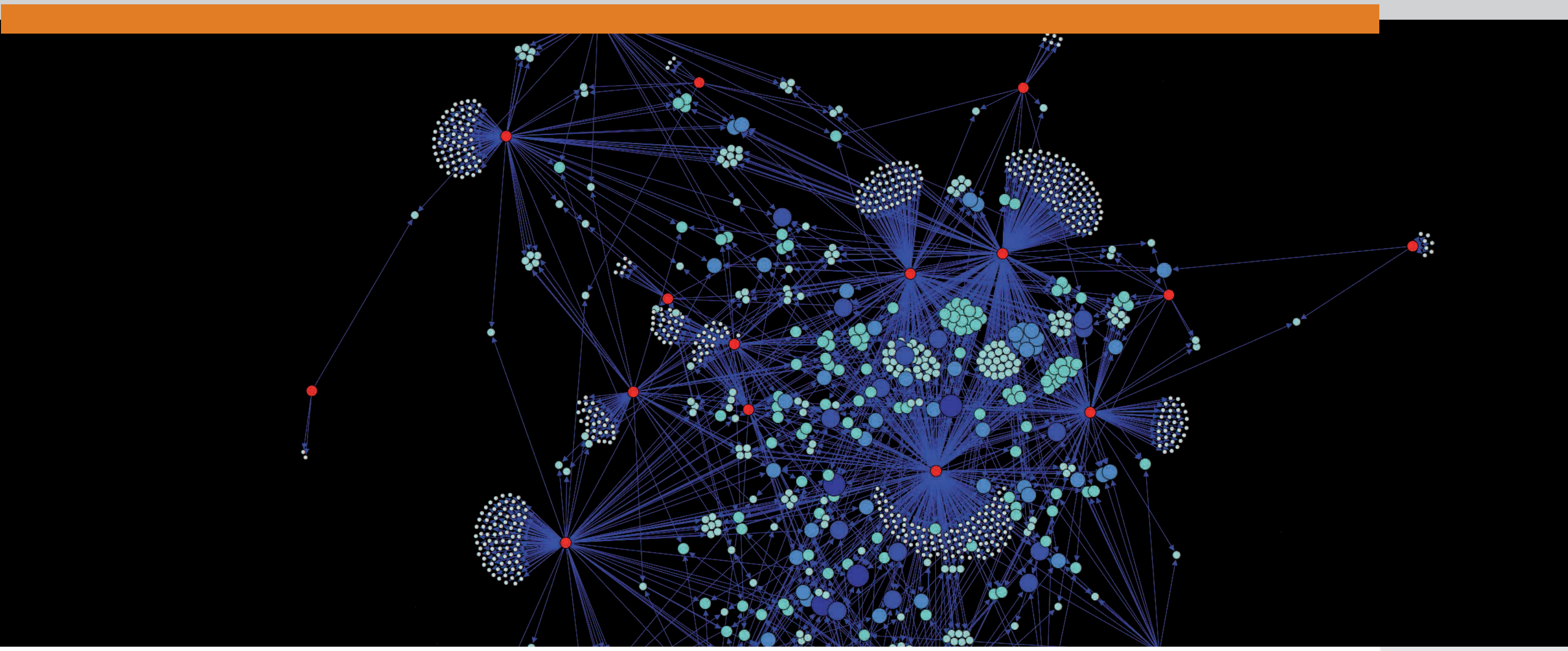
German
Research Center
for Artificial
Intelligence

Document D-14-03



Proceedings of the RIC Project Day

Workgroups *'Framework & Standardization'* and *'Manipulation & Control'*

Frank Kirchner (Editor)
Thomas M. Roehr, Bertold Bongardt (Associate Editors)

06/2014

Document D-14-03
German Research Center for Artificial Intelligence (DFKI) GmbH

Bibliographic information published by the German National Library

The German National Library lists this publication in the German National Biography; detailed bibliographic data are available in the internet at `http://dnb.ddb.de`.

Issue D-14-03 (2014)
ISSN 0946-0098

German Research Center for Artificial Intelligence
Deutsches Forschungszentrum für Künstliche Intelligenz
DFKI GmbH

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation – from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern, Saarbrücken and Bremen, the German Research Center for Artificial Intelligence ranks among the important 'Centers of Excellence' worldwide. An important element of DFKI's mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science DFKI has the strength to meet its technology transfer goals.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO). DFKI's research departments are directed by internationally recognized research scientists:

- Knowledge Management (Prof. A. Dengel)
- Cyber-Physical Systems (Prof. R. Drechsler)
- Robotics Innovation Center (Prof. F. Kirchner)
- Innovative Retail Laboratory (Prof. A. Krüger)
- Institute for Information Systems (Prof. P. Loos)
- Embedded Intelligence (Prof. P. Lukowicz)
- Agents and Simulated Reality (Prof. P. Slusallek)
- Augmented Vision (Prof. D. Stricker)
- Language Technology (Prof. H. Uszkoreit)
- Intelligent User Interfaces (Prof. W. Wahlster)
- Innovative Factory Systems (Prof. D. Zühlke)

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

Proceedings of the RIC Project Day

Workgroups 'Framework & Standardization' and 'Manipulation & Control'

Frank Kirchner [1,2] (Editor)

Thomas M. Roehr [1], Bertold Bongardt [1] (Associate Editors)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany
(2) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

06/2014

Document D-14-03 des

Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI)

Abstract

This document is the current edition of a publication series which records the topics, discussions and efforts of the workgroups at the DFKI Robotics Innovation Center (RIC). Each edition contains presentation slides and posters of a project day which is organized by two workgroups.

Workgroups provide a platform for cross-project communication and knowledge transfer. They are formed by peers dedicated to a specific topic. Each workgroup has one administrator. In 2008, the workgroups started to present their results and efforts in an open presentation format called brown-bag talk. From 2009 onwards, these presentation were held at so-called project days. Since 2014, a project day consists of two main parts: an oral session and a poster session. Both sessions are documented in a proceedings using the DFKI Document format.

Zusammenfassung

Dieses Dokument enthält die aktuelle Ausgabe einer Tagungsbandserie, welche die Themen, Diskussionen und Bemühungen der Arbeitsgruppen am DFKI Robotics Innovation Center (RIC) protokolliert. Jede Ausgabe enthält Vortragsfolien und Poster eines Projekttages, der von je zwei Arbeitsgruppen gestaltet wird.

Arbeitsgruppen widmen sich einem bestimmten Themengebiet und stellen eine Plattform dar, um über Projekte hinaus zu kommunizieren und Wissen zu transferieren. Jede Arbeitsgruppe wird von einem sogenannten Kümmerer administriert. Im Jahr 2008 begannen die Arbeitsgruppen ihre Ergebnisse und Arbeiten in einem offenen Vortragsformat – dem sogenannten ‘Brown Bag Talk’ – vorzustellen, welches ein Jahr später in die Form von Projekttagen überführt wurde. Seit 2014 besteht ein Projekttag nicht nur aus Vorträgen, sondern beinhaltet zudem Posterpräsentationen. Beide Formate werden seitdem in einem Tagungsband in Form eines ‘DFKI Document’ festgehalten.

Contents

1 Editorial

This is the first edition of a new format to document the efforts of the DFKI-RIC thematic workgroups. Workgroups are formed by peers and provide a means for cross-project communication on a deep content level and facilitate knowledge transfer amongst the peers. In 2008 we first started forming workgroups on specific topics around robotics and AI research. Among them were topics as ‘system design & engineering’, ‘machine learning’, ‘planning & representation’ as well as ‘frameworks & architectures’ and ‘man-machine interaction’. These workgroups were established with the intention to provide a platform for interested DFKI-RIC personnel for discussing the start of the art, recent achievements, and future developments in the respective fields.

Over time the workgroups gathered a collection of material in form of presentations, short papers, and posters which were worthwhile to be presented also to the rest of the institute. Due to this development, in 2009, we started to have a project day once every quarter. Each project day provided a platform for two of the workgroups to present their material and to discuss it with the further colleagues of the institute. Nowadays, the project day is organized as a one-day workshop with oral presentations, poster sessions, and a free pizza lunch for everybody who attends. Until now, the talks and posters have only been collected on our servers but were not assembled in a citable document.

This format at present is the next evolutionary step and it aims at eliminating this deficit by compiling the material of the workgroups presented during a project day into a single, citable document of unified format. We will see which steps can be taken in the future to enhance the presentation quality of this material.

Frank Kirchner

This year’s second project day presented the material of the workgroups ‘Framework & Standardization’ and ‘Manipulation & Control’.

The workgroup ‘Framework & Standardization’ focuses its efforts on continuously improving the software development workflow and aims at supporting a software framework which fulfills the special needs in the domain of robotics. The workgroup’s main motivation is to facilitate and accelerate routine tasks and to increase the robustness of the developed software. The workgroup has successfully established the Robot Construction Kit (Rock) as the main in-house development framework which can coexist with the well-known Robot Operating System (ROS). Furthermore, it has introduced a flexible git-based hosting for software projects in parallel to the existing subversion-based infrastructure. The presentations of this year are dealing with workflow optimization and the requirements of distributed systems with respect to modeling and infrastructure.

The purpose of the workgroup ‘Manipulation & Control’ is to discuss problems and solution approaches in the fields of kinematics, dynamics, and joint control. Questions in these areas arise commonly during the development of the various, unique robotic systems that are manufactured at the DFKI-RIC. The findings of the workgroup’s members enable to apply the control and planning approaches to the mechatronic systems of the institute. These include walking machines, wheeled systems, robotic arms, exoskeletons, and the humanoid AILA. The contributions of the workgroup to the project day 2014 distribute across the areas ‘motion planning along waypoints’, ‘generation of dynamic trajectories’, ‘real-time computation of manipulator dynamics’, and ‘representation of spatial rotations’.

We would like to thank the authors of the second project day 2014 for their contributions and for the effort to provide their material in a standardized format.

