions and PKM design constraints. This section is organized into three subsections, explaining the local search, global search, and the approach used to combine them for faster and more efficient solutions, respectively.

20.2.1 Local search algorithm: the NM (NM) algorithm

The NM-algorithm, a derivative-free optimization algorithm, was proposed by John Nelder and Roger Mead [34]. It is also called the *downhill-simplex algorithm* since it employs *simplexes* to conduct a local space search. In this section, we introduce the algorithm for a *single start*, which looks for the optimal solution in the local vicinity of the initial simplex. We then discuss the algorithm's application in mechanism optimization and describe the method for extracting the best actuator ranges from the solution. The section concludes with an overview of the algorithm and its implementation, highlighting its strengths and weaknesses.

To avoid *premature convergence* in an n -dimensional optimization space (\mathcal{O}), a simplex with at least $n+1$ points in \mathcal{O} is needed. As shown in the figure, this can be visualized with a simple graphic for a 2-dimensional, \mathcal{O} . The algorithm starts with a sorted simplex of $n+1$ points ($\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n$) such that the objective function evaluated at the i th vertex has a value better than or equal to that of the $(i+1)$ th vertex. A mean point (\mathbf{v}_m) is calculated by

Algorithm 20.1: Method to calculate the evaluation and range of actuators for a given set of parameters.

Result: evaluation at a given point in optimization space and the corresponding actuator lengths

```

1 input  $\rightarrow \mathbf{v}$   $\triangleright$  It is a n-dimension point in given n-dimension optimization
   space;
2  $x_i, i \in 1, \dots, n,$   $\triangleright$   $i^{\text{th}}$  variable of the n-dimension optimization space;
3  $\rho_1$  and  $\rho_2$   $\triangleright$  actuator lengths at a given configuration;
4  $e = 0$   $\triangleright$  Initializing the evaluation;
5 for  $x_1$  from  $x_{1min}$  to  $x_{1max}$  by  $interval_1$  do
6   ...  $\triangleright$  Add loops as a function of the dimension of the space
7   for  $x_n$  from  $x_{nmin}$  to  $x_{nmax}$  by  $interval_n$  do
8      $f(\mathbf{v})$   $\triangleright$  function that solves IGS, collision distance and  $\kappa^{-1}$ ;
9      $[\det(\mathbf{J}), \mathbf{q}_p, \rho_1, \rho_2, \kappa^{-1}, d_c] = f(\mathbf{v});$ 
10     $f(\mathbf{v})$  returns the value of the determinant of Jacobian, the passive
       joint angle vector,  $\mathbf{q}_p$ , actuator lengths,  $[\rho_1, \rho_2]$ , the inverse of
       the conditioning number,  $\kappa^{-1}$  and the collision distance,  $d_c$ ,
       between the actuators;
11     $\triangleright$  1. Checking for singularity constraints;
12    if  $\det(\mathbf{J})$  is 0 then
13      |  $e = -\infty;$ 
14      | break;
15    else
16      |  $reward = \kappa^{-1}$ 
17    end
18     $\triangleright$  2. Checking the passive joint limits;
19    for  $i$  from 1 to  $length$  of  $\mathbf{q}_p$  do
20      | if  $q_{pi} \geq q_{pmax}$  or  $q_{pi} \leq q_{pmin}$  then
21        |  $reward = 0$ 
22      | else
23        |  $reward = \kappa^{-1}$ 
24      | end
25    end
26     $\triangleright$  3. Checking for collision constraints;
27    if  $d_c \geq threshold$  then
28      |  $reward = 0$ 
29    end
30     $e = e + reward;$ 
31     $valid\_points[i] = [\rho_1, \rho_2, reward];$ 
32  end
33  ...
34 end
35 Implement the algorithm presented in [33] (Algorithm 1);
36 return  $valid\_points, e, \rho_1, \rho_2$ 

```

excluding the worst point (\mathbf{v}_n):

$$\mathbf{v}_m := \frac{\sum_{i=0}^{n-1} \mathbf{v}_i}{n}. \quad (20.5)$$

The optimization algorithm then compares the mean point and searches for better points by geometrical operations termed as (i) reflection, (ii) expansion, (iii) inside contraction, (iv) outside contraction, and (v) shrinkage. These operations are defined as follows:

1. Reflection (\mathbf{v}_r):

$$\mathbf{v}_r = \mathbf{v}_m + r(\mathbf{v}_m - \mathbf{v}_n), \quad r = \text{reflection coefficient } (r > 0) \quad (20.6)$$

2. Expansion (\mathbf{v}_e):

$$\mathbf{v}_e = \mathbf{v}_m + e(\mathbf{v}_r - \mathbf{v}_m), \quad e = \text{expansion coefficient } (e > 1) \quad (20.7)$$

3. Outside contraction (\mathbf{v}_{oc}):

$$\mathbf{v}_{oc} = \mathbf{v}_m + k(\mathbf{v}_m - \mathbf{v}_n), \quad k = \text{contraction coefficient } (0 < k < r) \quad (20.8)$$

4. Inside contraction (\mathbf{v}_{ic}):

$$\mathbf{v}_{ic} = \mathbf{v}_m - k(\mathbf{v}_m - \mathbf{v}_n), \quad k = \text{contraction coefficient} \quad (20.9)$$

5. Shrinkage:

$$\forall i \in [1, n] \quad \mathbf{v}_i = s \cdot \mathbf{v}_i, \quad s := \text{shrinkage factor } (0 < s < 1) \quad (20.10)$$

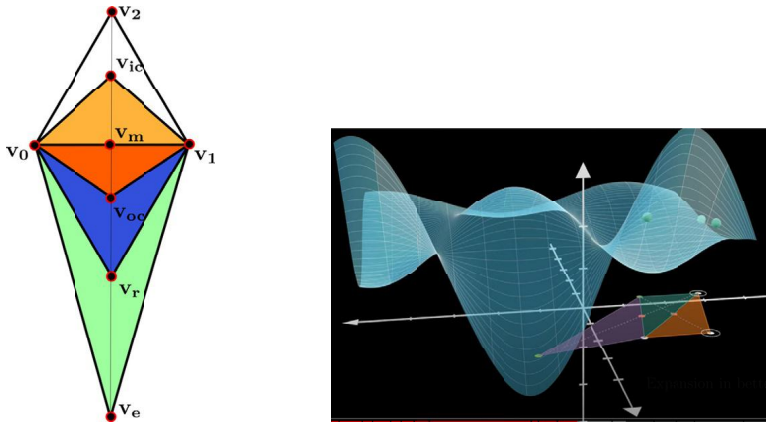
The introduction of a new point (\mathbf{v}_n) into the simplex relies on the evaluation of \mathbf{v}_r , \mathbf{v}_e , \mathbf{v}_{oc} , and \mathbf{v}_{ic} (refer to Algorithm 20.2). The process continues until the stopping criteria are met. The simplex halts if it shrinks below a specific value, ϵ_1 , and the evaluations of every vertex of the reduced simplex deviate by a maximum threshold, ϵ_2 . The algorithm can also be stopped by limiting the number of iterations. Algorithm 20.2 provides the full procedure for a single start of the Nelder–Mead (NM)-algorithm, and the stopping criteria algorithm can be found in [33] (Algorithm 3). Fig. 20.3a illustrates an example of the operations in a 2-dimensional optimization space, \mathcal{O} , demonstrating the geometric search nature of the \mathcal{O} in the NM-algorithm. Fig. 20.3b graphically depicts an example of the points explored during an optimization process. The optimization space of 2 dimensions of the evaluation is a function of these parameters.

Algorithm 20.2: Single start of the NM-algorithm.**Result:** Local minimum evaluation and the optimized parameters

```

1 initial sorted simplex  $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{n-1}, \mathbf{v}_n\}$ ;
2 evaluations  $\{e_0, e_1, e_2, \dots, e_{n-1}, e_n\}$ ;
3 while  $stop = 0$  do
4   calculate  $\mathbf{v}_m, \mathbf{v}_r$  and  $e_r$ ;
5   if  $(e_n < e_r < e_0)$  then
6      $\mathbf{v}_n = \mathbf{v}_r$ ;
7   else if  $(e_0 < e_r)$  then
8     if  $(e_r < e_e)$  then
9        $\mathbf{v}_n = \mathbf{v}_e$ ;
10    else
11       $\mathbf{v}_n = \mathbf{v}_r$ ;
12    end
13  else if  $(e_n < e_r < e_{n-1})$  then
14    if  $(e_{oc} > e_r)$  then
15       $\mathbf{v}_n = \mathbf{v}_{oc}$ ;
16    else
17       $\forall i \in [1, n] \quad \mathbf{v}_i = s \cdot \mathbf{v}_i$ ;
18    end
19  else if  $(e_r > e_n)$  then
20    if  $(e_{ic} > e_r)$  then
21       $\mathbf{v}_n = \mathbf{v}_{ic}$ ;
22    else
23       $\forall i \in [1, n] \quad \mathbf{v}_i = s \cdot \mathbf{v}_i$ ;
24    end
25  sort the simplex;
26  if  $\mathbf{v}_{0new} > \mathbf{v}_0$  then
27     $iter = 0$ 
28  else
29     $iter = iter + 1$ 
30  end
31  Update 'stop' from algorithm in [33] (Algorithm 3)
32 end
33 return  $\mathbf{v}_0, e_0$ 

```



(a) An example of an operation on a simplex (defined by v_0, v_1, v_2) in 2-dimensional \mathcal{O} . (b) An example of the travel path of optimization in NM-algorithm.

Figure 20.3 The single start of the Nelder–Mead local search.

20.2.1.1 Advantages and drawbacks of the NM-algorithm

The NM-algorithm offers a simple approach for modeling optimization problems in mechanism design, enabling the development of a general methodology applicable to any parallel mechanism. As a derivative-free algorithm, it introduces complex objective functions that may be difficult to formalize, such as the quality index, κ_g^{-1} , described in Section 20.1.2. Additionally, the NM-algorithm is a local search algorithm that returns a stationary point in a relatively short time compared to currently employed global optimization methods. This enables the designer to develop computationally expensive objective functions and construct constraints modularly, facilitating experimentation with different constraints throughout development. The geometric search method inherent to the NM-algorithm is another significant advantage relevant to mechanism design. The optimization space's foundation in the NM-algorithm is the optimization variables themselves, making it logical to use this method as the following best design parameters are chosen based on the combination of previous simplex parameters, rather than using complex methods to represent a mechanism in the optimization space that may not have a geometrical explanation for selecting the next best proposal (e.g., chromosomes in Genetic Algorithm). It is also possible to tune exploring parameters, such as reflection, expansion, contraction, and shrinkage coefficients, using human intuition and prior knowledge regarding the importance of different parameters.

Although well-suited for our application, the NM-algorithm has certain disadvantages. It has been proven to converge to dimension two under specific hypotheses [35], but lacks proof of convergence for optimization beyond two dimensions. It can collapse simplex patterns if implemented incorrectly, leading to convergence to a non-stationary solution [36]. Convergence strongly depends on the initial simplex size and the coefficient choices, as discussed in [37]. Despite these limitations, the NM-algorithm is valuable for our purposes, as the goal is not to find the absolute optimized design parameter, but to satisfy all constraints and achieve acceptable performance quality. Indeed, it has been successfully implemented in various applications [30,31]. Convergent variants have been proposed to circumvent premature convergence [38], allowing the algorithm to explore additional points in case of near collapse.

The NM-algorithm's local search is combined with a multi-start technique for global search in the optimization space, as discussed in the following section.

20.2.2 Global search algorithm

The NM algorithm has been combined with other global search methods, such as low-discrepancy points [39], genetic algorithm [28], and Powell optimization [29], in previous work. To explore global optimization space, we implement a multi-start NM-algorithm with low discrepancy points [40–42]. In this approach, the NM-algorithm is executed with different initial simplexes. It is crucial to have uniformly distributed initial simplexes to explore the maximum optimization space area.