Thomas M. Roehr, Bertold Bongardt

2 ‘Framework & Standardization’

2.1 ‘AG Framework and Standardization’ (FS-T-01)

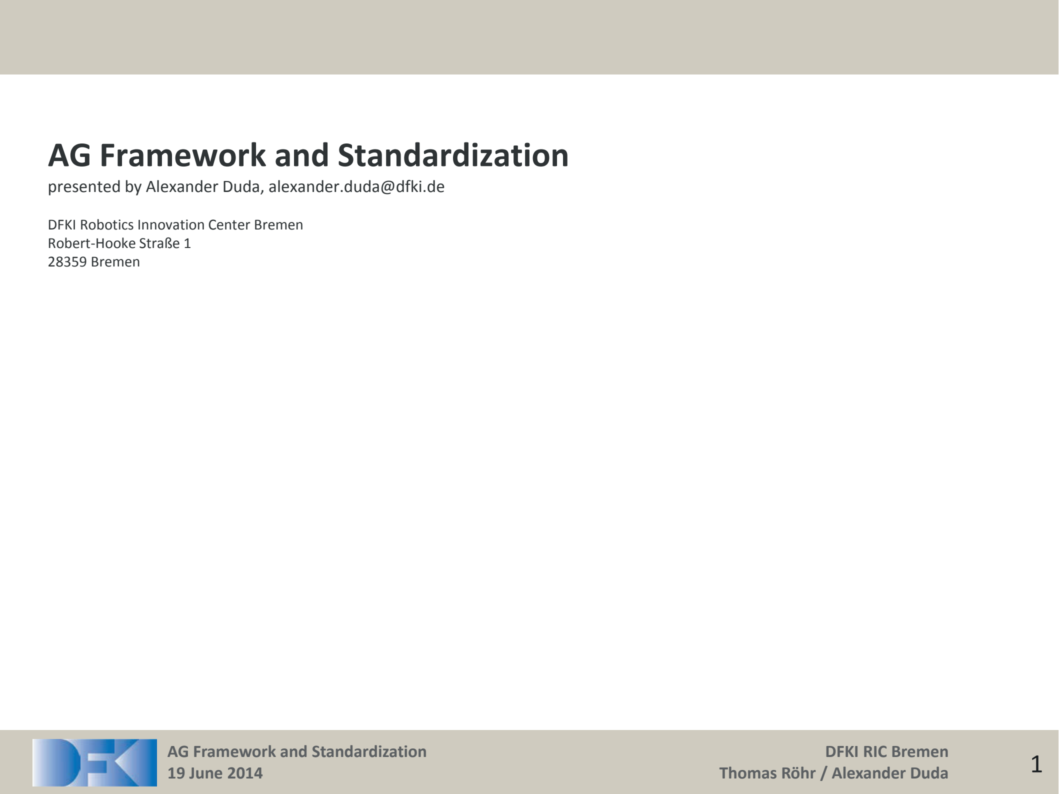
Alexander Duda(1), *Thomas M. Roehr*(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany
(2) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `alexander.duda@dfki.de`, `thomas.roehr@dfki.de`

Abstract

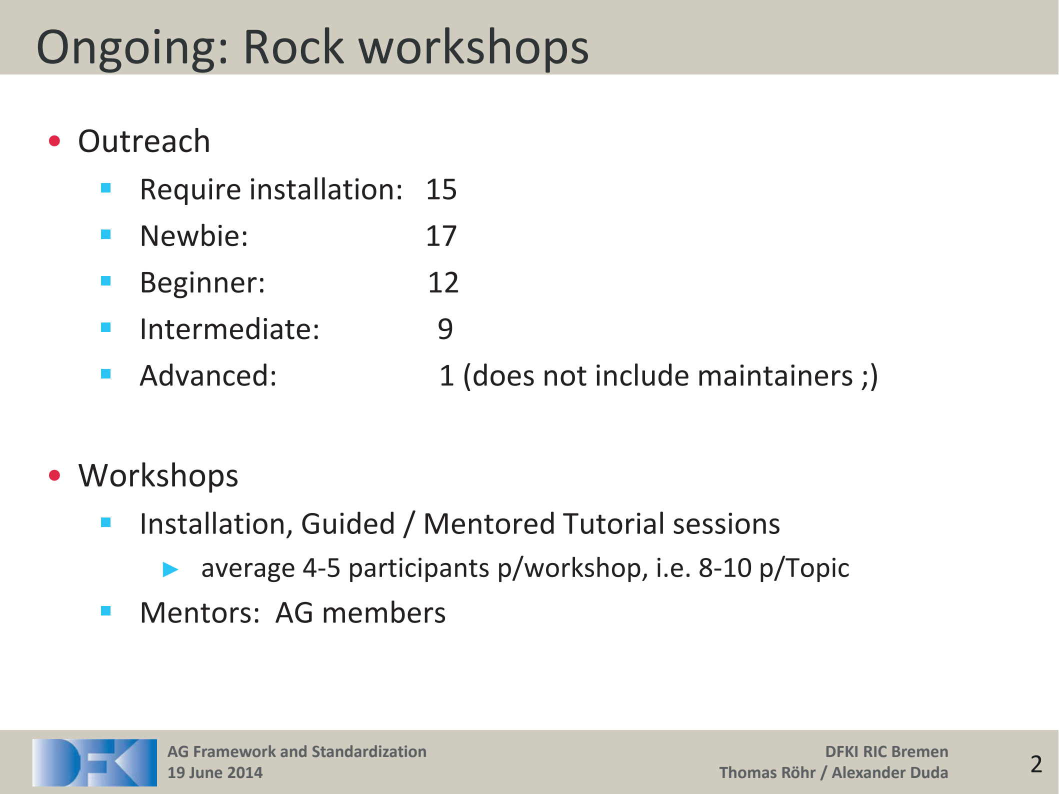
This introduction provides a compact outline of the activities of the work group over the past year. The activities cover the past organization of Rock workshops and improving the build and development infrastructure and also points to upcoming changes.



AG Framework and Standardization

presented by Alexander Duda, alexander.duda@dfki.de

DFKI Robotics Innovation Center Bremen
Robert-Hooke Straße 1
28359 Bremen

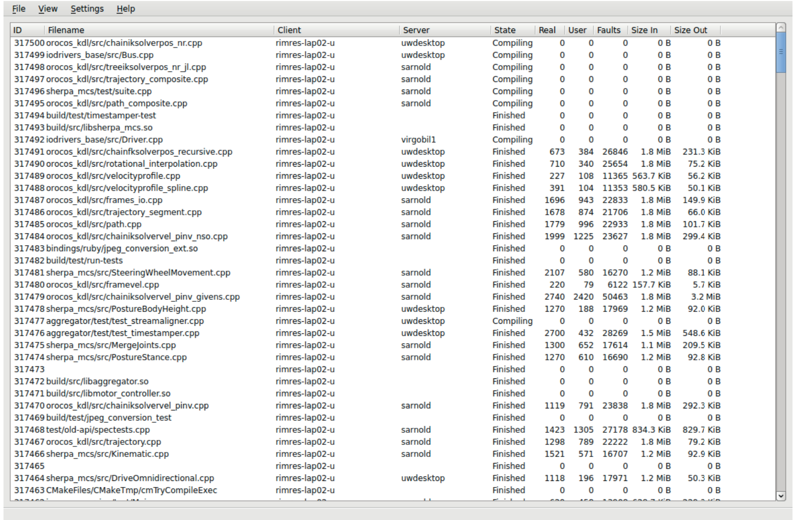


Ongoing: Rock workshops

- Outreach
  - Require installation: 15
  - Newbie: 17
  - Beginner: 12
  - Intermediate: 9
  - Advanced: 1 (does not include maintainers ;)

- Workshops
  - Installation, Guided / Mentored Tutorial sessions
    - average 4-5 participants p/workshop, i.e. 8-10 p/Topic
  - Mentors: AG members