20.2.2.1 Starting simplexes for multi-start

A practical approach to generate a sampling set $\mathcal{O}_M \subset \mathcal{O}$ is *Monte Carlo sampling* using a uniform distribution (refer to, for instance, [43]), or in other words, *random sampling*. Unfortunately, it is recognized [41] that these points tend to create clusters, particularly in high-dimensional scenarios, compromising the uniformity of the discretization. A better alternative involves distributing the M points of the discretization, \mathcal{O}_M of \mathcal{O} more evenly. Specifically, the points should be sufficiently close together without leaving any under-sampled regions. Certain deterministic sampling methods can be employed for this purpose, as demonstrated in [42,44]. The characteristics of such techniques are explained in [42]. The study in [42] proposes that an effective strategy to generate uniformly dispersed deterministic point sets involves using finite segments of so-called *low-discrepancy sequences* like the

Halton sequence, the *Hammersley sequence*, and the *Sobol sequence*. The present work applies initial simplexes selected from the Sobol sequences, as they demonstrate a more uniform distribution.

In [44], a sampling of a 2-dimensional unit cube was compared using a sequence of 500 points based on the uniform distribution and a sampling of the same cube obtained through a low-discrepancy sequence (in this instance, the *Sobol sequence* [45]). It has been sufficiently demonstrated that the second sequence provides better space coverage, with the largest voids among the points occurring in the case of uniform distribution.

20.2.3 Cascade optimization

In a standard execution of the NM-algorithm, the iteration ceases either when the simplex has contracted to a desirable size with approximately equal evaluations or if the same best point is encountered for a predetermined maximum number of iterations, as referred to the stopping algorithm [33] (Algorithm 3). Aiming to reduce the time for local convergence and enable the exploration of more initial simplexes, we adopt a method inspired by rough and fine-turning practices in lathe machines. Generally, when removing excess material from a workpiece as quickly as possible, the feed rate is increased, and the focus is not on the work's finish. Later, the feed rate is reduced when approaching the desired dimensions and the emphasis shifts to the work's finishing. Fig. 20.4 depicts the algorithm's entire flow. Initially, the simplexes derived from the Sobol sequence are integrated into the multi-start NM-algorithm, and a coarse search is conducted for local optima. Subsequently, local optima from some selected initialized simplexes are utilized to enforce stricter stopping criteria, enabling convergence to a stationary point with higher precision. We discard local optima that do not promise satisfactory evaluations even after an extended search, significantly reducing computation time. Moreover, with an optimized vertex as an initial simplex, we can construct the remaining vertices according to our preferences, thereby regulating the initial simplex's size.

20.2.3.1 Coarse search

In the coarse search, our goal is to hasten local convergence, enabling us to maximize the number of starts in our optimization approach. This is achieved by employing a coarser search space and easing the stopping criteria. During the coarse search, the output space is discretized using an

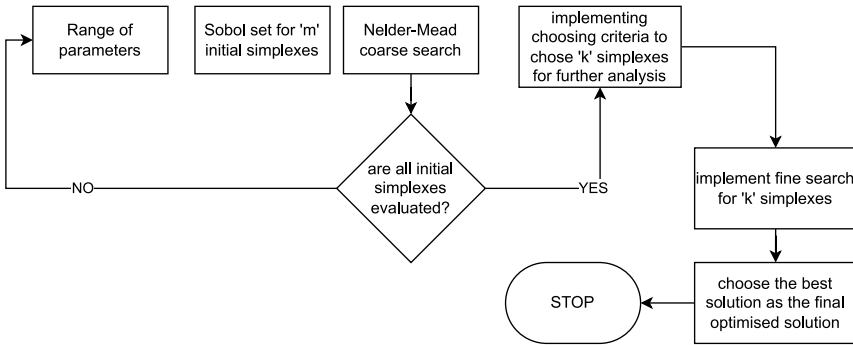


Figure 20.4 The flowchart for the complete implemented optimization methodology.

interval ten times larger than in the finer search, significantly reducing computational time. The objective of the coarse search is to identify simplexes situated on comparatively steeper slopes in the optimization space. By relaxing the stopping criteria, the maximum number of iterations permitted to repeat with the same evaluation is limited to 10, which aids in terminating the local search more quickly. One example of such a coarse implementation is detailed in Algorithm 20.3, where the condition for incrementing the iteration is altered. We apply a condition stating that the new evaluation found is considered better than the previous one only if it surpasses the last evaluation by 5%. The algorithm ceases as soon as we reach 90% of the maximum expected value.

20.2.3.2 Fine search

In the fine search, we sift through various local optima obtained from the coarse search. The evaluations of the local optima are sorted in ascending order, and the top 10% of the gathered optima are selected for further evaluation. In the fine search, we enforce stricter stopping criteria, modify the constraint of maximum expected evaluation to 100%, and discretize the output space with a ten times finer interval. The threshold considered an improvement is reduced to 1%. These changes directly impact the computational time and result in a significantly longer duration with an increasing dimension of the output space. All the optimized parameter sets from the NM-algorithm with stricter constraints are compared, and the best point is suggested as an optimized parameter of the PKM.

Algorithm 20.3: Implementation of coarse and fine local search criteria.

Result: Optimized parameter set \mathbf{v}_0

- 1 input: Initial set of simplexes;
- 2 e_0 : the best evaluation from the previous iteration;
- 3 e_{max} : Maximum expected evaluation;
- 4 limit: The percentage of maximum evaluation that is considered best;
- 5 For coarse search;
- 6 max_iter = 3 n;
- 7 margin = 1.05 ... (suggesting $\geq 5\%$ increment is considered improvement);
- 8 limit = 0.8 ... (suggesting that 80% of maximum evaluation is a criterion to stop);
- 9 For fine search;
- 10 max_iter = 10 n;
- 11 margin = 1.01 ... (suggesting $\geq 1\%$ increment is considered improvement);
- 12 limit = 1;
- 13 stop = 0;
- 14 **while** stop = 0 **do**
- 15 Perform Algorithm 20.2 except for last step of checking stop from algorithm in [33] (Algorithm 3);
- 16 Perform Algorithm 20.1 with finer intervals;
- 17 **if** $e_{new} \geq margin \times e_0$ **then**
- 18 | iter = 0
- 19 **else**
- 20 | iter = iter + 1
- 21 **end**
- 22 **if** iter \geq max_iter **then**
- 23 | **return** stop = 1;
- 24 **end**
- 25 **if** $e_{new} \geq limit \times e_{max}$ **then**
- 26 | **return** stop = 1;
- 27 **end**
- 28 **end**
- 29 **return** \mathbf{v}_0 from the Algorithm 20.2

Algorithm 20.4: An example of implemented multi-start optimization.

Result: Optimized parameter set of the mechanism and its evaluation

- 1 Assuming we have ‘m’ starts for an ‘n’ dimensional optimization problem;
 - 2 Choose $m \cdot (n+1)$ valid n-dimensional points from the Sobol set generated;
 - 3 Choose ‘k’ local optima for further fine search; generally, $k \leq 0.1 m$;
 - 4 **for** $start = 1:m$ **do**
 - 5 Initial simplex = $\{\mathbf{v}_{(m-1) \cdot (n+1)} \dots \mathbf{v}_{mn+m-1}\}$;
 - 6 Implement Single start from Algorithm 20.2 with coarse search from Algorithm 20.3;
 - 7 $\mathbf{v}_{chosen}(start, 1 : n + 1) = [\mathbf{v}_0, e_0]$;
 - 8 **end**
 - 9 sort \mathbf{v}_{chosen} by evaluation of the corresponding parameter set;
 - 10 **for** $fine_start = 1:k$ **do**
 - 11 Generate n more parameter sets around $\mathbf{v}_{chosen}(fine_start)$;
 - 12 Implement Single start from Algorithm 20.2 with fine search from Algorithm 20.3;
 - 13 $\mathbf{v}_{fine}(fine_start, 1:n+1) = [\mathbf{v}_0, e_0]$;
 - 14 **end**
 - 15 sort \mathbf{v}_{fine} by evaluation of the corresponding parameter set;
 - 16 **return** $\mathbf{v}_{fine}[1, 1 : n], \mathbf{v}_{fine}[n + 1]$
-

20.3 Results and discussion

The optimization algorithm described in this chapter was employed to optimize a 2UPS-1U parallel mechanism to verify the general implementation. The chosen mechanism is widely utilized in the industry, and the significance of the selected objective function is also discussed in this section. An open-source implementation of the proposed algorithm and examples, including the lambda mechanism (1 dof) and 3RRR mechanism (3 dof), can be found at: <https://github.com/salunkhedurgesh/ParaOpt>.

20.3.1 1-DOF lambda mechanism

The λ -mechanism is a singular closed-loop (1-RR_pR) mechanism utilized in legged robots as a simplification of the revolute joint [2,46,47], as depicted in Fig. 20.5. This mechanism is employed for stiffer actuation when a compact yet strong force is necessary and non-linear transmission characteristics are preferred. The constraint equations in this situation are simple and have been thoroughly examined in [48]. The mechanism was optimized using the determinant of the Jacobian, $\det(\mathbf{J})$, as the GCI and a modified VAF. The determinant is a scalar for the given case. For the lengths and variables illustrated in Fig. 20.5, the computations for these measures are:

$$\begin{aligned} \rho^2 &= l_1^2 + l_2^2 - 2l_1l_2 \cos(\theta) \\ \det(\mathbf{J}) &= l_1l_2 \frac{\sin(\theta)}{\rho} \\ \text{GCI}_i &= \det(\mathbf{J}), \\ \text{VAF}_i &= \left. \begin{aligned} &= \frac{1}{1 + 2(\det(\mathbf{J}) - 1)^2} \\ &= 0 \end{aligned} \right\} \begin{aligned} &\text{VAF}_{\min} < j < \text{VAF}_{\max} \\ &\text{otherwise} \end{aligned} \quad (20.11) \end{aligned}$$

$$\text{GCI} = \frac{\sum_{i=1}^n \text{GCI}_i}{n}$$

$$\text{VAF} = \frac{\sum_{i=1}^n \text{VAF}_i}{n}$$

Table 20.1 The parameters set for the optimization of 1-DOF lambda mechanism.

Parameters	Value	Parameters	Value
optimization dimension	1	Range of parameter	[1, 4]
Number of starts	100	Number of iterations	10
Objective choice	Workspace, GCI, VAF	Velocity amplification range	[0.3, 3]
Workspace (θ_1 range)	45° to 135°	stroke ratio	1.5

In this mechanism, the $l_1(\text{OA})$ length is optimized with respect to $l_2(\text{OB})$, utilizing three distinct objective functions with parameters provided in Table 20.1. Initially, the workspace was maximized to identify an optimal length that covers the revolute joint's range from 45° to 135°. Subsequently,

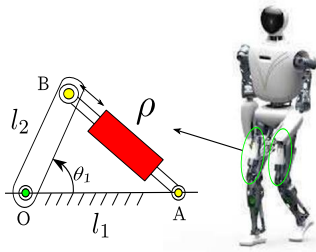


Figure 20.5 1-DOF lambda mechanism with real-life implementation [33].

the GCI and VAF were employed as objective functions. The mechanism's acceptable velocity amplification span ranged from 0.3 to 3. The stroke ratio, the prismatic actuator's fully extended length divided by its unextended length, was $\frac{3}{2}$. To approach a superior global optimum, 100 individual local Nelder–Mead optimization starts were employed, and the number of iterations for the same evaluation within a single start was limited to 10. For all objective functions, multiple solutions with equal evaluation exist. It was observed that $l_1 = 4$ was proposed as the global optimum, while optimizing for all different objective functions. Since the optimization dimension was only 1, this process was swift, completing 100 coarse single starts and 10 refined starts in 21 seconds. The evaluation increases to a specific value (3.39) and remains constant. This value is also the maximum possible evaluation in an ideal scenario. The results for the 1-DOF lambda mechanism optimization are summarized in Table 20.2.

Table 20.2 The results for the optimization of 1-DOF lambda mechanism.