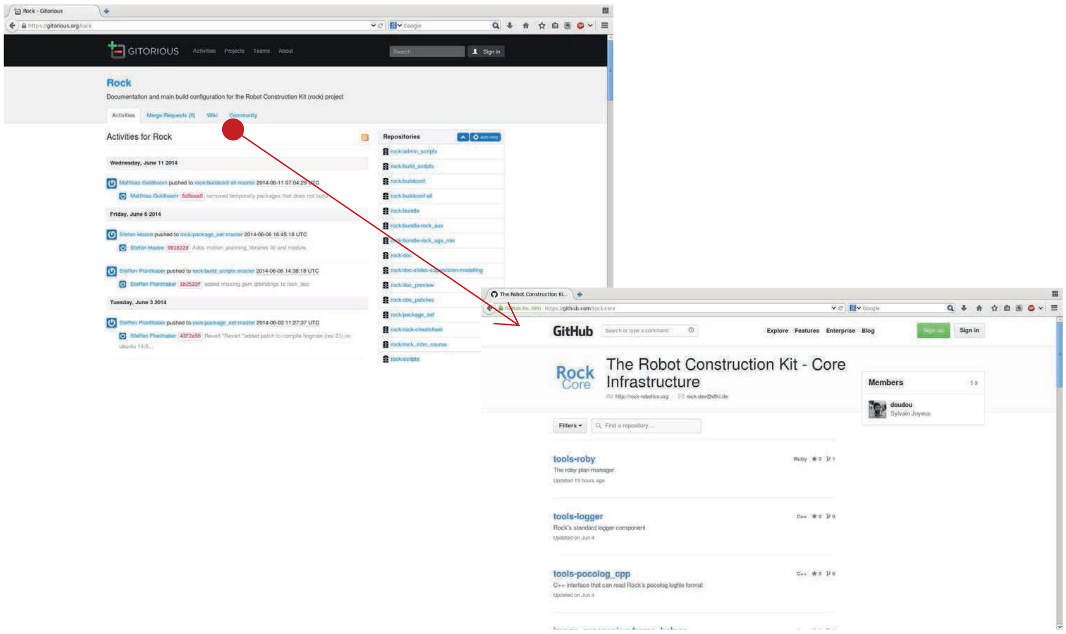
Ongoing: Distributed Compilation



ID	Filename	Client	Server	State	Real	User	Faults	Size In	Size Out
317500	orocos_kdl/src/chainiksolverpos_nr.cpp	rimres-lap02-u	uwdesktop	Compiling	0	0	0	0 B	0 B
317499	iodrivers_base/src/Bus.cpp	rimres-lap02-u	uwdesktop	Compiling	0	0	0	0 B	0 B
317498	orocos_kdl/src/treeiksolverpos_nr_jl.cpp	rimres-lap02-u	sarnold	Compiling	0	0	0	0 B	0 B
317497	orocos_kdl/src/trajectory_composite.cpp	rimres-lap02-u	sarnold	Compiling	0	0	0	0 B	0 B
317496	sherpa_mcs/test/suite.cpp	rimres-lap02-u	sarnold	Compiling	0	0	0	0 B	0 B
317495	orocos_kdl/src/path_composite.cpp	rimres-lap02-u	sarnold	Compiling	0	0	0	0 B	0 B
317494	build/test/timestamper-test	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317493	build/src/libsherpa_mcs.so	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317492	iodrivers_base/src/Driver.cpp	rimres-lap02-u	virgobil1	Compiling	0	0	0	0 B	0 B
317491	orocos_kdl/src/chainfksolverpos_recursive.cpp	rimres-lap02-u	uwdesktop	Finished	673	384	26846	1.8 MiB	231.3 KiB
317490	orocos_kdl/src/rotational_interpolation.cpp	rimres-lap02-u	uwdesktop	Finished	710	340	25654	1.8 MiB	75.2 KiB
317489	orocos_kdl/src/velocityprofile.cpp	rimres-lap02-u	uwdesktop	Finished	227	108	11365	563.7 KiB	56.2 KiB
317488	orocos_kdl/src/velocityprofile_spline.cpp	rimres-lap02-u	uwdesktop	Finished	391	104	11353	580.5 KiB	50.1 KiB
317487	orocos_kdl/src/frames_io.cpp	rimres-lap02-u	sarnold	Finished	1696	943	22833	1.8 MiB	149.9 KiB
317486	orocos_kdl/src/trajectory_segment.cpp	rimres-lap02-u	sarnold	Finished	1678	874	21706	1.8 MiB	66.0 KiB
317485	orocos_kdl/src/path.cpp	rimres-lap02-u	sarnold	Finished	1779	996	22933	1.8 MiB	101.7 KiB
317484	orocos_kdl/src/chainiksolvervel_pinv_nso.cpp	rimres-lap02-u	sarnold	Finished	1999	1225	23627	1.8 MiB	299.4 KiB
317483	bindings/ruby/jpeg_conversion_ext.so	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317482	build/test/run-tests	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317481	sherpa_mcs/src/SteeringWheelMovement.cpp	rimres-lap02-u	sarnold	Finished	2107	580	16270	1.2 MiB	88.1 KiB
317480	orocos_kdl/src/framevel.cpp	rimres-lap02-u	sarnold	Finished	220	79	6122	157.7 KiB	5.7 KiB
317479	orocos_kdl/src/chainiksolvervel_pinv_givens.cpp	rimres-lap02-u	sarnold	Finished	2740	2420	50463	1.8 MiB	3.2 MiB
317478	sherpa_mcs/src/PostureBodyHeight.cpp	rimres-lap02-u	uwdesktop	Finished	1270	188	17969	1.2 MiB	92.0 KiB
317477	aggregator/test/test_streamaligner.cpp	rimres-lap02-u	uwdesktop	Compiling	0	0	0	0 B	0 B
317476	aggregator/test/test_timestamper.cpp	rimres-lap02-u	uwdesktop	Finished	2700	432	28269	1.5 MiB	548.6 KiB
317475	sherpa_mcs/src/MergeJoints.cpp	rimres-lap02-u	sarnold	Finished	1300	652	17614	1.1 MiB	209.5 KiB
317474	sherpa_mcs/src/PostureStance.cpp	rimres-lap02-u	sarnold	Finished	1270	610	16690	1.2 MiB	92.8 KiB
317473		rimres-lap02-u		Finished	0	0	0	0 B	0 B
317472	build/src/libaggregator.so	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317471	build/src/libmotor_controller.so	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317470	orocos_kdl/src/chainiksolvervel_pinv.cpp	rimres-lap02-u	sarnold	Finished	1119	791	23838	1.8 MiB	292.3 KiB
317469	build/test/jpeg_conversion_test	rimres-lap02-u		Finished	0	0	0	0 B	0 B
317468	test/old-api/spectests.cpp	rimres-lap02-u	sarnold	Finished	1423	1305	27178	834.3 KiB	829.7 KiB
317467	orocos_kdl/src/trajectory.cpp	rimres-lap02-u	sarnold	Finished	1298	789	22222	1.8 MiB	79.2 KiB
317466	sherpa_mcs/src/Kinematic.cpp	rimres-lap02-u	sarnold	Finished	1521	571	16707	1.2 MiB	92.9 KiB
317465		rimres-lap02-u		Finished	0	0	0	0 B	0 B
317464	sherpa_mcs/src/DriveOmnidirectional.cpp	rimres-lap02-u	uwdesktop	Finished	1118	196	17971	1.2 MiB	50.3 KiB
317463	CMakeFiles/CMakeTmp/cmTryCompileExec	rimres-lap02-u		Finished	0	0	0	0 B	0 B

icemon screenshot – when using distributed compilation

Ongoing: Moving from gitorious to github



Rock

Documentation and main build configuration for the Robot Construction Kit (rock) project

Activities for Rock

Wednesday, June 11 2014

Friday, June 6 2014

Stefan Haase pushed to rock/package_set:master 2014-06-06 16:45:18 UTC

Steffen Planthaber pushed to rock/build_scripts:master 2014-06-06 14:38:18 UTC

Tuesday, June 3 2014

Steffen Planthaber pushed to rock/package_set:master 2014-06-03 11:27:37 UTC

rock/admin_scripts
rock/build_scripts
rock/buildconf
rock/buildconf-all
rock/bundle
rock/bundle-rock_auv
rock/bundle-rock_ugv_nav
rock/doc
rock/doc-slides-supervision-modelling
rock/doc_preview
rock/obs_patches
rock/package_set
rock/rock-cheatsheet
rock/rock_intro_course
rock/scripts

Rock Core

The Robot Construction Kit - Core Infrastructure

http://rock-robotics.org rock-dev@dfki.de

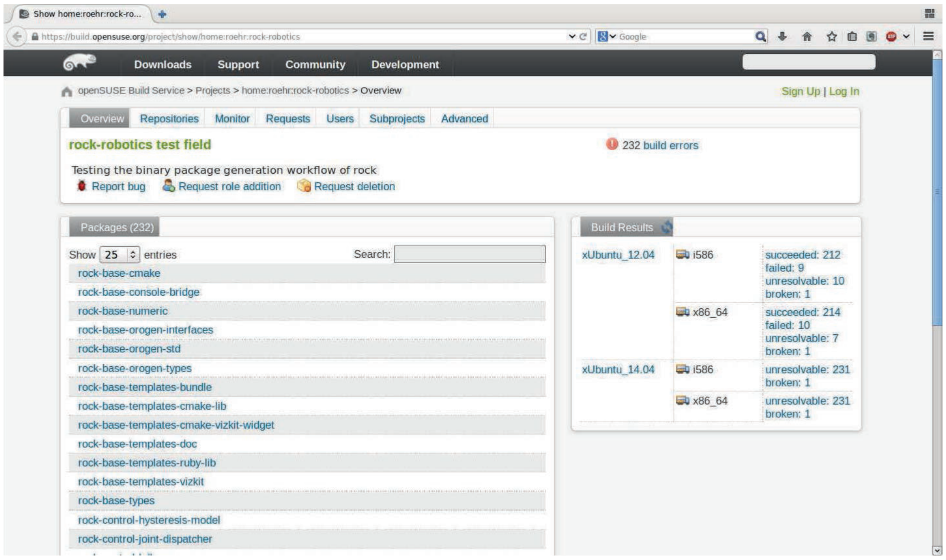
doudou
Sylvain Joyeux

tools-roby
The roby plan manager
Updated 13 hours ago

tools-logger
Rock's standard logger component
Updated on Jun 4

tools-pocolog_cpp
C++ interface that can read Rock's pocolog logfile format
Updated on Jun 4

Ongoing: Building Debian packages



openSUSE Build Service > Projects > home:roehr:rock-robotics > Overview

rock-robotics test field

232 build errors

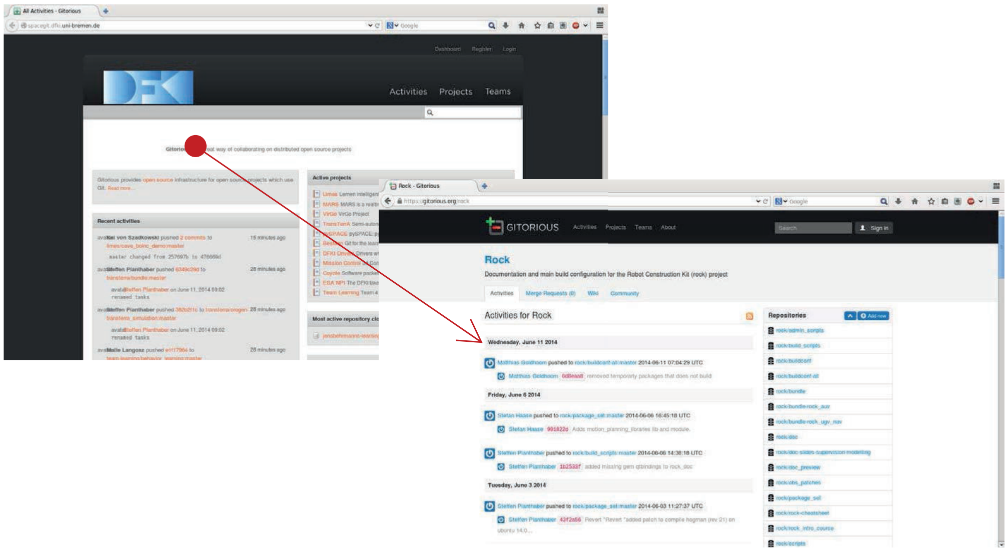
Testing the binary package generation workflow of rock

rock-base-cmake
rock-base-console-bridge
rock-base-numeric
rock-base-orogen-interfaces
rock-base-orogen-std
rock-base-orogen-types
rock-base-templates-bundle
rock-base-templates-cmake-lib
rock-base-templates-cmake-vizkit-widget
rock-base-templates-doc
rock-base-templates-ruby-lib
rock-base-templates-vizkit
rock-base-types
rock-control-hysteresis-model
rock-control-joint-dispatcher

xUbuntu_12.04	i586	succeeded: 212 failed: 9 unresolvable: 10 broken: 1
	x86_64	succeeded: 214 failed: 10 unresolvable: 7 broken: 1
xUbuntu_14.04	i586	unresolvable: 231 broken: 1
	x86_64	unresolvable: 231 broken: 1

Using OpenSUSE for building Debian packages for Rock

TBD: Upgrading spacegit



Gitorious provides open source infrastructure for open source projects which use Git. Read more...

Rock

Documentation and main build configuration for the Robot Construction Kit (rock) project

Activities for Rock

Wednesday, June 11 2014

Matthias Goldhoorn pushed to rock/buildconf-all:master 2014-06-11 07:04:29 UTC

Matthias Goldhoorn 6d8eaa8 removed temporarly packages that does not build

Friday, June 6 2014

Stefan Haase pushed to rock/package_set:master 2014-06-06 16:45:18 UTC

Stefan Haase 991822d Adds motion_planning_libraries lib and module.

Steffen Planthaber pushed to rock/build_scripts:master 2014-06-06 14:38:18 UTC

Steffen Planthaber 1b2533f added missing gem qtbindings to rock_doc

Tuesday, June 3 2014

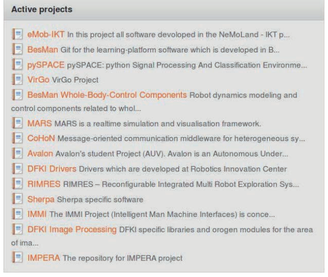
Steffen Planthaber pushed to rock/package_set:master 2014-06-03 11:27:37 UTC

Steffen Planthaber 43f2a56 Revert "Revert "added patch to compile hogman (rev 21) on ubuntu 14.0...

rock/admin_scripts
rock/build_scripts
rock/buildconf
rock/buildconf-all
rock/bundle
rock/bundle-rock_auv
rock/bundle-rock_ugv_nav
rock/doc
rock/doc-slides-supervision-modelling
rock/doc_preview
rock/obs_patches
rock/package_set
rock/rock-cheatsheet
rock/rock_intro_course
rock/scripts

Upgrading Spacegit from „experimental“ to production, i.e.
to latest version of gitorious

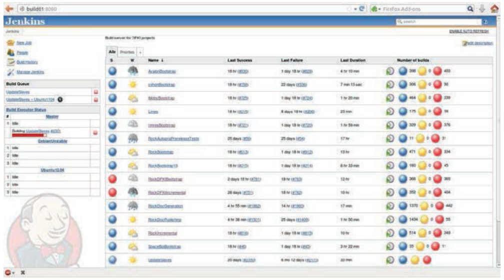
… and still going …



Active projects

- eMob-IKT In this project all software devoloped in the NeMoLand - IKT p...
- BesMan Git for the learning-platform software which is developed in B...
- pySPACE pySPACE: python Signal Processing And Classification Environme...
- VirGo VirGo Project
- BesMan Whole-Body-Control Components Robot dynamics modeling and control components related to whol...
- MARS MARS is a realtime simulation and visualisation framework.
- CoHoN Message-oriented communication middleware for heterogeneous sy...
- Avalon Avalon's student Project (AUV). Avalon is an Autonomous Under...
- DFKI Drivers Drivers which are developed at Robotics Innovation Center
- RIMRES RIMRES – Reconfigurable Integrated Multi Robot Exploration Sys...
- Sherpa Sherpa specific software
- IMMI The IMMI Project (Intelligent Man Machine Interfaces) is conce...
- DFKI Image Processing DFKI specific libraries and orogen modules for the area of ima...
- IMPERA The repository for IMPERA project

Spacegit



Buildserver

Rock

the Robot Construction Kit

What is Rock ?

Get it

Get started with the step-by-step installation guide.

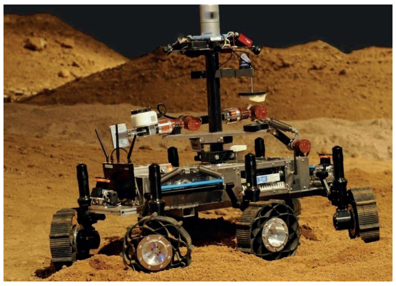
Use it

Community

Robot Construction Kit (Rock)

… suspended …

Reference architecture, though there is



2.2 ‘Cross Compiling’ (FS-T-02)

Martin Zenzes(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `martin.zenzes@dfki.de`

Abstract

This presentation provides a general introduction into the challenges of cross compiling for different types of target systems. It illustrates the need for cross compilation and presents the existing cross compiling approach. Furthermore, it motivates the need for an integrated cross-compilation workflow to fully support cross platform software development.

AG Framework

Cross Compiling

Martin Zenzes
martin.zenzes@dfki.de

June 19, 2014

hello_world.cpp

```
#include <iostream>

int main(int argc, char* argv[])
{
    std::cout << "hello world!\n";
    return 0;
}
```

ELF 64-bit LSB executable, x86-64, version 1 (SYSV)

```
00007a0: 910d 6000 e877 feff ffba 680c 6000 be91  ..`..w....h.`...
00007b0: 0d60 00bf 5006 4000 e883 feff ffc9 c355  .`..P.@........U
00007c0: 4889 e5be ffff 0000 bf01 0000 00e8 b0ff  H...............
00007d0: ffff 5dc3 662e 0f1f 8400 0000 0000 6690  ..].f.........f.
00007e0: 4157 4189 ff41 5649 89f6 4155 4989 d541  AWA..AVI..AUI..A
00007f0: 544c 8d25 f801 2000 5548 8d2d 0002 2000  TL.%.. .UH.-.. .
0000800: 534c 29e5 31db 48c1 fd03 4883 ec08 e8cd  SL).1.H...H.....
0000810: fdff ff48 85ed 741e 0f1f 8400 0000 0000  ...H..t.........
0000820: 4c89 ea4c 89f6 4489 ff41 ff14 dc48 83c3  L..L..D..A...H..
0000830: 0148 39eb 75ea 4883 c408 5b5d 415c 415d  .H9.u.H...[]A\A]
0000840: 415e 415f c366 662e 0f1f 8400 0000 0000  A^A_.ff.........
0000850: f3c3 0000 4883 ec08 4883 c408 c300 0000  ....H...H.......
0000860: 0100 0200 6865 6c6c 6f20 776f 726c 6421  ....hello world!
0000870: 0000 0000 011b 033b 4000 0000 0700 0000  .......;@.......
0000880: 8cfd ffff 8c00 0000 fcfd ffff 5c00 0000  ............\...
0000890: e9fe ffff b400 0000 0eff ffff d400 0000  ................
00008a0: 4bff ffff f400 0000 6cff ffff 1401 0000  K.......l.......
00008b0: dcff ffff 5c01 0000 1400 0000 0000 0000  ....\...........
00008c0: 017a 5200 0178 1001 1b0c 0708 9001 0710  .zR..x..........
00008d0: 1400 0000 1c00 0000 98fd ffff 2a00 0000  ............*...
00008e0: 0000 0000 0000 0000 1400 0000 0000 0000  ................
00008f0: 017a 5200 0178 1001 1b0c 0708 9001 0000  .zR..x..........
0000900: 2400 0000 1c00 0000 f8fc ffff 7000 0000  $...........p...
0000910: 000e 1046 0e18 4a0f 0b77 0880 003f 1a3b  ...F..J..w...?.;
```

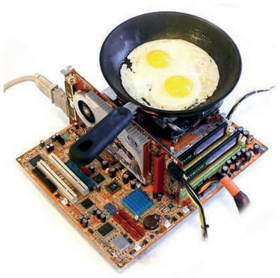
ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV)

```
00006e0: 0300 52e1 0500 001a 1c00 9fe5 8aff ffeb  ..R.............
00006f0: 1400 9fe5 1410 9fe5 1420 9fe5 8fff ffeb  ......... ......
0000700: 04d0 4be2 0088 bde8 ffff 0000 b809 0100  ..K.............
0000710: 2885 0000 2009 0100 0048 2de9 04b0 8de2  (... ....H-.....
0000720: 0100 a0e3 0410 9fe5 e2ff ffeb 0088 bde8  ................
0000730: ffff 0000 f845 2de9 0070 a0e1 4c50 9fe5  .....E-..p..LP..
0000740: 0180 a0e1 4860 9fe5 02a0 a0e1 0550 8fe0  ....H`.......P..
0000750: 60ff ffeb 0660 8fe0 0660 65e0 4661 b0e1  `....`...`e.Fa..
0000760: f885 bd08 0450 45e2 0040 a0e3 0430 b5e5  .....PE..@...0..
0000770: 0700 a0e1 0810 a0e1 0a20 a0e1 0140 84e2  ......... ...@..
0000780: 33ff 2fe1 0600 54e1 f7ff ff1a f885 bde8  3./...T.........
0000790: 8c80 0000 8c80 0000 1eff 2fe1 0840 2de9  ........../..@-.
00007a0: 0880 bde8 0100 0200 6865 6c6c 6f20 776f  ........hello wo
00007b0: 726c 6421 0000 0000 409b 0181 b0b0 8084  rld!....@.......
00007c0: 0000 0000 88fd ff7f 0100 0000 b4fe ff7f  ................
00007d0: e8ff ff7f e4fe ff7f 0100 0000 0000 0000  ................
00007e0: 4c86 0000 1887 0000 2486 0000 0000 0000  L.......$.......
00007f0: 0100 0000 0100 0000 0100 0000 ea00 0000  ................
0000800: 0100 0000 f400 0000 0100 0000 1901 0000  ................
0000810: 0c00 0000 d884 0000 0d00 0000 9c87 0000  ................
0000820: 1900 0000 e007 0100 1b00 0000 0800 0000  ................
0000830: 1a00 0000 e807 0100 1c00 0000 0400 0000  ................
0000840: f5fe ff6f 8c81 0000 0500 0000 9c82 0000  ...o............
0000850: 0600 0000 cc81 0000 0a00 0000 6a01 0000  ............j...
```

Contents

Contents

Introduction



- Heterogeneous environments
- Use case → architecture → machine code

x86	lot of power and volume, fast
ARM	efficient, flexible PCB designs, slow
MIPS	even more efficient, even fewer ressources
$other	interesting quirks, different instruction set

- Some architectures are cumbersome to use
- Native development sometimes impossible
- Not always need for build-environment *inside* target system
- Pre-compiling and binary distribution speeds up co-workers

Introduction

- Heterogeneous environments
- Use case → architecture → machine code

x86	lot of power and volume, fast
ARM	efficient, flexible PCB designs, slow
MIPS	even more efficient, even fewer ressources
$other	interesting quirks, different instruction set

- Some architectures are cumbersome to use
- Native development sometimes impossible
- Not always need for build-environment *inside* target system
- Pre-compiling and binary distribution speeds up co-workers

Virtualization

They use it in the cloud, don't they?