Parameters	GCI	VAF
Time for 1 coarse evaluation	1 second	1 second
Time for single coarse start	0.01 seconds	0.01 seconds
Time for one fine evaluation	5 seconds	3.1 seconds
Time for single fine start	0.04 seconds	0.02 seconds
Best point(l_2)	4	3.4
Best actuator range	[3.37 4.76]	[2.78, 4.17]

20.3.2 2-DOF RCM mechanism

In this section, a popular 2-DOF parallel mechanism, 2UPS-1U, is optimized. This mechanism features a motion constraint generator and can

be considered relatively complex for design optimization. This class of mechanisms has been employed in medical applications [7] as well as in implementing joint modules in humanoids (see [49,50] for application as an ankle joint and [47,51] for application as a torso joint). The first joints in leg 1 and leg 2 with respect to the base can be given as:

$$A_1 = \begin{bmatrix} a_1 \cos \phi_1 \\ a_1 \sin \phi_1 \\ h_1 \end{bmatrix}, A_2 = \begin{bmatrix} a_2 \cos \phi_2 \\ a_2 \sin \phi_2 \\ h_2 \end{bmatrix},$$

where a_i is the distance of the first joint of i th leg from the origin of the base frame and ϕ_1 is the angle between the xy -projection of vector from the origin of the base frame to the joint and the x -axis. Similarly, ϕ_2 is the angle between the xy -projection of vector from the origin of the base frame to the joint and the y -axis. The joints of each leg are at height h_1 and h_2 respectively. The universal joint (U) in the motion constraint generator leg is given as $[0, 0, t]^T$ with respect to the base frame. The spherical joints in each leg are represented with respect to a frame with U as its origin and are given as:

$$B_1 = \begin{bmatrix} b_1 \cos \psi_1 \\ b_1 \sin \psi_1 \\ h_3 + t \end{bmatrix}, B_2 = \begin{bmatrix} b_2 \cos \psi_2 \\ b_2 \sin \psi_2 \\ h_4 + t \end{bmatrix},$$

where b_i and ψ_i are used to express the spherical joints in the legs and have similar interpretation to that of a_i and ϕ_i . The joints of each leg are at height $h_3 + t$ and $h_4 + t$ respectively.

Thus, the mechanism can be parameterized by 13 parameters after, assuming that the motion constraint generator lies on the z -axis of the base. The 13 mechanism parameters to be optimized, as shown in Fig. 20.6 and detailed above are: $[a_1, \phi_1, h_1, b_1, \psi_1, h_2, a_2, \phi_2, h_3, b_2, \psi_2, h_4, t]$. The optimization parameters and the constraints along with their range are shown in Table 20.3.

The computational expense of optimizing this mechanism stems from the increase in optimization space, the number of degrees of freedom, and the considered workspace. A thorough examination of the regular dextrous workspace for the specified mechanism can be found in [52]. Results are contingent upon both the chosen objective and the reward strategy. Table 20.3 presents the outcomes obtained by optimizing the GCI and awarding valid points in the workspace with a value of 1 and invalid points

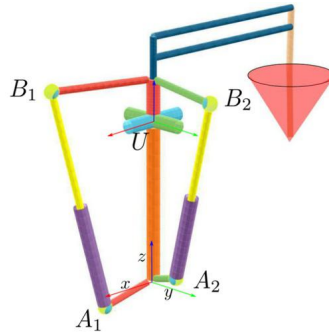


Figure 20.6 The parameters to be optimized in 2UPS-1U.

Table 20.3 The parameters set for the optimization of 2-DOF RCM mechanism.

Parameters	Value	Parameters	Value
optimization dimension	13	Range of a_i	[0.25, 1.5]
Range of b_i	[0.25, 2]	Range of ϕ_i and ψ_i	[-1.745, 1.745]
Range of h_i	[-0.1, 0.1]	Range of t	[1, 4]
Number of starts	200	Number of iterations	10 and 20
Objective choice	Workspace, GCI, VAF	Velocity amplification range	[0.3, 3]
Range of b_i	[0.25, 2]	Range of ϕ_i and ψ_i	[-1.745, 1.745]
Workspace (in roll and pitch)	circle of radius 1	stroke ratio	1.5
limits on spherical joints	$\pm\pi/6$ radians	Collision constraint	considered

with 0. The time necessary for evaluating one instance (a particular set of parameters) and the mean time for a single start (the full operation until the algorithm stops and returns locally optimized parameters) are documented in Table 20.4. Further analysis was conducted to determine the effects of different objectives on the overall optimization time. The fine search process was found to be significantly more time-consuming compared to coarse searches, highlighting the algorithm's efficiency. Table 20.4 contains the results, and the computational time was measured using the same system, intended for comparison purposes only. The schematic plot of the mechanism optimized for maximum GCI, along with the heatmap for GCI evaluation using the optimized parameters, is shown in Fig. 20.7.

Table 20.4 The results for the optimization of 2-DOF RCM mechanism.

Parameters	GCI	VAF
Time for 1 coarse evaluation	14 seconds	18.3 seconds
Time for single coarse start	291 seconds	347.5 seconds
Time for one fine evaluation	50.5 seconds	51 seconds
Time for single fine start	1072 seconds	1077 seconds
Best point $[a_1, \phi_1, h_1, b_1, \psi_1, h_2, a_2, \phi_2, h_3, b_2, \psi_2, h_4, t]$ (refer to Fig. 20.6)	[1.13, -1.02, -0.06, 1.47, -1.01, -0.05, 0.72, 0.44, -0.02, 1.52, 0.54, 0.02, 3.04]	[0.68, -0.25, 0.08, 1.03, 0.1, 0.04, 0.25, -1, 0.01, 1.1, -1.45, 0.17, 2.4]
Best actuator range evaluation	[2.54, 3.8]	[2, 3]
mean	GCI 0.79	VAF 0.48
standard deviation	0.18	0.29
maximum evaluation configuration $([\alpha, \beta])$	1 [0.39, 0.13]	0.99 [0, 0.43]
minimum evaluation configuration $([\alpha, \beta])$	0.318 [0.86, 0.51]	-1.2 [-0.99, 0.14]

Likewise, Fig. 20.8 displays the schematic and heatmap of the quality linked to the VAF for the associated optimized parameters. The schematics shown in both figures indicate that the optimized parameters gravitate towards an architecture with actuated legs separated by $\frac{\pi}{2}$ radians and aligned with the universal joint axes found in the motion constraint generator. This observation implies that human intuition and experience can be employed to decrease the optimization space's dimension, leading to accelerated optimization and more easily manufacturable designs.

20.3.3 Dimension reduction

Human intuition can be implemented to further reduce the optimization space such that the hybrid series-parallel system can be optimized faster and in an efficient manner. The observations presented in the previous section confirm that basic analysis of the mechanism can greatly help in reducing the optimization space. In the case of 2UPS-1U, fixing the z-coordinate of the first universal joint in each leg as zero reduces 2 parameters (h_1, h_2).

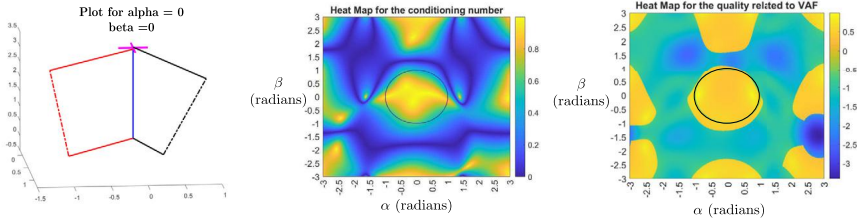


Figure 20.7 The schematic plot for the mechanism optimized for GCI and the heatmap for the evaluation. Calculation of GCI for this mechanism is discussed in [52]. The rightmost subfigure is the heatmap for the VAF quality, corresponding to the same parameters.

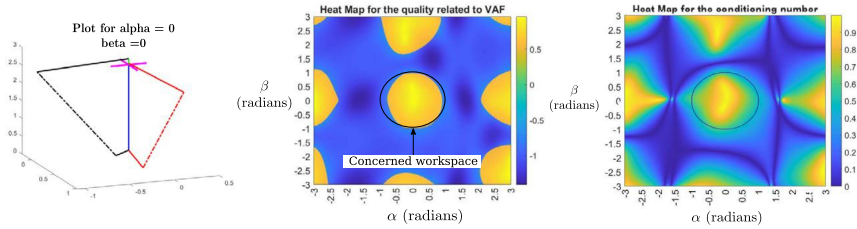


Figure 20.8 The schematic plot for the mechanism optimized for VAF and the heatmap for the evaluation. The rightmost subfigure is the heatmap for the GCI, corresponding to the same parameters.

As we observed that the two legs are optimized when 90° apart, if we fix A_1 along x -axis and A_2 along y -axis, we further reduce two parameters (ϕ_1, ϕ_2) . Similar process is used for B_1 and B_2 , such that $h_3 = h_4 = h$, $\psi_1 = 0$, and $\psi_2 = 0$. In order to make the mechanism modular, the legs can be made symmetrical such that $a_1 = a_2 = a$, and $b_1 = b_2 = b$. This process reduces the 13-parameter space to only 4-dimensional space with a, b, h , and t as the optimization parameters. Such reduction can help optimize mechanisms in cascade and also provide designs that are easier to manufacture and assemble, thus adding practical advantages to the optimization.

20.4 Conclusions

In this chapter, we presented a novel optimization algorithm for parallel manipulators that is able to implement the joint limits and the collision of prismatic joints as constraints. The optimization methodology is also able to optimize the length of the actuator stroke, which enables the designer

greater flexibility and clarity in the choice of the actuators. The Nelder–Mead algorithm uses geometric methods to search for a local optimum, which is relevant for mechanism optimization. The algorithm implements a two-step search by combining a faster local search Nelder–Mead algorithm with initial simplexes spread over all the parameter space and then uses a finer search by using the locally optimized points in the step 1. The algorithm is general and can be adapted to any non-redundant parallel mechanisms with prismatic as well as revolute joint. The paper presents two different mechanism optimization as an example to present the flexibility of the algorithm. It is observed in the design of 2UPS-1U that the optimal solutions correspond to an orthogonal arrangements of the legs. This confirms that human feedback and mechanism knowledge can be used to reduce the dimension of the search space.

When it comes to design optimization of series-parallel hybrid mechanisms, we have only scratched the surface of the problem. A holistic treatment of the design optimization problem for series-parallel hybrid robots would require dealing with a very large dimensional space of design variables, for which one would require more computationally efficient optimization schemes. Additionally, the study presented in this chapter took into account only the kinematic properties of the mechanism. Including the dynamics into account during the co-design process is also a very important avenue for future work. Millions of years of biological evolution has led to the interesting muscle combinations that we witness in animals. It remains an open problem in the robotics community to develop co-design frameworks, which are capable of producing robot designs that have similar athletic performance to their natural counterparts.

References

- [1] B. Lee, C. Knabe, V. Orekhov, D. Hong, Design of a human-like range of motion hip joint for humanoid robots, in: Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Buffalo, New York, USA, 2014, pp. 8–18.
- [2] S. Lohmeier, T. Buschmann, H. Ulbrich, F. Pfeiffer, Modular joint design for performance enhanced humanoid robot LOLA, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, IEEE, Orlando, FL, USA, 2006, pp. 88–93.
- [3] D. Kuehn, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, System design and testing of the hominid robot Charlie, *Journal of Field Robotics* 34 (4) (2017) 666–703.
- [4] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, *Journal of Intelligent & Robotic Systems* 94 (2) (2018) 303–325.