- Simulate target system, with fewer constraints
- Use simulated *native* tools for development
- VirtualBox, LXC, Qemu, VMware...
- Unsupported syscalls, slooooouw

Full system:

- Full OS (kernel+userland)
  Can boot production image
- Filesystem access a hassle
- Reference system to be shared among developers

User mode:

- Detect foreign binaries, emulate on the fly
- Combine with `chroot`, LXC
- Use hosts kernel (+userland)

🔗 `U-Boot + Linux on Qemu`

Contents

Cross-Compiler

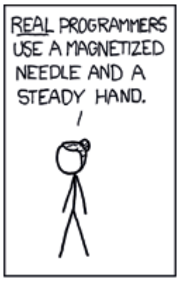
Dammit, just gimme what I need...

Tasks:

- Compile and link on developers system (*host*)
- Create binaries to be executed on *target* system
- Headers and libraries present on the host, suitable for target
- Keep target hardware out of the loop

Get one:

- Ubuntu: shipped by default!
  `sudo apt-get install gcc-arm-linux-gnueabi`
- Debian: from Emdebian repository
- Configure & build your own (buildroot, crosstool-NG, bitbake)



Cross-Compiler

Dammit, just gimme what I need...

Tasks:

- Compile and link on developers system (*host*)
- Create binaries to be executed on *target* system
- Headers and libraries present on the host, suitable for target
- Keep target hardware out of the loop

Get one:

- Ubuntu: shipped by default!
  `sudo apt-get install gcc-arm-linux-gnueabi`
- Debian: from Emdebian repository
- Configure & build your own (buildroot, crosstool-NG, bitbake)

Cross-Compiler

Split development and execution

- Cross-Compiler:

host	System on which the compiler is building software
target	This is the platform it will generate machine code for

- Identify compiler with *triplet*:
  `arch[-vendor][-os]-abi`
- 30 years of erosion: only loose policy survived...

```
guy@host:~$ gcc -dumpmachine
x86_64-linux-gnu
# others (non exhaustive):
arm-linux-gnueabihf
arm-none-eabi
x86_64-pc-cygwin
i386-unknown-openbsd
i686-apple-darwin10-gcc-4.2.1
```

valid for gcc only
different model in clang

Cross-Compiler

Split build, development and execution

- Canadian-Cross-Compiler:

build	System on which the compiler is building a compiler
host	System executing the resulting compiler to compile
target	This is the platform it will generate machine code for

- Identify compiler with *triplet*:
  `arch[-vendor][-os]-abi`
- 30 years of erosion: only loose policy survived...

```
guy@host:~$ gcc -dumpmachine
x86_64-linux-gnu
# others (non exhaustive):
arm-linux-gnueabihf
arm-none-eabi
x86_64-pc-cygwin
i386-unknown-openbsd
i686-apple-darwin10-gcc-4.2.1
```

valid for gcc only
different model in clang

Cross-Compiler

Split build, development and execution

- Canadian-Cross-Compiler:

build	System on which the compiler is building a compiler
host	System executing the resulting compiler to compile
target	This is the platform it will generate machine code for

- Identify compiler with *triplet*:
  `arch[-vendor][-os]-abi`
- 30 years of erosion: only loose policy survived...

```
guy@host:~$ gcc -dumpmachine
x86_64-linux-gnu
# others (non exhaustive):
arm-linux-gnueabihf
arm-none-eabi
x86_64-pc-cygwin
i386-unknown-openbsd
i686-apple-darwin10-gcc-4.2.1
```

valid for `gcc` only
different model in `clang`

Vocabulary

binutils	Tools for creating and managing binaries, object files, libraries, profile data, and assembly code
prefix	Where to install the software in the target system
sysroot	Location for toolchain on the build machine
gnueabihf	*Embedded ABI* for GNU on ARM with hardfloat
bare metal	No OS, binary executed directly on target CPU
native	Binaries which can be executed on *this* system

- Proper compiler location on the host:
  `$SYSROOT/$PREFIX/bin/$TARGET-gcc`
- A *cross*-toolchain cannot be used inside target!

```
guy@host:~/dev$ ls
hello_world.cpp

guy@host:~/dev$ g++ -print-sysroot
/
guy@host:~/dev$ g++ -dumpmachine
x86_64-linux-gnu
guy@host:~/dev$ g++ hello_world.cpp -o hello_world-x86_64
guy@host:~/dev$ file hello_world-x86_64
hello_world-x86_64:  ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=8c2504a1f446f0751b51319fc6fa269513e764c6, not stripped
guy@host:~/dev $ ./hello_world-x86_64
hello world!

guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -print-sysroot
/opt/poky/1.4.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -dumpmachine
arm-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ hello_world.cpp -o hello_world-arm
guy@host:~/dev$ file hello_world-arm
hello_world-arm:  ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16,
BuildID[sha1]=e5d3665c193b9d864378f9b5410ba9bfb564ec66, not stripped
guy@host:~/dev$ qemu-arm-static -L $(arm-poky-linux-gnueabi-g++ \
    -print-sysroot) ./hello_world-arm
hello world!
```

```
guy@host:~/dev$ ls
hello_world.cpp

guy@host:~/dev$ g++ -print-sysroot
/
guy@host:~/dev$ g++ -dumpmachine
x86_64-linux-gnu
guy@host:~/dev$ g++ hello_world.cpp -o hello_world-x86_64
guy@host:~/dev$ file hello_world-x86_64
hello_world-x86_64:  ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=8c2504a1f446f0751b51319fc6fa269513e764c6, not stripped
guy@host:~/dev $ ./hello_world-x86_64
hello world!

guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -print-sysroot
/opt/poky/1.4.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -dumpmachine
arm-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ hello_world.cpp -o hello_world-arm
guy@host:~/dev$ file hello_world-arm
hello_world-arm:  ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16,
BuildID[sha1]=e5d3665c193b9d864378f9b5410ba9bfb564ec66, not stripped
guy@host:~/dev$ qemu-arm-static -L $(arm-poky-linux-gnueabi-g++ \
    -print-sysroot) ./hello_world-arm
hello world!
```

```
guy@host:~/dev$ ls
hello_world.cpp

guy@host:~/dev$ g++ -print-sysroot
/
guy@host:~/dev$ g++ -dumpmachine
x86_64-linux-gnu
guy@host:~/dev$ g++ hello_world.cpp -o hello_world-x86_64
guy@host:~/dev$ file hello_world-x86_64
hello_world-x86_64:  ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=8c2504a1f446f0751b51319fc6fa269513e764c6, not stripped
guy@host:~/dev $ ./hello_world-x86_64
hello world!

guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -print-sysroot
/opt/poky/1.4.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -dumpmachine
arm-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ hello_world.cpp -o hello_world-arm
guy@host:~/dev$ file hello_world-arm
hello_world-arm:  ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16,
BuildID[sha1]=e5d3665c193b9d864378f9b5410ba9bfb564ec66, not stripped
guy@host:~/dev$ qemu-arm-static -L $(arm-poky-linux-gnueabi-g++ \
    -print-sysroot) ./hello_world-arm
hello world!
```

```
guy@host:~/dev$ ls
hello_world.cpp

guy@host:~/dev$ g++ -print-sysroot
/
guy@host:~/dev$ g++ -dumpmachine
x86_64-linux-gnu
guy@host:~/dev$ g++ hello_world.cpp -o hello_world-x86_64
guy@host:~/dev$ file hello_world-x86_64
hello_world-x86_64:  ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=8c2504a1f446f0751b51319fc6fa269513e764c6, not stripped
guy@host:~/dev $ ./hello_world-x86_64
hello world!

guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -print-sysroot
/opt/poky/1.4.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -dumpmachine
arm-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ hello_world.cpp -o hello_world-arm
guy@host:~/dev$ file hello_world-arm
hello_world-arm:  ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16,
BuildID[sha1]=e5d3665c193b9d864378f9b5410ba9bfb564ec66, not stripped
guy@host:~/dev$ qemu-arm-static -L $(arm-poky-linux-gnueabi-g++ \
    -print-sysroot) ./hello_world-arm
hello world!
```

```
guy@host:~/dev$ ls
hello_world.cpp

guy@host:~/dev$ g++ -print-sysroot
/
guy@host:~/dev$ g++ -dumpmachine
x86_64-linux-gnu
guy@host:~/dev$ g++ hello_world.cpp -o hello_world-x86_64
guy@host:~/dev$ file hello_world-x86_64
hello_world-x86_64:  ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=8c2504a1f446f0751b51319fc6fa269513e764c6, not stripped
guy@host:~/dev $ ./hello_world-x86_64
hello world!

guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -print-sysroot
/opt/poky/1.4.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ -dumpmachine
arm-poky-linux-gnueabi
guy@host:~/dev$ arm-poky-linux-gnueabi-g++ hello_world.cpp -o hello_world-arm
guy@host:~/dev$ file hello_world-arm
hello_world-arm:  ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16,
BuildID[sha1]=e5d3665c193b9d864378f9b5410ba9bfb564ec66, not stripped
guy@host:~/dev$ qemu-arm-static -L $(arm-poky-linux-gnueabi-g++ \
    -print-sysroot) ./hello_world-arm
hello world!
```

Contents

You need more than a Cross-Compiler

Configure, compile, link, test, deploy, execute...

Piece it all together:

- Initialize environment variables: `$CC`, `$CFLAGS` etc
- provide build-time dependencies
- compile and link into executables
- create binary packages of generated software
- check whole process for common pitfalls
- prevent native tools from creeping in
- provide simulated environment for testing

This quickly becomes tedious...

You need more than a Cross-Compiler

Configure, compile, link, test, deploy, execute...

Piece it all together:

- Initialize environment variables: `$CC`, `$CFLAGS` etc
- provide build-time dependencies
- compile and link into executables
- create binary packages of generated software
- check whole process for common pitfalls
- prevent native tools from creeping in
- provide simulated environment for testing

This quickly becomes tedious...

Questions? Remarks?