- [5] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E.A. Kirchner, F. Kirchner, Modular design and decentralized control of the Recupera exoskeleton for stroke rehabilitation, *Applied Sciences* 9 (4) (2019).
- [6] J. Arata, H. Kondo, M. Sakaguchi, H. Fujimoto, A haptic device DELTA-4: kinematics and its analysis, in: *Proceedings of World Haptics 2009 – Third Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Salt Lake City, UT, USA, 2009, pp. 452–457.
- [7] G. Michel, D. Salunkhe, D. Chablat, P. Bordure, A new RCM mechanism for an ear and facial surgical application, in: *Proceedings of Advances in Service and Industrial Robotics*, Springer International Publishing, Poitiers, France, 2020, pp. 408–418.
- [8] A. Dutta, D.H. Salunkhe, S. Kumar, A.D. Udai, S.V. Shah, Sensorless full body active compliance in a 6 DOF parallel manipulator, *Robotics and Computer-Integrated Manufacturing* 59 (2019) 278–290.
- [9] A.D. Udai, S.K. Saha, A. Dayal, Overlaid orthogonal force oscillations for robot assisted localization and assembly, *ISME Journal of Mechanics and Design* 2 (1) (2018) 9–25.
- [10] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, *Journal of Mechatronics* 68 (2020).
- [11] J. Brinker, N. Funk, P. Ingenlath, Y. Takeda, B. Corves, Comparative study of serial-parallel Delta robots with full orientation capabilities, *IEEE Robotics and Automation Letters* 2 (2) (2017) 920–926.
- [12] S. Caro, D. Chablat, R. Ur-Rehman, P. Wenger, Multiobjective design optimization of 3-PRR planar parallel manipulators, in: *Proceedings of 20th CIRP Design Conference*, Nantes, France, 2010, pp. 373–383.
- [13] Z. Ma, A.-N. Poo, M.H. Ang, G.-S. Hong, H.-H. See, Design and control of an end-effector for industrial finishing applications, *Robotics and Computer-Integrated Manufacturing* 53 (2018) 240–253.
- [14] P. Wenger, D. Chablat, Kinematic analysis of a new parallel machine tool: the orthoglide, in: *Proceedings of Advances in Robot Kinematics*, Slovenia, 2000, pp. 1–11.
- [15] C. Gosselin, J. Angeles, A global performance index for the kinematic optimization of robotic manipulators, *Journal of Mechanical Design* 113 (3) (1991) 220–226.
- [16] S. Chiu, Kinematic characterization of manipulators: an approach to defining optimality, in: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, USA, 1988, pp. 828–833.
- [17] D. Chablat, P. Wenger, F. Majou, The optimal design of three degree-of-freedom parallel mechanisms for machining applications, in: *Proceedings of 11th International Conference on Advanced Robotics*, 2003, Coimbra, Portugal, 2003, pp. 1–6.
- [18] D. Chablat, P. Wenger, F. Majou, J.-P. Merlet, A novel method for the design of 2-DOF parallel mechanisms for machining applications, *International Journal of Robotics Research* 23 (6) (2007) 615–624.
- [19] R.A. Srivatsan, S. Bandyopadhyay, Determination of the safe working zone of a parallel manipulator, in: *Proceedings of Computational Kinematics*, Dordrecht, Netherlands, 2014, pp. 201–208.
- [20] C. Germain, S. Caro, S. Briot, P. Wenger, Optimal design of the IRSBot-2 based on an optimized test trajectory, in: *Proceedings of 37th Mechanisms and Robotics Conference*, American Society of Mechanical Engineers, Portland, Oregon, USA, 2013, pp. 1–11.
- [21] M.H. Saadatzi, M.T. Masouleh, H.D. Taghirad, C. Gosselin, M. Teshnehlab, Multi-objective scale independent optimization of 3-RPR parallel mechanisms, in: *Proceedings of 13th World Congress in Mechanism and Machine Science*, Guanajuato, Mexico, 2011, pp. 1–11.

- [22] M. Gallant, R. Boudreau, The synthesis of planar parallel manipulators with prismatic joints for an optimal, singularity-free workspace, *Journal of Robotic Systems* 19 (1) (Jan. 2002).
- [23] S. Caro, D. Chablat, A. Goldsztejn, D. Ishii, C. Jermann, A branch and prune algorithm for the computation of generalized aspects of parallel robots, in: *Principles and Practice of Constraint Programming*, Berlin, Heidelberg, 2012, pp. 867–882.
- [24] S. Kucuk, A dexterity comparison for 3-DOF planar parallel manipulators with two kinematic chains using genetic algorithms, *Mechatronics* 19 (6) (2009) 868–877.
- [25] S. Ganesh, A. Koteswara Rao, B. Sarath kumar, Design optimization of a 3-DOF star triangle manipulator for machining applications, *Materials Today: Proceedings* 22 (12) (2020) 1845–1852.
- [26] V. Muralidharan, A. Bose, K. Chatra, S. Bandyopadhyay, Methods for dimensional design of parallel manipulators for optimal dynamic performance over a given safe working zone, *Mechanism and Machine Theory* 147 (2020) 103721.
- [27] S. Ha, S. Coros, A. Alspach, J. Kim, K. Yamane, Computational co-optimization of design parameters and motion trajectories for robotic systems, *The International Journal of Robotics Research* 37 (13–14) (2018) 1521–1536.
- [28] N. Durand, J.-M. Alliot, A combined Nelder–Mead simplex and genetic algorithm, in: *Proceedings of GECCO 1999, Genetic and Evolutionary Computation Conference*, Orlando, FL, USA, 1999, pp. 1–7.
- [29] A. Koscianski, M. Luersen, Globalization and parallelization of Nelder–Mead and Powell optimization methods, in: *Proceedings of Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, Dordrecht, Netherlands, 2008, pp. 93–98.
- [30] M.A. Luersen, R. Le Riche, Globalized Nelder–Mead method for engineering optimization, *Computers & Structures* 82 (23–26) (2004) 2251–2260.
- [31] P. Niegodajew, M. Marek, W. Elsner, L. Kowalczyk, Power plant optimisation—effective use of the Nelder–Mead approach, *Processes* 8 (3) (2020) 357.
- [32] K.H. Hunt, Review: don't cross-thread the screw!, *Journal of Robotic Systems* 20 (7) (2003) 317–339.
- [33] D.H. Salunkhe, G. Michel, S. Kumar, M. Sanguineti, D. Chablat, An efficient combined local and global search strategy for optimization of parallel kinematic mechanisms with joint limits and collision constraints, *Mechanism and Machine Theory* 173 (2022) 104796.
- [34] J.A. Nelder, R. Mead, A simplex method for function minimization, *The Computer Journal* 7 (4) (1965) 308–313.
- [35] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, Convergence properties of the Nelder–Mead simplex method in low dimensions, *SIAM Journal on Optimization* 9 (1998) 7.
- [36] K.I.M. McKinnon, Convergence of the Nelder–Mead simplex method to a nonstationary point, *SIAM Journal on Optimization* 9 (1) (1998) 148–158.
- [37] P.C. Wang, T.E. Shoup, Parameter sensitivity study of the Nelder–Mead simplex method, *Advances in Engineering Software* 42 (7) (2011) 529–533.
- [38] D. Byatt, *Convergent variants of the Nelder–Mead algorithm.pdf*, Ph.D. thesis, University of Canterbury, England, 2000.
- [39] S. Zapotecas Martínez, C.A. Coello Coello, A proposal to hybridize multi-objective evolutionary algorithms with non-gradient mathematical programming techniques, in: *Proceedings of Parallel Problem Solving from Nature – PPSN X*, Berlin, Heidelberg, 2008, pp. 837–846.
- [40] H. Niederreiter, Random number generation and quasi-Monte Carlo methods, in: *Proceedings of CBMS-NSF Regional Conference Series in Applied Mathematics*, vol. 63, Philadelphia, Pennsylvania, 1992, pp. 1–243.

- [41] K.-T. Fang, Y. Wang, *Number-Theoretic Methods in Statistics*, Chapman & Hall, London, 1994.
- [42] A. Alessandri, C. Cervellera, D. Macciò, M. Sanguineti, Optimization based on quasi-Monte Carlo sampling to design state estimators for nonlinear systems, *Journal of Optimization* 59 (2010) 963–984.
- [43] J.M. Hammersley, D.C. Handscomb, *Monte Carlo Methods*, Methuen, London, 1964.
- [44] A. Alessandri, C. Cervellera, M. Sanguineti, Design of asymptotic estimators: an approach based on neural networks and nonlinear programming, *IEEE Transactions on Neural Networks* 18 (1) (2007) 86–96.
- [45] I.M. Sobol', The distribution of points in a cube and the approximate evaluation of integrals, *Zhurnal Vychislitelnoi Matematiki I Matematicheskoi Fiziki* 7 (1967) 784–802.
- [46] S. Bartsch, M. Manz, P. Kampmann, A. Dettmann, H. Hanff, M. Langosz, K. von Szadkowski, J. Hilljegerdes, M. Simnofske, P. Kloss, M. Meder, F. Kirchner, Development and control of the multi-legged robot Mantis, in: *Proceedings of ISR 2016: 47th International Symposium on Robotics*, Munich, Germany, 2016, pp. 1–8.
- [47] J. Esser, S. Kumar, H. Peters, V. Bargsten, J.d.G. Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid RH5 humanoid robot, in: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021, Munich, Germany, 2021, pp. 400–407.
- [48] S. Kumar, *Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots*, Ph.D. thesis, University of Bremen, DFKI-RIC, Bremen, Germany, Sep. 2019.
- [49] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, in: *Proceedings of Advances in Robot Kinematics*, Bologna, Italy, 2018, pp. 431–439.
- [50] C. Stoeffler, S. Kumar, H. Peters, O. Bröls, A. Müller, F. Kirchner, Conceptual design of a variable stiffness mechanism in a humanoid ankle using parallel redundant actuation, in: *Proceedings of 18th International Conference on Humanoid Robots (Humanoids)*, Beijing, China, 2018, pp. 462–468.
- [51] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing RH5 Manus: a powerful humanoid upper body design for dynamic movements, in: *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 01–07.
- [52] D. Chablat, G. Michel, P. Bordure, S. Venkateswaran, R. Jha, Workspace analysis in the design parameter space of a 2-DOF spherical parallel mechanism for a prescribed workspace: application to the otologic surgery, *Mechanism and Machine Theory* 157 (Mar. 2021) 104224.

Index

A

- Abstract
 - class, 409
 - joint space, 36, 38, 39
- Abstracted
 - mechanism, 416
 - serial robot, 418
- Abstraction serial, 199, 214, 215, 223, 224
- Acceleration level explicit constraint, 151
- Acceleration-based WBC, 202
- Active foot support, 249
- Actuated
 - ankle joint, 270
 - joints, 199, 200, 203, 208, 217, 347, 404
 - legs, 471
 - prismatic joints, 61, 430
 - universal joint, 430
- Actuation, 337
 - space, 38, 39, 53, 159, 198, 199, 215, 219, 220, 341, 416, 418, 422
 - Jacobian, 200
 - torques, 208
- Actuator, 308
 - ankle, 204
 - base, 269
 - configuration, 53, 54
 - constraints, 79
 - design, 399
 - drivers, 275
 - forces, 39, 85, 142, 155, 156, 170, 202, 419
 - input, 38, 53
 - Jacobian, 200
 - Jacobian matrix, 170, 175
 - legs, 338
 - lengths, 74, 76–78, 366, 368, 455
 - level, 316
 - level control architecture, 323
 - level modularity, 323
 - limits, 88, 201, 208
 - linear, 61, 70, 73, 198, 220, 238, 265, 267, 314, 335, 338
 - modules, 31, 96
 - placement possibilities, 159
 - positions, 219, 269, 323, 365, 366, 368, 419, 421
 - range, 455
 - robot, 195
 - rotary, 269, 336, 337
 - rotational, 338
 - selection, 454
 - setup, 340
 - space, 18, 370, 415, 417, 418, 421
 - specification, 81
 - speeds, 83
 - spine, 246
 - states, 26
 - status, 414
 - stroke, 472
 - torques, 412
 - trajectories, 414, 418
 - units, 335, 338
 - valve, 357
 - velocities, 200, 206, 366
- Actuator Control Unit (ACU), 314, 316
- Adaptive
 - footplacement algorithm, 256
 - footstep planner, 253
 - footstep planning, 252, 253
- Adjoint mapping, 131
- Admissible
 - actuator range, 198
 - velocity range, 206
 - workspace, 203, 210
- Adopted practices, 37
- Agile
 - robots, 333
 - wrist mechanism, 20
- Algebraic
 - constraint equations, 57
 - formulation, 64
 - geometry, 55
 - variety, 55
- Almost spherical parallel mechanism (ASPM), 314
- Almost-spherical parallel manipulator (ASPM), 94

- Analytic
 - formulation, 63
 - variety, 55
 - Analytical
 - resolution, 156
 - solutions, 409
 - Ankle, 72
 - actuators, 204
 - admittance
 - control, 346
 - controller, 346
 - method, 348
 - axes, 337
 - design, 70, 86, 88, 89
 - drives, 314
 - humanoid, 79
 - joint, 18, 70, 76, 94, 99, 161, 164, 204, 238, 239, 266–268, 314, 322, 337, 401, 469
 - controller, 245
 - pitch, 246
 - level, 348
 - mechanism, 79, 80, 197, 204, 269
 - module, 25, 322
 - pitch, 337
 - joint, 204
 - movement, 70
 - position, 249
 - stabilization method, 346
 - submechanism, 407
 - workspace, 89
 - Annotated submechanism, 407
 - Anthropomorphic robot, 333
 - Application Programming Interface (API), 277
 - Application scenarios, 235, 264, 283, 306, 333, 358
 - Arbitrary parallel submechanisms, 166
 - Argos mechanism, 94
 - Arm
 - exoskeleton, 323
 - joints, 206
 - Articulated body algorithm (ABA), 180, 182, 185
 - Articulated body inertia, 180, 185, 187
 - calculation, 181
 - Artificial intelligence (AI), 359
 - Assistive
 - controllers, 289
 - robotic rehabilitation, 330
 - Athletic bipedal robots, 19
 - Automated
 - control, 357
 - robot analysis, 421
 - Automatic
 - grading control, 363
 - ground adaption, 375
 - WBC controllers, 347
 - Autonomous control, 285, 362, 364, 365
 - Autonomous Rough Terrain Excavator Robot (ARTER), 357, 359, 360, 362
 - Autonomy modules, 289
- B**
- Backward recursion, 138, 140
 - Ball screw, 266, 309, 314, 336–338
 - Bare metal systems, 31
 - Base frame, 73, 74, 125, 128, 137, 140, 469
 - robot, 243
 - Basic operation mode, 323
 - BeagleBone (BB), 429
 - Behavior modules, 247, 255
 - Bio-inspired robot, 442
 - Biological inspiration, 233, 234, 262, 263, 281, 282, 357, 358
 - Biped robot, 342
 - Bipedal
 - locomotion, 240
 - pose, 237, 243, 254
 - robot, 18, 19
 - BLDC motor, 266, 269, 310, 311, 314, 316, 335, 336
 - Blind locomotion, 250
 - Board connecting sensors, 339
 - Body Height Control (BHC), 291, 294, 295
 - Boost-Graph-Library (BGL), 251
 - Bottom-up
 - composition, 163
 - modeling, 407
 - Box constraints, 197–199, 201, 204, 214, 219, 220

C

- Cable driven parallel mechanism, 19
- Cartesian
 - positioning task, 201, 203, 205, 208
 - space control, 36
- Cascade optimization, 463
- Cascaded
 - controller, 245
 - position, 97, 322
- Center of Gravity (CoG), 291
- Center of mass (COM), 133, 138, 263
- Center of Pressure (CoP), 342
- Central
 - control, 245
 - control PC, 340
 - controllers, 32, 339
 - electronics, 243, 316, 340
 - module, 444
- Central Pattern Generator (CPG), 247, 273
- Central Power Management Board (CPMB), 318
- Centralized control
 - loops, 338
 - systems, 26
- Charlie
 - foot, 239
 - hominid system, 236
 - robot, 235, 236, 272
 - robustness, 256
 - unique mechatronic subsystems, 254
 - walking pattern, 253
- Chasles' Theorem, 58
- Circular movement, 312
- Classical formulation, 133
- Closed loop, 146, 185
 - mechanisms, 217
- Coarse search, 463
- Collision constraints, 456
- Command line tools, 388
- Commanded
 - actuator forces, 420, 421
 - actuator position, 420
 - joints, 346
 - torque, 417
- Communication hardware, 32
- Complex robots, 383
- Comprehensive kinematic analysis, 322
- Computational
 - complexity, 141
 - effort, 174
 - performance, 208, 413
 - timings, 226
- Computer-Aided Design (CAD), 33, 381
- Computerized control, 361
- Computing nodes, 26
- Configurable
 - submechanism libraries, 399
- Connectivity graph, 160
- Constrained multi-body dynamics, 215
- Constraints, 454
 - actuator, 79
 - embedding, 183
 - embedding approach validation, 191
 - equations, 72, 100
 - implementation, 456
 - Jacobian, 78
 - kinematic, 51, 153, 213–216
 - loop closure, 35, 38, 52, 60, 150, 151, 180, 183, 200, 210, 213, 214, 402
 - position, 204, 205, 405
 - torque, 207
 - velocity, 205
- Contact
 - constraints stability, 342
 - friction cone, 344
 - Jacobian, 200
- Contact-based state estimation, 250
- Continuous integration (CI), 320, 389, 390
- Control
 - algorithms, 30, 266
 - architecture, 26, 243, 246, 274, 319, 322, 402, 447
 - central, 245
 - cycle, 195
 - design, 307
 - electronics, 272, 288
 - engineer, 398, 399, 401
 - exoskeleton, 322
 - FPGAs, 272
 - hydraulic, 374
 - joint, 222, 245, 375
 - joint space, 36, 416
 - law, 201, 419, 440, 442
 - level, 339
 - locomotion, 273, 376

- loop, 246, 272, 322, 338, 351, 438, 440, 442
 - methods, 36
 - mode, 323, 417
 - motion, 242, 247, 250, 419
 - objectives, 292
 - optimal, 36, 214, 217, 341, 346, 348
 - pitch, 293, 297
 - position, 277, 324
 - purposes, 365, 401
 - regularization, 222
 - robot, 217
 - robotic, 28
 - scheme, 347, 361
 - signal, 201
 - strategies, 28, 36
 - structure, 361, 365, 374
 - systems, 26, 227, 300, 427
 - theory, 398
 - torque, 203, 322, 323, 343, 399, 401
 - unit, 255, 447
 - valves, 375
 - Controller
 - central, 32, 339
 - designs, 376
 - execution, 347
 - joints, 375
 - local, 38, 339
 - outputs, 246
 - Cooperating robots, 301
 - Cost model definition, 343
 - Counter-clockwise (CCW) circular trajectory, 438
 - Coupler angle, 367
 - Crank points, 101
 - Cross product, 131
 - Custom
 - control software stacks, 197
 - hardware accelerator, 30
 - Cylindrical algebraic decomposition (CAD), 435
- D**
- Data Distribution Service (DDS), 32
 - Decentralized
 - actuator-level controllers, 316
 - control, 339
 - control architecture, 243
 - control units, 307
 - Decoupled Natural Orthogonal Compliment (DeNOC) formulation, 35
 - Degree of Freedom (DoF), 94, 237, 238, 265, 282–285, 308
 - ankle joint, 337
 - humanoid robot, 399
 - joint modules, 20
 - prismatic joint, 52, 322
 - revolute joint, 52
 - robot, 52
 - swivel joint, 314
 - wrists, 21, 336
 - Delta robot, 21
 - Denavit–Hartenberg (DH) parameters, 33
 - Design
 - comparison, 98
 - considerations, 452
 - features, 97
 - limitations, 257, 300, 375
 - motivation, 334
 - Developer workflow, 399
 - Differential dynamic programming (DDP), 10
 - Differential Dynamic Programming (DDP), 205, 215, 341
 - Differential Evolution (DE), 451
 - Dimension reduction, 471
 - Direct kinematics problem (DKP), 75
 - Discretized workspace, 456
 - Disjoint union, 162
 - Distributive aspects, 21
 - Double parallelogram mechanism, 20
 - Downhill-simplex algorithm, 457
 - Dynamics, 34, 38, 166, 320
 - constraints, 227
 - forward, 155, 179, 182, 185, 412, 413
 - inverse, 137, 141, 155, 156, 398, 412, 416
 - robot, 34, 132, 198, 210, 217
- E**
- ECCE humanoid robot, 29
 - Elbow, 220
 - actuator velocity, 206
 - adapters, 315

- drive, 311, 337
 - joint, 207
 - joint velocity, 207
 - mechanism, 198, 206, 220
 - positions, 306
 - rotation, 219
 - torques, 223
 - velocity, 206
 - Electric control lines, 362
 - Electrical actuators, 428
 - Electromechanical Interface (EMI), 284, 285
 - Electronic
 - control, 360, 362
 - controllers, 29
 - design, 287, 338, 362
 - Electronically controllable valves, 362
 - Electronics, 26
 - Electronics design, 241, 271
 - Elevation and Depression Reflex (EDR), 248
 - Emergent robot behavior, 247
 - End-effector (EE)
 - frame, 73
 - points, 100
 - position, 111, 113
 - Equation of Motion (EOM), 141, 152–154, 170, 341
 - Evaluation function, 456
 - Excavator
 - joints, 365
 - robot, 300
 - Exceptional generator, 56
 - Exoskeleton, 20
 - active, 306
 - arms, 323
 - Bleex, 100
 - central electronic system, 316
 - central processing system, 316
 - classification, 328
 - configurations, 319
 - control, 322
 - design, 330
 - development, 305
 - dynamic model, 307
 - hand interfaces, 324
 - operation mode, 324
 - Recupera, 100, 306, 308, 315, 322, 417
 - safe operation, 318
 - systems, 328
 - weight, 98
 - Experimental
 - design, 217
 - results, 203, 226
 - Explicit loop constraints, 154
 - Exponential coordinates, 59
 - Extension movement, 72
 - Exteroceptive sensors, 243, 254
- ## F
- Feasible actuator range, 454
 - Feedforward
 - control, 340
 - torque, 347
 - Field Programmable Gate Array (FPGA), 287
 - Field Programmable Grid Array (FPGA), 339
 - File structure consistency, 393
 - Fine search, 464
 - First level
 - control, 322
 - processing, 271
 - First optimization problem, 442
 - Flexible robot assembly, 445
 - Flexion, 72
 - Floating base, 200, 202, 203, 208, 341
 - accelerations, 202
 - coordinates, 416
 - joints, 202
 - robots, 415, 416
 - velocities, 200
 - Floating kinematic tree, 181
 - Floating Point Unit (FPU), 38
 - Foot
 - Charlie, 239
 - contact
 - area, 337
 - points, 252, 256, 337
 - surface, 242
 - controller, 246
 - design, 249
 - frame, 346

- human, 238, 315
- joint, 238
- link, 197
- mechanical, 239
- mechanism, 315
- passive, 269
- pedal, 319
- pitch, 250
 - behavior, 250
 - modulation, 250
 - module, 250
- placement, 255, 344
- points, 250
- sensors, 257
- surface, 269
- trajectories, 247, 273
- Footprint, 252
- Footprint command, 290
- Force Fuse module, 295
- Force Leveling Control (FLC), 290–292
 - module, 293, 294
- Force-Torque Sensor (FTS), 286
- Forearms, 266
 - Mantis, 266
 - mechanism, 311
- Forward
 - dynamics, 179, 182, 185, 412
 - function, 413
 - problem, 155
 - solution, 155
 - kinematic
 - Jacobian matrix, 85
 - mapping, 54
 - problem, 54
 - kinematics, 25, 26, 34, 36, 37, 76, 77, 124, 126, 161, 289, 368, 398
 - motion, 248–250
 - recursion, 138, 139
- Forward kinematics problem (FKP), 103
- Free
 - flyer joint, 416
 - joint, 26, 159
 - movement, 330
 - positioning, 283
- Full hybrid model, 223, 224
- Fuzzy control, 36