2.3 ‘The Artemis Rover as an Example for Model Based Engineering in Space Robotics’ (FS-T-03)

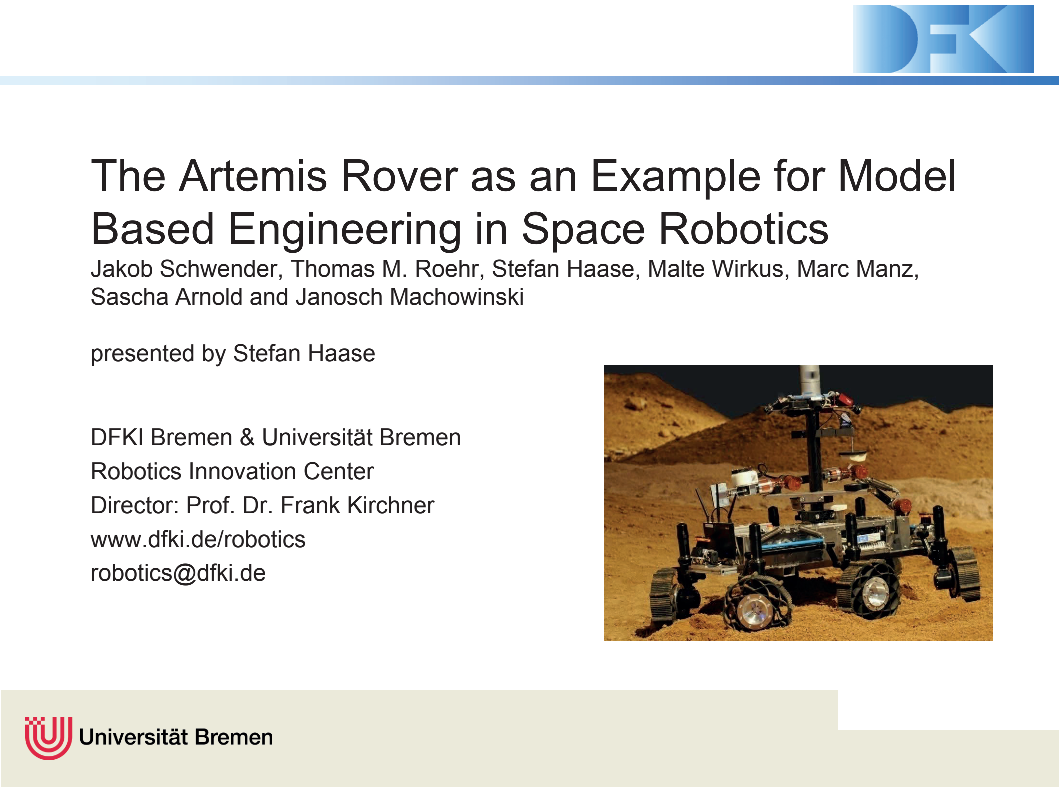
Stefan Haase (1), *Thomas M. Roehr* (1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `stefan.haase@dfki.de`, `thomas.roehr@dfki.de`

Abstract

Future application of robotic missions in the space context will require the systems to have both mobility and manipulation capabilities. The limited direct communication with the systems due to visibility, and severe time delays also make it a requirement for the system to perform its actions mainly autonomously. The increasing complexity of the task, as well as the strict requirements for reliability and fault tolerance pose a significant challenge to both engineering and research activities. The SpaceBot Cup was held in November 2013 to probe those capabilities in the context of a competition. In this paper we present the Artemis rover and its software architecture as well as the competition results and lessons learned. Special attention is given to the modular design based on the Robot Construction Kit (Rock); a component based software framework, which uses a component model based on the Orocos Real-Time-Toolkit (RTT).



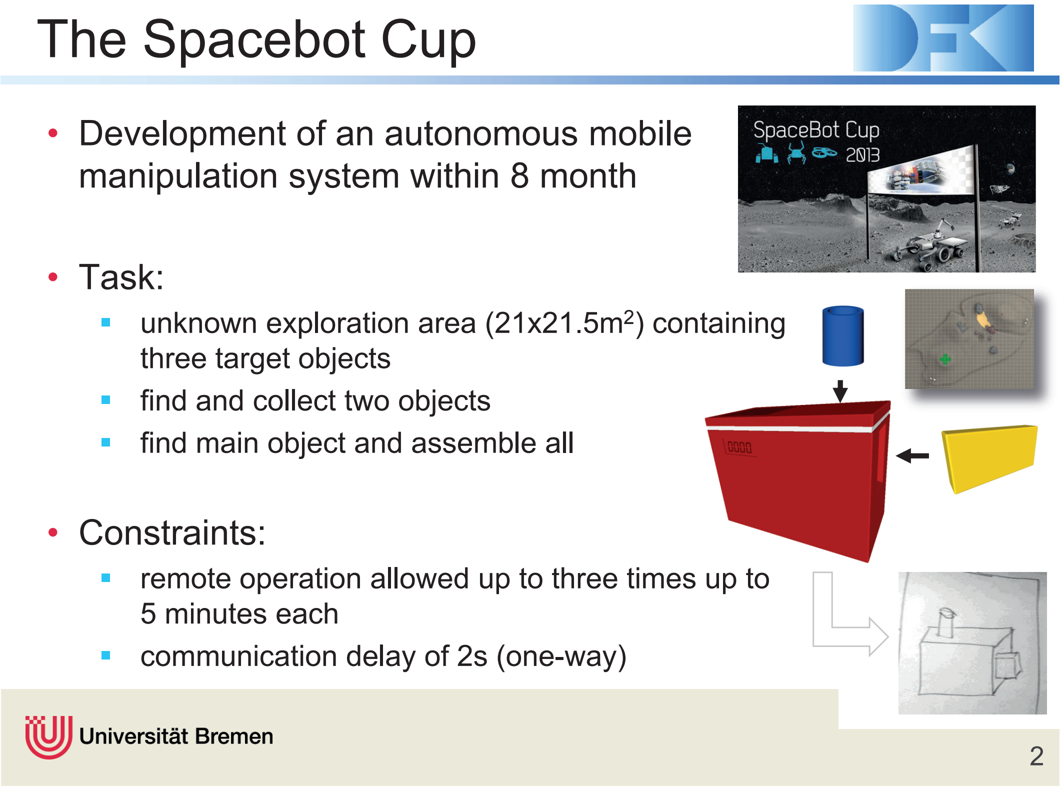
The Artemis Rover as an Example for Model Based Engineering in Space Robotics

Jakob Schwender, Thomas M. Roehr, Stefan Haase, Malte Wirkus, Marc Manz,
Sascha Arnold and Janosch Machowinski

presented by Stefan Haase

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



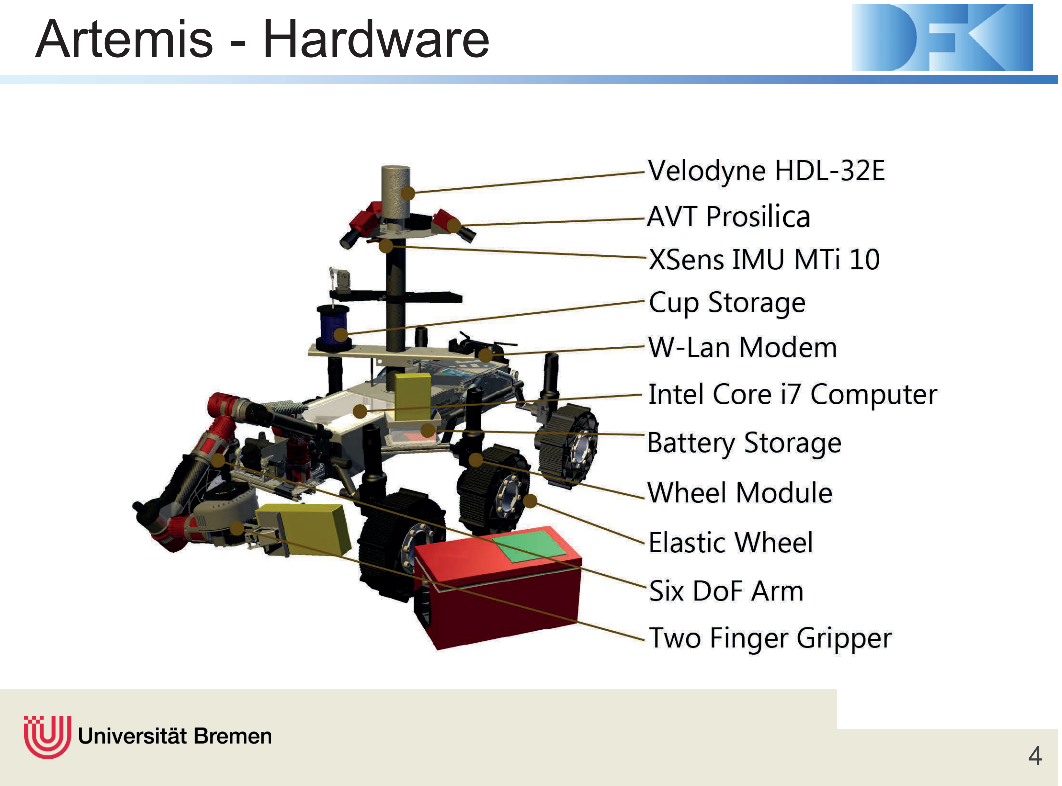
The Spacebot Cup

- Development of an autonomous mobile manipulation system within 8 month
- Task:
  - unknown exploration area (21x21.5m^2) containing three target objects
  - find and collect two objects
  - find main object and assemble all
- Constraints:
  - remote operation allowed up to three times up to 5 minutes each
  - communication delay of 2s (one-way)

Universität Bremen

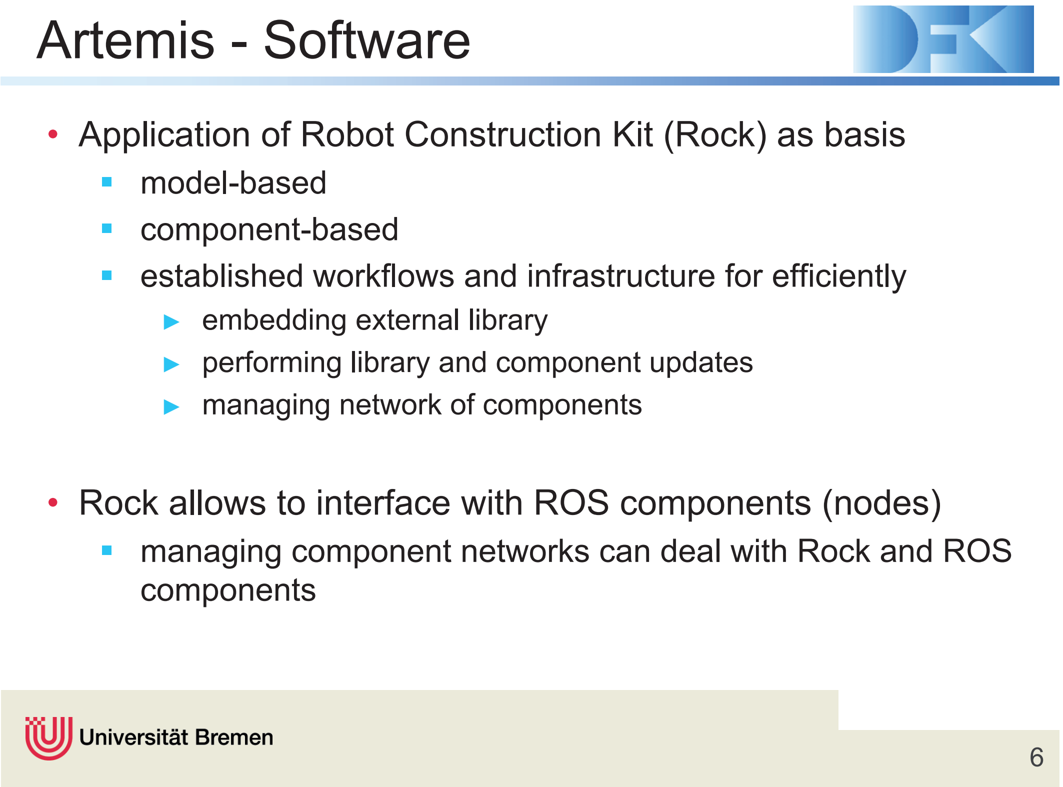
Development approach

- Top down
    - high-level mission decomposition and identification of required capabilities
    - distribution of tasks to specialized (sub-)teams
    - maximization of component and library reuse
- Main development lines / (sub) teams
    - Hardware
        - Arm
        - Manipulator
        - Rover
        - Wheels
    - Software
        - Navigation
        - Manipulation
        - Exploration
        - Object detection
        - Integration



Artemis - Hardware

Velodyne HDL-32E
AVT Prosilica
XSens IMU MTi 10
Cup Storage
W-Lan Modem
Intel Core i7 Computer
Battery Storage
Wheel Module
Elastic Wheel
Six DoF Arm
Two Finger Gripper

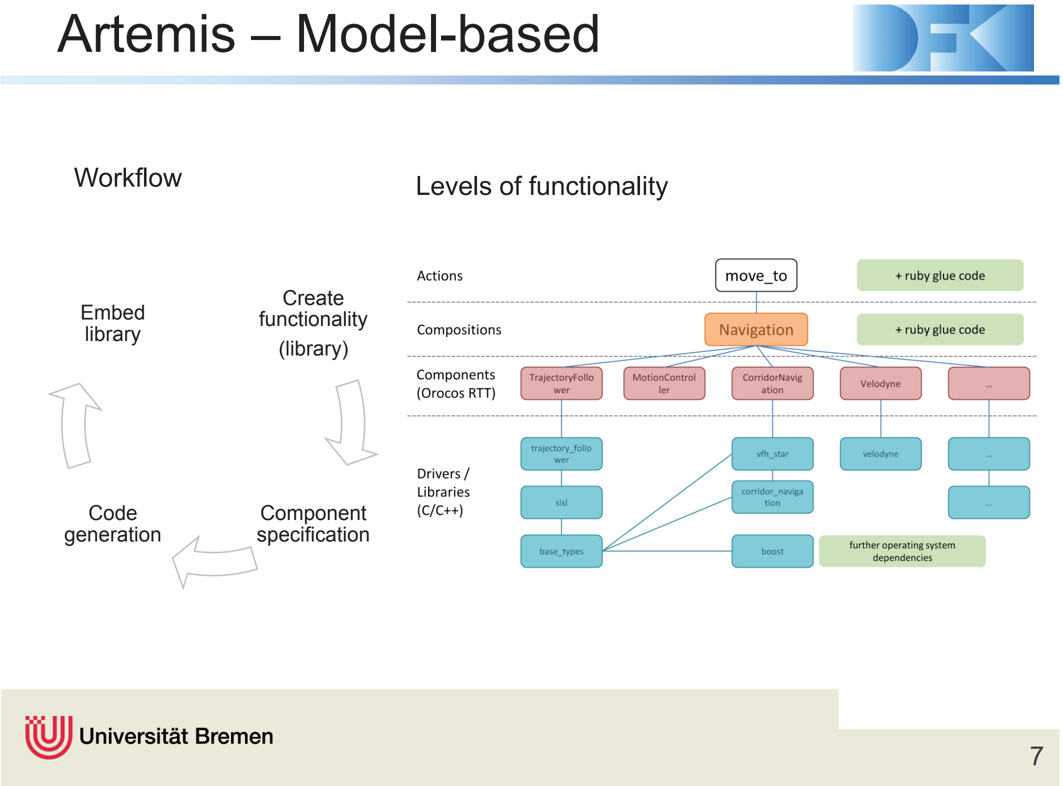


Artemis - Software

- Application of Robot Construction Kit (Rock) as basis
  - model-based
  - component-based
  - established workflows and infrastructure for efficiently
    - embedding external library
    - performing library and component updates
    - managing network of components
- Rock allows to interface with ROS components (nodes)
  - managing component networks can deal with Rock and ROS components

Universität Bremen

6



Artemis – Model-based

Workflow

Embed library

Create functionality (library)

Component specification

Code generation

Levels of functionality

Actions: move_to; + ruby glue code

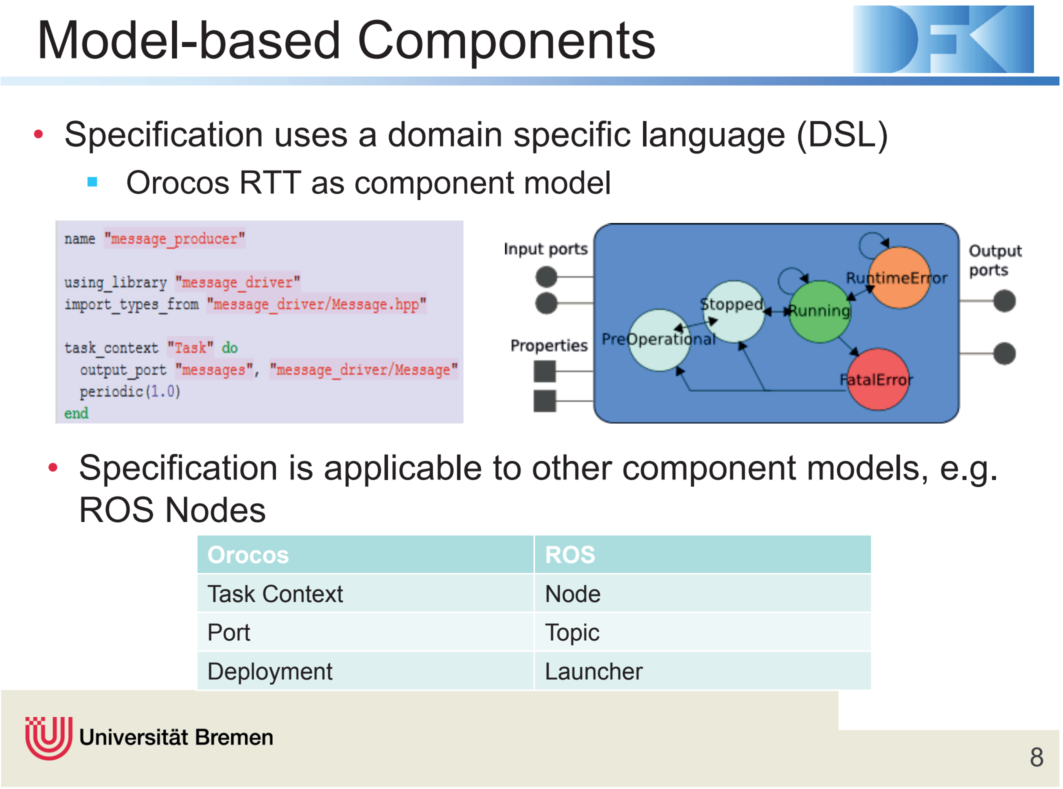
Compositions: Navigation; + ruby glue code

Components (Orocos RTT): TrajectoryFollower; MotionController; CorridorNavigation; Velodyne; ...

Drivers / Libraries (C/C++): trajectory_follower; sisl; base_types; vfh_star; corridor_navigation; boost; velodyne; ...; ...; further operating system dependencies

Universität Bremen

7



Model-based Components

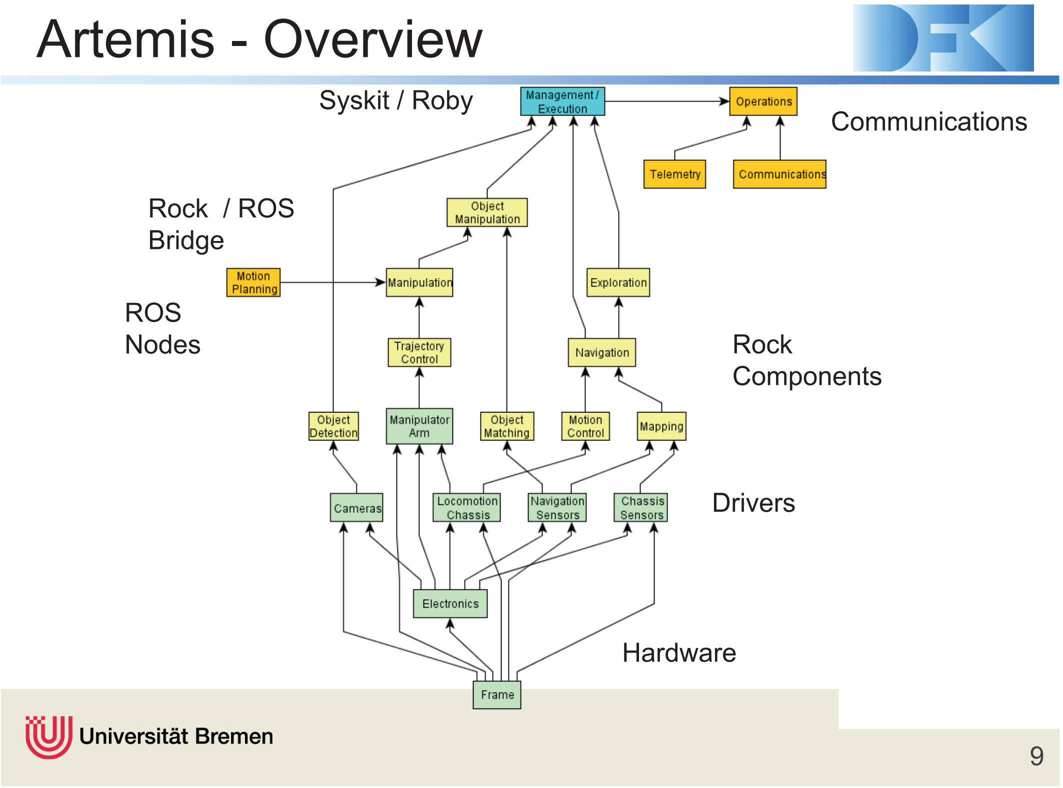
- Specification uses a domain specific language (DSL)
  - Orocos RTT as component model

```
name "message_producer"

using_library "message_driver"
import_types_from "message_driver/Message.hpp"

task_context "Task" do
  output_port "messages", "message_driver/Message"
  periodic(1.0)
end
```