G

- Gait motion, 347
- General mobility, 97
- Generalized Iterative Closest Point (GICP)
 - algorithm, 251
- Generic
 - parallel submechanism, 403
 - submechanisms, 410
- Genetic Algorithm (GA), 451
- Geometric analysis, 63
- Geometric information, 382
- Geometry of motion, 34
- Global
 - kinematics, 57
 - search algorithm, 462
- Global conditioning index (GCI), 454, 467–469
- Global Positioning System (GPS), 296
- Graph based topological description, 121
- Gravity Compensation (GC), 323, 326
 - control, 417
 - mode, 323, 326, 330, 417
 - model, 326, 417
- Gröbner bases, 34, 57, 75, 77, 166
- Ground
 - adaptation, 282
 - adaption, 282, 293, 294
 - clearance, 294
 - contact, 240, 249, 265, 267, 282, 286, 290, 291, 372
 - detection, 372
 - force, 291, 292
 - force errors, 297
 - points, 295
 - control station, 297
 - detection, 248
 - interaction, 277
 - unevenness, 269
- Ground Adaption Process (GAP), 290, 293, 294
- Ground Plane Estimator module, 295

H

- Hall sensors, 242
- Hand, 336
- Hardware, 22
 - acceleration, 29
 - components, 272

- drivers, 271
 - joint limits, 392
 - locomotive system, 300
 - modules, 32
 - robotic, 384
 - Harmony exoskeleton, 312
 - Head, 336
 - controller, 247, 274
 - joints, 338
 - HEAP robot, 356
 - Heterogeneous robots, 235, 276
 - Hexapod walking robots, 270
 - High-level control, 275, 296, 324, 372
 - Hip
 - joint, 257, 337, 338
 - joint connection, 314
 - pitch, 337
 - Holonomic constraints, 215
 - Hominid robot Charlie, 19, 234
 - Human
 - arm, 323, 326
 - arm model, 326
 - foot, 238, 315
 - joints, 20, 305
 - motion ranges, 72
 - movement, 328
 - Humanoid, 18
 - ankle, 79
 - ankle joint, 79
 - leg, 164, 166, 400, 401, 407, 411, 419
 - robot, 5, 18, 70, 161, 195, 197, 213, 306, 308, 333, 451
 - robotics, 18
 - wrist applications, 72
 - Hybrid
 - control approach, 338
 - mechanism, 175
 - robot, 5, 19, 22, 26, 159, 162, 163, 226
 - design serial nature, 161
 - models, 217
 - Hybrid Robot Dynamics (HyRoDyn), 157, 198, 226, 382, 389, 399, 401
 - component, 416–418
 - computational performance, 413
 - developer, 399
 - library, 198, 200, 210, 401, 411
 - orogen component, 415, 417, 418
 - software, 410
 - architecture, 398
 - framework, 157, 424
 - library, 408
 - Hydraulic
 - actuator, 365
 - actuators, 362
 - pilot control valves, 375
 - Hydraulics control, 363
- ## I
- Implementation scheme, 347
 - Implicit loop constraints, 153
 - Inclination, 74
 - Independent
 - coordinates, 161, 167, 174, 187, 200, 206, 219
 - joint, 161, 168, 191, 196, 199, 208, 217, 220, 223, 404
 - selection matrix, 411
 - space, 199, 202, 204, 206, 219, 411
 - status, 414
 - variables, 154
 - position variables, 154
 - velocity coordinates, 199
 - Industrial
 - automation, 20
 - manipulator, 5
 - robotics, 32
 - robots, 5, 21, 233
 - Inertia Measurement Unit (IMU), 243, 250, 293
 - Input motion, 419
 - Inspection robots, 427, 428, 442
 - Instantaneous
 - joint screws, 128
 - screw coordinates, 128, 130
 - Intelligent control framework, 262
 - Intersecting revolute joint axes, 88
 - Intra-system, 26
 - Inverse
 - dynamics, 137, 141, 155, 156, 398, 412, 416
 - analysis, 157
 - output, 413

- problems, 155, 156
 - solution, 156
 - kinematic
 - function, 368
 - Jacobian, 83
 - Jacobian matrix, 83
 - mapping, 54
 - model, 112, 113
 - module, 290, 371, 372
 - kinematics, 32, 38, 57, 74, 77, 103, 161, 196, 247, 249, 272, 273, 288, 322, 368, 398, 401, 415, 434
 - parallel kinematics, 368
 - Inverse kinematics problem (IKP), 38, 55, 65, 75, 78, 103, 108, 431
 - Isotropic configuration, 453
- J**
- Jacobian, 199, 200, 418
 - actuation space, 200
 - actuators, 200
 - condition number, 79, 422
 - determinant, 467
 - inverse kinematic, 83
 - loop closure, 170, 200, 202
 - robot, 199
 - Jacobian matrix, 80, 130, 150, 160, 184, 187, 216, 451, 453, 454
 - inverse kinematic, 83
 - kinematic, 79–81, 422
 - loop closure, 174
 - Joint
 - abstractions, 72, 90
 - accelerations, 155, 191, 208, 347, 412
 - actuated, 200, 203, 208, 217, 347, 404
 - angles, 55, 63, 102, 103, 105, 248, 250, 335, 366, 368, 371, 372
 - axes, 61, 90, 94, 138, 148, 305, 319, 336, 337
 - commanded, 346
 - configuration, 36, 220
 - configuration space, 223
 - control, 222, 245, 375
 - controllers, 375
 - data, 365, 374
 - designs, 78
 - elements, 384
 - encoders, 366
 - floating base, 202
 - forces, 79, 174
 - frame, 61
 - friction, 39
 - human, 20, 305
 - independent, 161, 168, 191, 196, 199, 208, 217, 220, 223, 404
 - kinematic, 58, 338, 400
 - limit, 221, 392, 413, 422, 435, 454, 472
 - limit avoidance, 201, 203, 204
 - linear, 305
 - locations, 74
 - modules, 30
 - motion, 61, 391
 - movements, 20
 - orientational, 22
 - parallel, 38
 - passive, 26, 79, 453
 - position, 141, 152, 191, 226, 290, 309, 340, 418
 - position limits, 201, 205
 - position vector, 321
 - positioning, 201, 203
 - robot, 31, 200, 202
 - rotary, 416
 - rotational, 241, 314
 - screw, 128
 - screw coordinate vector, 181
 - selection matrix, 411
 - serial, 288
 - space, 174, 179, 198, 199, 202, 220, 326, 340, 369, 370, 413–415
 - command output, 416
 - configuration, 53
 - control, 36, 416
 - input, 416
 - trajectories, 418
 - velocities, 199, 210, 418
 - specification, 86
 - spine, 245
 - status, 416
 - stiffness, 334, 338
 - torque, 140, 245, 330, 341, 372, 393
 - control, 344
 - limits, 223
 - torso, 18, 26, 69, 86, 159, 272, 469

- trajectories, 344
 - variable, 75, 163
 - velocities, 126, 199, 208, 334, 371, 411
 - wrist, 19, 77, 320, 338
- K**
- Kinematic
 - actuation principle, 70
 - analysis, 66, 89, 398–400
 - chain, 57, 123, 124, 126, 127, 183, 185, 189, 404, 405, 407
 - acceleration, 130
 - position, 124
 - velocity, 126
 - complexity, 213
 - constraints, 51, 153, 213–216
 - definitions, 384
 - designs, 381
 - image points, 56
 - image space, 56
 - integrity, 393
 - Jacobian matrix, 79–81, 422
 - joints, 58, 338, 400
 - links, 385
 - loop, 61, 146, 148, 154, 214, 216, 217, 381
 - loop closure, 216
 - mapping, 55, 214
 - model, 196, 381, 407
 - modeling, 245, 307, 319
 - motion, 51
 - pairs, 61
 - properties, 381
 - robot, 389
 - setup, 305
 - suspension system design, 285
 - topology, 122
 - tree, 138, 180, 181, 183, 185, 360, 381, 384, 385
 - Kinematics, 34, 38, 166, 320
 - Kinematics errors, 35
 - Knee
 - joint, 18, 164, 314, 337, 401, 411
 - submechanism modules, 413
 - Kutzbach–Grübler
 - criteria, 52, 62, 72, 74
 - formula, 95, 97
- L**
- Lambda mechanism, 61, 66, 147, 150, 189, 336, 337, 466
 - Leg, 337
 - actuated, 471
 - actuators, 338
 - kinematics, 284
 - mechanisms, 428, 429, 442, 444, 445, 447
 - movements, 247, 273
 - Recupera, 100
 - submechanisms, 322
 - Leg End Point (LEP), 286, 288, 291, 292
 - command generator, 290
 - interpolator, 290
 - Legged
 - locomotion, 19, 358
 - robots, 25, 233, 467
 - Levenberg–Marquardt (LM), 113
 - Lie group
 - concepts, 184, 193
 - formulations, 191
 - methods, 57, 136
 - theory, 124, 132, 143
 - Lifting motion, 227
 - Liftoff pitch angle, 250
 - Lightweight
 - robot remote control, 277
 - robotic systems, 5
 - Limb
 - controllers, 247, 274
 - design, 237
 - joints, 25
 - Linear
 - actuator, 61, 70, 73, 198, 220, 238, 265, 267, 314, 335, 338
 - actuator forces, 224, 314
 - joints, 305
 - velocity, 133, 135
 - Link
 - mass consistency, 393
 - mass symmetry, 393
 - transformation consistency, 393
 - transformation symmetry, 393
 - Lithium Polymer (LiPo), 244
 - Load transfer devices, 100

Local

- analysis, 61
- control loops, 272, 339
- controllers, 38, 339
- foot controller, 246
- ground features, 270
- kinematics, 398
- search algorithm, 457

Locomotion

- abilities, 262
- active foot support, 249
- behaviors, 276
- bipedal, 240
- capabilities, 235, 250
- conditions, 428
- control, 273, 286, 376
- legged, 19, 358
- modes, 234, 264, 358, 376
- pattern, 250
- principles, 427

Locomotive system's hardware, 300

Look Up Table (LUT), 38

Loop closure, 200, 217, 415

- condition, 149
- constraints, 35, 38, 52, 60, 150, 151, 180, 183, 200, 210, 213, 214, 398, 402
- equations, 159
- errors, 155
- Jacobian, 170, 200, 202
- Jacobian matrix, 151, 174
- kinematic, 216

Loop closure function (LCF), 52, 75, 154, 320, 399, 401

Loop constraints, 147, 153, 154, 403, 404

Low-discrepancy sequences, 462

Low-level

- control, 245, 272, 289, 369
- processing, 271

M

Manipulation control, 274

Manipulator

- architecture, 73, 88
- arm, 335
- base, 430
- industrial, 5

pose, 373

quality, 453

workspace, 453

Mantis

- ankle joint, 269
- development, 262
- features, 267
- forearms, 266
- mechanical structure, 265
- prototype, 275
- robot, 272
- software stack, 271

Manual pitch control, 295

Mars Exploration Rover (MER), 282

Mars Science Laboratory (MSL), 282

Mass matrix factorization, 188

Matrix

- exponential, 59
- logarithm, 60
- logarithm maps, 59

Maximum

- output velocity, 84
- pitch velocity, 82
- roll velocity, 83, 84
- torque, 207, 220
- velocity, 83, 207, 220

Mean absolute error (MAE), 326, 417

Mechanical

- design, 236, 265, 284, 308, 315, 334, 359, 360
- foot, 239
- joints, 407
- joints rotational, 305

Mechanics, 22

Mechanism

- ankle, 79, 80, 204, 269
- architecture, 100
- behavior, 61
- constraint equations, 75
- definition, 407
- design, 401, 452, 457, 461
- design description, 94
- design optimization, 452
- elbow, 198, 206, 220
- foot, 315
- hybrid, 175
- Jacobian, 86

- modular, 402, 472
- optimization, 452, 453, 457, 473
- orientation, 455
- parallel kinematic, 336
- passive, 270
- serial, 29, 53, 381, 400, 409
- synthesis, 451
- theory, 314
- torso, 74
- workspace, 80
- wrist, 71, 72, 74, 77, 86, 219
- Mechatronic system design, 236, 265, 283, 308, 334, 359
- Microcontrollers, 29, 38, 249, 271, 446
- Mid-level control, 273, 289, 323, 370
- Mimic joint
 - concept, 417
 - definition, 320
- Minimal loop cluster, 160
- Minimum position, 436
- Mirror (M) mode, 323, 326
- Mission control, 236, 372
- Mobile
 - robotic systems, 233
 - robotics, 356
 - robots, 241, 281, 355
- Mobility
 - analysis, 62, 116
 - characteristics, 252
 - concepts, 235
 - features, 100
 - robot, 254
- Model
 - deployment, 393
 - formats, 383
 - predictive control, 257
- Model predictive control (MPC), 372
- Modeling rigid-body systems, 146
- Modular
 - approach, 159, 175, 383, 401
 - architecture, 275
 - aspects, 21
 - central pattern generator, 247
 - choice, 170
 - composition, 403
 - concept, 307
 - description, 33
 - design, 19, 94
 - distributed systems, 31
 - form, 174
 - formulation, 175
 - graph, 164, 403
 - graph enumeration scheme, 162
 - hybrid robotic systems, 31
 - mechanism, 402, 472
 - methods, 399
 - numbering scheme, 164
 - parallel joint concept, 18
 - robot, 30, 31
 - robot description models, 319
 - robotic system, 386
 - software workbench, 399
 - spherical unit, 100
 - structure, 319
 - submechanism definitions, 409
 - system, 308
 - way, 415
- Modularity, 28, 31, 159, 272, 283, 285
 - level, 29
 - notion, 21, 159
- Module
 - ankle, 25
 - central, 444
 - foot pitch, 250
 - inverse kinematic, 290, 371, 372
 - submechanism, 26, 38, 159–163, 166, 401
- Monte Carlo sampling, 462
- Motion
 - capabilities, 342
 - commands, 252
 - constraint, 154, 382
 - constraint generator, 468, 469, 471
 - constraint generator leg, 70, 469
 - control, 242, 247, 250, 419
 - controller, 249, 252, 253, 275
 - forward, 248–250
 - generation, 341
 - geometry, 34
 - joint, 61, 391
 - kinematic, 51
 - parameter, 52, 73, 97
 - pitch, 238
 - planner, 251

planning, 213, 255
 quality, 453
 range, 70
 robot, 236
 roll, 238, 335, 336
 rotary, 66
 rotational, 25, 56, 133
 screw, 59, 61, 124
 space, 306
 subspace, 26, 159
 tracking, 420
 tracking system, 251
 trajectories, 414, 452
 variables, 147
 Motion Control System (MCS), 289
 Motor torques, 137, 439–442
 Movement
 free, 330
 human, 328
 pitch, 18, 70, 80, 197, 204, 337
 range, 248–250
 roll, 70, 83–85
 rotational, 52
 space, 238, 308
 walking, 267
 MPC controller, 351
 Multi Layer Surface (MLS), 252
 Multi-body system (MBS), 398
 Multi-legged robots, 19
 Multi-Robot System (MRS), 282, 284
 Multimodal sensors, 359
 Muscle torques, 330

N
 Navigation purposes, 251, 285
 Networked control system, 26
 Newton–Euler equations, 132, 134
 Nodes, 31
 Non-smoothness, 52
 Nonlinear dynamics, 36, 37
 Normalized Energy Stability Margin
 (NESM), 372
 Numbering scheme, 122, 146, 160, 162,
 180, 183
 for topological graphs, 122
 modular, 164

Numerical
 decomposition techniques, 154
 solutions, 410

O
 Object request broker (ORB), 324, 414
 Objective function, 453
 Online
 control, 351
 controller, 345
 Operational grant (OG), 266
 Opposing movement, 266
 Optimal
 actuator ranges, 455
 controller, 214
 design, 442
 design parameters identification, 430
 Optimal Control (OC), 36, 214, 217, 341,
 346, 348
 formulation, 221
 motion, 344
 Optimal Control Problem (OCP), 216,
 222, 341, 342
 Orientation
 mechanism, 455
 parallel mechanisms, 69
 workspace, 79, 80, 89
 Orientational
 joints, 22
 parallel mechanism, 272
 Orientational parallel mechanism (OPM), 8
 Outdoor mobile robots, 374

P
 Parallel
 ankle mechanism, 197
 joints, 38
 kinematic
 machine, 313
 mechanism, 336
 kinematics, 267, 305, 308, 334, 337, 374
 kinematics mechanisms, 272
 manipulators, 25, 74, 79, 430, 434, 453,
 472
 mechanisms, 5, 17, 18, 60, 69, 79, 161,
 174, 196, 197, 213, 215, 245, 269,
 334, 335, 381, 383, 400, 403, 461,
 466

- actuator constraints, 204
- actuator limits, 199
- inverse kinematics, 161
- robots, 4, 26, 34–36, 52, 54, 166, 213, 437
- submechanism, 26, 33, 166, 184, 196–198, 402
- modules, 22, 38, 159, 161, 163, 320, 403, 422
- unnecessary dynamics, 198
- Parallel kinematic manipulator (PKM), 5, 451
 - design constraints, 457
 - modules, 18, 19, 399, 400
 - optimization, 452
 - submechanism module, 410
- Parallelogram mechanism, 5, 20, 21
- Parasitic motion, 109
- Passive
 - foot, 269
 - joint, 26, 79, 453
 - angles, 63, 74
 - limits, 76, 453, 456
 - kinematics, 314
 - mechanism, 270
 - rotational joints, 197
 - sensors, 236
 - spherical joints, 109, 197
- Path planning, 252
- PID controllers, 344, 346, 347
- Peak torque, 70, 338
- Performance analysis, 78
- Phobos, 381, 384, 386, 406, 407
 - environment, 407
 - package, 388
 - shell command, 388
- PID
 - control, 36
 - control law, 440
 - control scheme, 439
 - controller, 36, 291, 292, 439
- Pitch
 - angle, 72, 79, 82, 292, 295, 346, 360
 - ankle, 337
 - control, 293, 297
 - direction, 336, 337
 - foot, 250
 - moment, 84, 85
 - motion, 238
 - movement, 18, 70, 80, 197, 204, 337
 - torque, 84
 - velocity component, 82–84
 - wrist, 219
- Planned trunk motion, 248
- Plücker
 - coordinates, 59
 - transformation matrix, 129
- Poinso't's Theorem, 58
- Point Jacobian, 172
- Point-Cloud-Library (PCL), 251
- Pose
 - bipedal, 237, 243, 254
 - computation, 375
 - configuration, 149
 - estimation, 250, 253
 - manipulator, 373
 - robot, 393
 - target, 296
- Position
 - ankle, 249
 - constraints, 204, 205, 405
 - control, 277, 324
 - control mode, 326
 - controller, 344
 - coordinates, 283
 - encoders, 323
 - interpolator, 420
 - joints, 141, 152, 191, 226, 290, 309, 340, 418
 - level, 204, 216
 - explicit constraints, 151
 - implicit constraint, 148
 - limits, 205
 - sensors, 309, 310
 - state, 413
 - tracking, 226
 - variables, 154
 - vectors, 101, 102, 112, 138, 150
- Positioning joint, 201, 203
- Postural control, 195
- Posture
 - controller, 247, 274
 - regularization, 222, 344
- Power management, 318

Praying mantis, 261–263
 Predecessor body, 148, 149, 404, 411
 Premature convergence, 457
 Pressure
 sensors, 365, 370
 Pressure sensors, 249
 Printed circuit board (PCB), 242
 Prismatic
 actuator, 90, 147, 219, 314, 404, 452, 454
 actuator forces, 71
 joints, 52, 61, 75, 76, 108, 111, 148, 196, 245, 404, 438, 454, 455
 joints actuated, 61, 430
 Problem description, 261, 281, 357
 Processing elements, 26
 Processing System (PS), 317
 Processing workflow, 391
 Product of exponential (POE), 60, 124
 Programmable Logic Controllers (PLC), 360
 Programmable Logic (PL), 317
 Proprioceptive sensors, 277

Q

Quadratic problem (QP), 345, 346
 Quadratic Programming (QP), 10
 Quadruped robot, 248
 Quadrupedal
 pose, 243, 254
 walking motions, 257
 Quality of Service (QoS) levels, 32

R

Radius Input module, 295
 Range of Motion (RoM), 79, 220, 238, 239, 308, 334
 Reachable workspace, 204
 Reactive
 modules, 247, 300
 walking control, 256
 Real Time Toolkit (RTT), 324, 414
 Real-time kinematic (RTK) GPS, 374
 Rear
 foot, 239
 legs, 242, 248, 264, 292
 Recalculated workspace, 437

Reconfigurable modular robots, 394
 Reconfigured locomotion pattern, 262
 Recupera
 exoskeleton, 100, 306, 308, 315, 322, 417
 legs, 100
 upper body exoskeleton, 328
 Recursive
 forward dynamics, 180, 183, 191
 kinematics computation, 123
 Newton–Euler algorithm, 141
 Redundant
 constraints, 157
 parallel mechanisms, 54
 Region of interest (ROI), 252
 Regular numbering, 122
 Regulated torque, 439
 Rehabilitation, 326
 Rehabilitation scenario, 306
 Relaxed
 inverse kinematic model, 108
 translative forward kinematic model, 109
 Remote control, 305, 306, 330, 362, 363, 370, 371
 Reorientation Hold, 295
 module, 290
 Reusable
 joint modules, 28, 29
 modules, 30
 Revolute joints, 19, 22, 52, 61, 70, 74, 94, 99, 147, 148, 189, 219, 245, 289, 404, 405, 467
 angles, 104, 107
 DoF, 52
 motion, 56
 RH5 Manus, 221, 404
 humanoid robot, 180
 robot, 191, 198, 205, 217, 219, 221, 226
 RH5 Pedes
 humanoid robot, 350
 robot, 334, 338, 348
 Rigid bio-inspired piping inspection robot, 428
 Rigid body
 kinematics, 130
 motion, 132
 physical properties, 132

- Rigid Body Dynamics Library (RBDL), 35
- Robomorphic Computing, 30
- Robot
 - actuators, 195
 - base, 296, 372, 373
 - base frame, 243
 - behavior, 247, 273, 274, 414
 - bipedal, 18, 19
 - Charlie, 235, 236, 272
 - chassis, 290
 - configuration, 53, 54
 - control, 217
 - description, 35, 384, 401, 402
 - description formats, 33, 35, 384, 401
 - designs, 19, 22, 61, 159, 161, 423
 - development, 397, 399
 - DoF, 52
 - dynamic discretization, 343
 - dynamics, 34, 132, 198, 210, 217
 - feet dimensions, 342
 - frameworks, 31
 - general mobility, 235
 - humanoid, 5, 18, 70, 161, 195, 197, 213, 306, 308, 333, 451
 - hybrid, 5, 19, 22, 26, 159, 162, 163, 226
 - Jacobian, 199
 - joints, 31, 200, 202
 - kinematics, 195, 389
 - locomotion, 250
 - Mantis, 272
 - middleware, 340, 414
 - middleware frameworks, 31
 - mobility, 254
 - model, 37, 198, 203, 208, 215, 227, 381, 383, 401, 409, 413
 - modeling, 341
 - morphology, 276
 - motion, 53, 54, 236
 - motion planning, 195
 - platform, 236
 - pose, 393
 - prototype, 262
 - SherpaTT, 299
 - software frameworks, 32
 - state, 217
 - steering joints, 371
 - systems, 28, 153
 - target, 247
 - tasks, 195, 201, 203
 - torque control, 343
 - torso, 238
 - velocities, 289
 - visualization, 414, 419
 - walking, 235, 247, 262, 277
 - workspace, 197, 198, 422
- Robot Construction Kit (RoCK), 31, 287, 324, 395, 414
- Robot Operating System (ROS), 31, 325, 365, 395, 414
- Robotic
 - abstractions, 235
 - agent, 236, 264
 - applications, 381, 393, 415
 - arm, 308, 324, 388
 - community, 30, 180, 183, 196, 409
 - control, 28
 - decontamination, 283
 - framework, 365
 - hardware, 384
 - platform, 214, 255, 372
 - projects, 266
 - rehabilitation device, 305
 - support structure, 305
 - swarm, 255
 - systems, 4, 22, 53, 143, 195, 233, 234, 261, 281, 282, 306, 308, 357, 384, 398, 407
 - team, 301
 - world, 17
- Roll
 - motion, 238, 335, 336
 - movement, 70, 83–85
 - velocity component, 82
- Roll-Pitch Adaption (RPA), 291, 292
- Root Mean Square (RMS), 298
- Rotary
 - actuators, 269, 336, 337
 - joints, 416
 - motion, 66
- Rotational
 - actuators, 338
 - joints, 241, 314
 - mechanical joints, 305
 - motion, 25, 56, 133

- movement, 52
 - sensors, 369
- Rotative inverse kinematic model, 107
- Rotative Inverse Kinematic Problem (RIKP), 103, 107, 117
- Rover system SherpaTT, 285
- S**
- Safe
 - locomotion, 255
 - working zone, 453
- Safety
 - aspects, 315, 318
 - controllers, 370
 - mechanisms, 318
- Sceleronomic constraints, 146
- Scientific robotics community, 402
- Screw
 - coordinates, 58–60, 125, 128, 130, 148, 185
 - motion, 59, 61, 124
 - product, 131
 - representation, 61
 - theory, 34, 35, 57, 58, 124, 132, 148, 184
- Search motion, 248
- Second optimization problem, 444
- Self-localization and mapping (SLAM)
 - module, 373
- Sensors
 - foot, 257
 - orientation control, 298
 - passive, 236
 - position, 309, 310
 - rotational, 369
- Serial
 - abstraction, 199, 214, 215, 223, 224
 - chain, 4, 189, 213, 313, 404, 405
 - chain robot, 57
 - chain submechanisms, 218
 - combination, 160
 - counterparts, 35, 36
 - joints, 288
 - kinematic chain, 20, 60, 122
 - kinematics, 335, 337
 - link chains, 148
 - manipulators, 33
 - mechanism, 29, 53, 381, 400, 409
 - models, 208, 215, 227
 - robotic systems, 52, 245
 - robots, 4, 20, 33, 34, 54, 124, 199
 - submechanism, 166
 - submechanism modules, 161, 166, 170
 - systems, 36
- Series composition, 162
- Series-parallel hybrid
 - composition, 170
 - design, 17, 305
 - manipulator, 34
 - robot, 4, 5, 17, 35, 161, 183, 213, 215, 217
- Servoing module, 371
- Sherpa rover, 284
- SherpaTT
 - features, 285
 - kinematics, 283
 - laboratories, 299
 - platform, 301
 - robot, 299
 - rover, 19, 283, 300
 - system, 289
- Shifting actuator, 338
- Shoulder joint, 305, 311, 320, 334–336, 338
- Simulation Description Format (SDF), 384
- Single rigid body dynamics, 132
- Single-point-contact feet (SPCF), 235
- Singularity, 78
- Singularity analysis, 61
- SLAM, 251, 296
- Sobol sequence, 463
- Socket joints, 98
- Software
 - design, 271, 287, 324, 365
 - modules, 32
- Solution approach, 111
- Spanning tree
 - joints, 160, 163, 187, 199, 208, 217, 403
 - state, 419
- Spatial
 - force, 58
 - Jacobian, 127, 128
 - mass-inertia matrix, 135
 - momentum, 135

- representation, 136
 - system Jacobian, 129
 - velocity, 58
 - Spherical joint, 22, 25, 74, 95, 97, 98, 156, 219, 314, 430, 469
 - module, 20, 117
 - passive, 109, 197
 - Spherical Parallel Manipulator (SPM), 94, 99
 - Spherical-revolute-spherical (SRS), 400
 - Spherical-revolute-universal (SRU), 400
 - Spindle mechanism, 272
 - Spine, 240
 - actuators, 246
 - controller, 246
 - joints, 245
 - sensor processing, 246
 - support, 248
 - Squatting motion, 204
 - Stacked
 - modules, 437
 - tensegrity modules, 446
 - Standing position, 308
 - Static force modeling, 445
 - Stereo cameras, 242
 - Stiff positioning
 - functionality, 95
 - system, 95
 - Stiffness control, 35, 36
 - Study's parameters, 56
 - Submechanism
 - ankle, 407
 - class, 409
 - definition, 402
 - description, 407, 411
 - design, 401
 - file, 416
 - graphs, 163
 - interface, 162–164
 - libraries, 401, 409
 - model, 163, 401
 - module, 26, 38, 159–163, 166, 310, 401
 - node, 183–187
 - torso, 219
 - YAML file, 403, 404, 411
 - Successor body, 122, 149, 404, 411
 - Supplementable, Mostly Universal Robot Format (SMURF), 385, 402
 - Support Polygon (SP), 248
 - Surrogate motion, 313
 - Swinging foot translation, 344
 - Swivel joints, 373
 - Synchronized motions, 334
- ## T
- Tactile sensors, 255, 271
 - Tangent half-angle substitution, 56
 - Target
 - footstep, 252
 - pose, 296
 - robot, 247
 - Task Jacobian, 202
 - Task Space Inverse Dynamics (TSID), 345
 - Teach & Replay (TR) mode, 323, 324
 - Teleoperated
 - control, 358
 - robot, 324
 - Teleoperation (TO), 327
 - mode, 324
 - scenario, 306
 - Temperature sensors, 271
 - Tensegrity
 - mechanism, 428–430, 432, 437, 438
 - modules, 445
 - Terrain adaption controller, 372, 376
 - Tilt, 74
 - Tilt position, 440, 442
 - Time of Flight (ToF) camera, 242, 244
 - Time-varying position, 133
 - Top-down
 - decomposition, 164
 - modeling, 407
 - Topological
 - decomposition, 164
 - graph, 122, 123, 147, 164, 179, 189, 217, 221, 405
 - modeling, 162
 - Torque
 - commanded, 417
 - consistency, 393
 - constraints, 207
 - control, 203, 322, 323, 343, 399, 401
 - level, 198, 207, 220
 - limits, 202, 217
 - maximum, 207, 220
 - measurements, 38

- moment, 341
- pitch, 84
- profile, 337, 419
- readings, 246
- regularization, 344
- requirements, 399
- symmetry, 393
- transmission characteristics, 70
- values, 246, 441
- variables, 141, 153
- Torso, 72, 219, 335
 - designs, 19
 - joints, 18, 26, 69, 86, 159, 272, 469
 - mechanism, 25, 74
 - pitch joint, 223
 - robot, 238
 - structure, 335, 337
 - submechanism, 219
 - yaw joint, 338
- Trajectory
 - movement, 323
 - optimization, 205, 211, 213, 215, 222, 223
 - formulation, 216, 223
 - process, 215, 227
 - planning, 438
- Translational
 - motions, 313
 - movements, 311
- Translative Forward Kinematic Problem (TFKP), 109
- Tree
 - abstraction, 221
 - abstraction model, 223
 - joints, 123, 141
- Tripod parallel mechanisms, 19
- Trunc controller, 247
- Trust Region Dog Leg (TRDL), 113
- Twist-wrench formulation, 135
- Two-terminal graph (TTG), 162
- Type synthesis, 94

U

- Unbranched kinematic chain, 124, 125, 128
- Unified Architecture (UA), 32

- Unified Robot Description Format (URDF), 195, 320, 324, 381, 384, 401, 407, 411
 - file, 403, 405, 416
- Universal joint, 22, 25, 69–71, 97, 269, 272, 338, 404, 430, 431, 469, 471
 - actuated, 430
 - axes, 471
- Universal Robot Description Format (URDF), 319
- Usable workspace, 291, 293, 294

V

- Valve controllers, 363
- Velocity
 - commands, 371
 - computation, 127
 - constraints, 205
 - control, 370
 - dependencies, 366
 - elbow, 206
 - equation, 168
 - factors, 444
 - joints, 334
 - level, 150, 151
 - explicit constraint, 151
 - implicit constraint, 150
 - loop constraints, 150
 - limits, 206, 221
 - linear, 133, 135
 - maximum, 83, 207, 220
 - requirements, 25
 - tracking, 421
 - transmission, 86, 398
 - vector, 167
- Velocity-based WBC, 200
- Virtual
 - drive control module, 371
 - joints, 115
- Voluntary movements, 326

W

- Walking
 - excavator robot HEAP, 356
 - movement, 267
 - robot, 235, 247, 262, 277
 - task, 342

- Wheel
 - ground contact, 372
 - positions, 290, 294
 - Wheel Steering Support (WSS), 291, 293
 - WheelSteering
 - actuator torque stall, 291
 - axis, 286
 - joint, 293, 295, 300
 - support, 293
 - Whole-body control (WBC), 195, 344, 347
 - applications, 418
 - architecture, 199
 - Workspace, 53, 78
 - admissible, 203, 210
 - analysis, 36, 434, 437, 447
 - ankle, 89
 - exploitation, 210
 - limits, 435
 - manipulator, 453
 - orientation, 79, 80, 89
 - restriction, 38
 - robot, 197, 198, 422
 - size, 451
 - Wrench, 58
 - Wrist, 220
 - geometry, 78
 - interface, 336
 - joints, 19, 77, 320, 338
 - mechanism, 71, 72, 74, 77, 86, 219
 - pitch, 219
 - poses, 327
 - position, 222
 - target placement, 222
 - target tracking, 222
- Y**
- YAML based submechanism, 416
 - Yaw motion, 238, 239, 335, 336
 - Yet Another Robot Platform (YARP), 31, 414
- Z**
- Zero Moment Point (ZMP), 248
 - ZYX ball joints, 156

This page intentionally left blank

Biologically Inspired Series-Parallel Hybrid Robots

Design, Analysis, and Control

Shivesh Kumar, Andreas Müller, and Frank Kirchner

Biologically Inspired Series-Parallel Hybrid Robots: Design, Analysis, and Control provides an extensive review of the state-of-the-art in series-parallel hybrid robots, covering all aspects of their mechatronic system design, modelling, and control. This book highlights the modular and distributed aspects of their mechanical, electronics, and software design, introducing various modern methods for modelling the kinematics and dynamics of complex robots. These methods are also introduced in the form of algorithms or pseudo-code which can be easily programmed with modern programming languages. Presenting case studies on various popular series-parallel hybrid robots which will inspire new robot developers, this book will be especially useful for academic and industrial researchers in this exciting field, as well as graduate-level students to bring them closer to the latest technology in mechanical design and control aspects of the area.

Key Features

- Introduces clear definitions for all relevant terms and the foundational theories
- Provides in-depth kinematics of various parallel mechanisms typically used in the design of series-parallel hybrid robots
- Presents holistic methods for solving kinematics, dynamics, trajectory generation, and control of series-parallel hybrid robots considering large number of holonomic constraints
- Investigates case studies on the mechatronic system design of various series-parallel hybrid robots for practitioners in the field

About the Authors

Shivesh Kumar is an assistant professor at the Division of Dynamics, Department of Mechanics and Maritime Sciences, Chalmers University of Technology in Gothenburg, Sweden. He is also a visiting researcher at the Robotics Innovation Center, German Research Center for Artificial Intelligence in Bremen, Germany. He obtained his PhD degree from the faculty of Mathematics and Computer Science at the University of Bremen (2019). His research interests include kinematics, dynamics, and control of robots with applications in the fields of exoskeletons, humanoids, rehabilitation, and industrial automation.

Andreas Müller obtained diploma degrees in mathematics, electrical engineering, and mechanical engineering, and a PhD in mechanics. He received his Habilitation in mechanics and is currently professor and director of the Institute of Robotics at the Johannes Kepler University, Linz, Austria. His current research interests include holistic modelling, model-based and optimal control of mechatronic systems, redundant robotic systems, parallel kinematic machines, biomechanics, and computational dynamics.

Frank Kirchner studied computer science and neurobiology at the University Bonn, where he received his PhD degree in computer science. He was senior scientist at the Gesellschaft für Mathematik und Datenverarbeitung (GMD) in Sankt Augustin, Germany, and a Senior Scientist at the Department for Electrical Engineering at Northeastern University in Boston, USA. Dr. Kirchner was first appointed adjunct and then tenure track assistant professor at the Northeastern University, and then as a full professor at the University of Bremen. Since December 2005, Dr. Kirchner has also been director of the Robotics Innovation Centre (RIC) in Bremen.



ACADEMIC PRESS

An imprint of Elsevier

elsevier.com/books-and-journals

ISBN 978-0-323-88482-2



9 780323 884822