- Specification is applicable to other component models, e.g. ROS Nodes

Orocos	ROS
Task Context	Node
Port	Topic
Deployment	Launcher

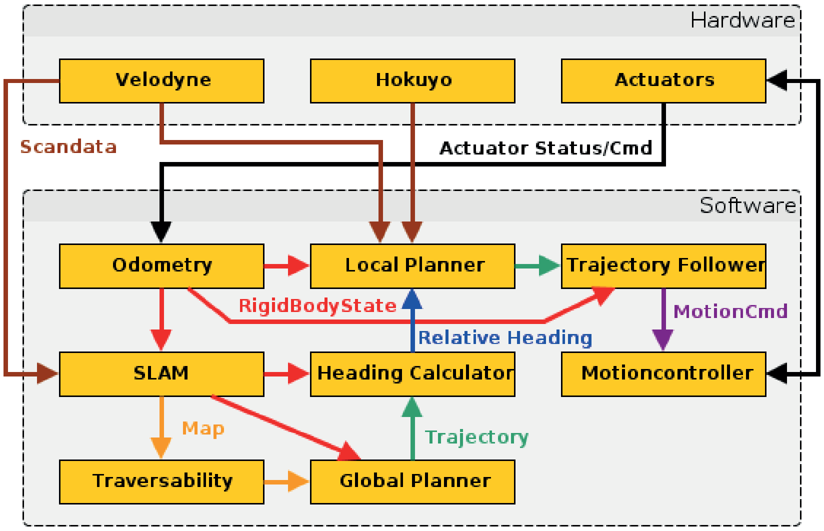
Universität Bremen



Artemis - Overview

Universität Bremen

Navigation



Hardware

Velodyne

Hokuyo

Actuators

Scandata

Actuator Status/Cmd

Software

Odometry

Local Planner

Trajectory Follower

RigidBodyState

MotionCmd

Relative Heading

SLAM

Heading Calculator

Motioncontroller

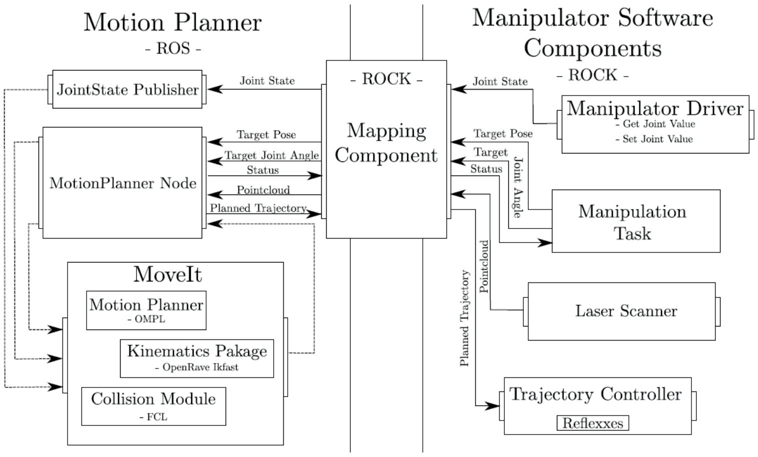
Map

Trajectory

Traversability

Global Planner

Manipulation



Motion Planner

- ROS -

JointState Publisher

MotionPlanner Node

MoveIt

Motion Planner
- OMPL

Kinematics Pakage
- OpenRave Ikfast

Collision Module
- FCL

Joint State

Target Pose

Target Joint Angle

Status

Pointcloud

Planned Trajectory

- ROCK -

Mapping Component

Manipulator Software Components

- ROCK -

Joint State

Target Pose

Target Joint Angle

Status

Pointcloud

Planned Trajectory

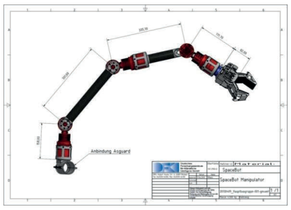
Manipulator Driver
- Get Joint Value
- Set Joint Value

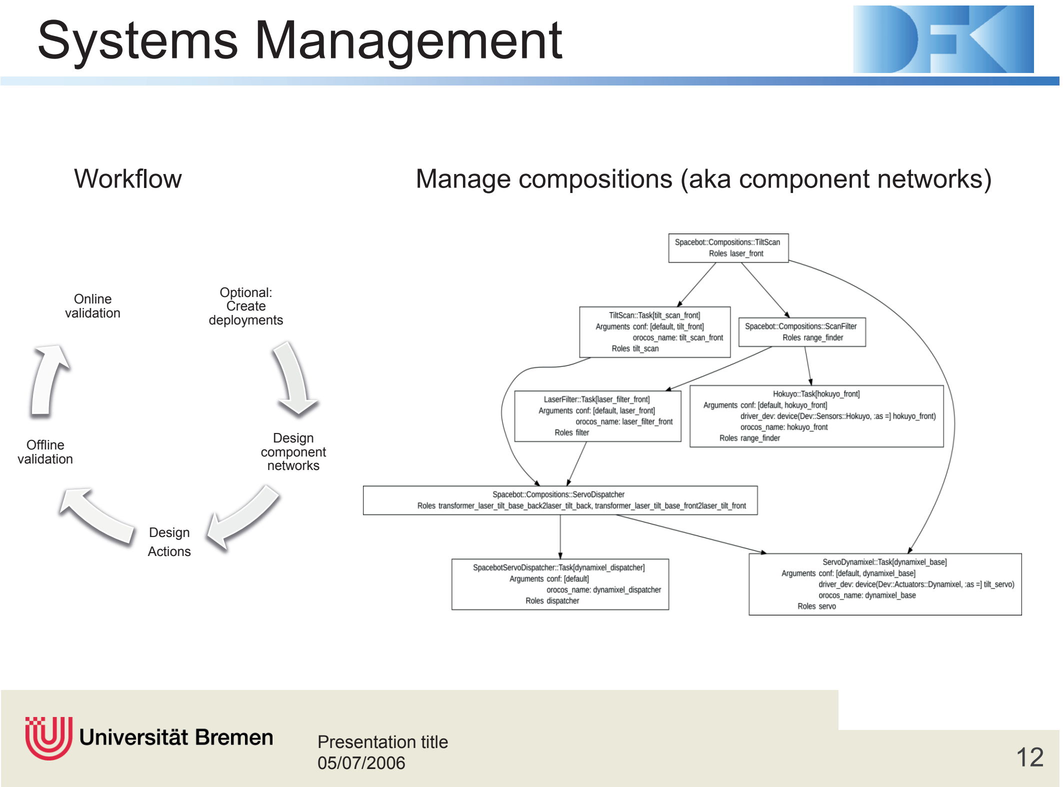
Manipulation Task

Laser Scanner

Trajectory Controller

Reflexxes





Systems Management

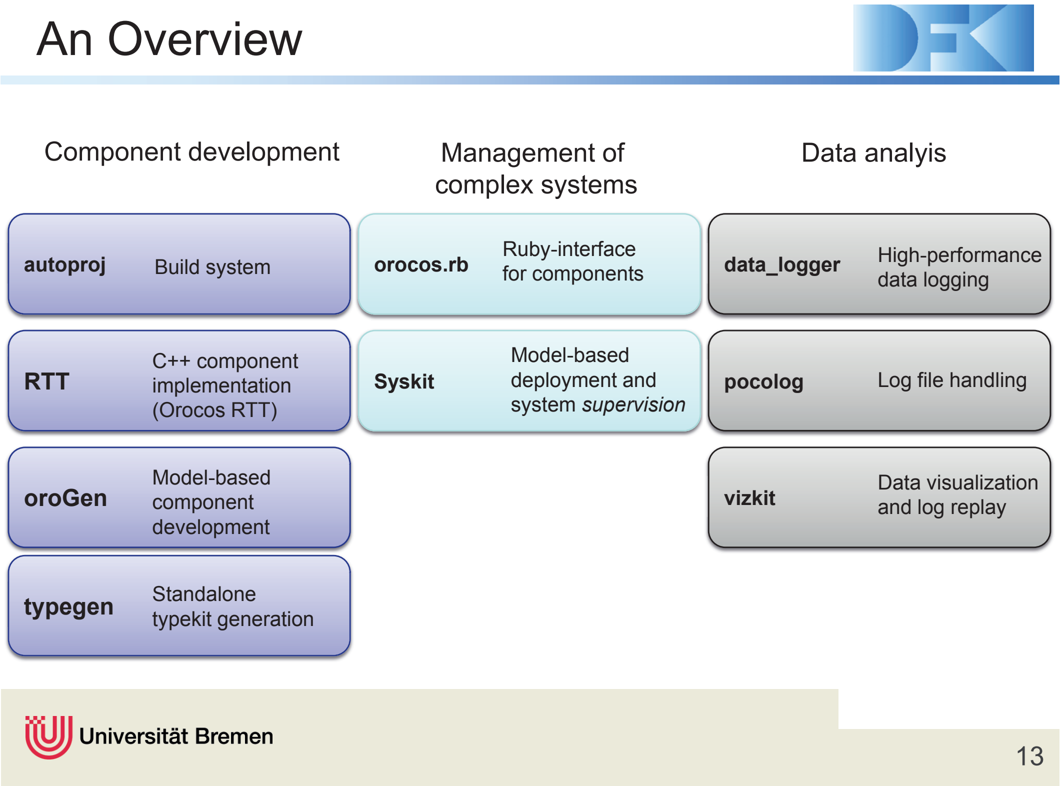
Workflow

Manage compositions (aka component networks)

Universität Bremen

Presentation title
05/07/2006

12



An Overview

Component development

- **autoproj** Build system
- **RTT** C++ component implementation (Orocos RTT)
- **oroGen** Model-based component development
- **typegen** Standalone typekit generation

Management of complex systems

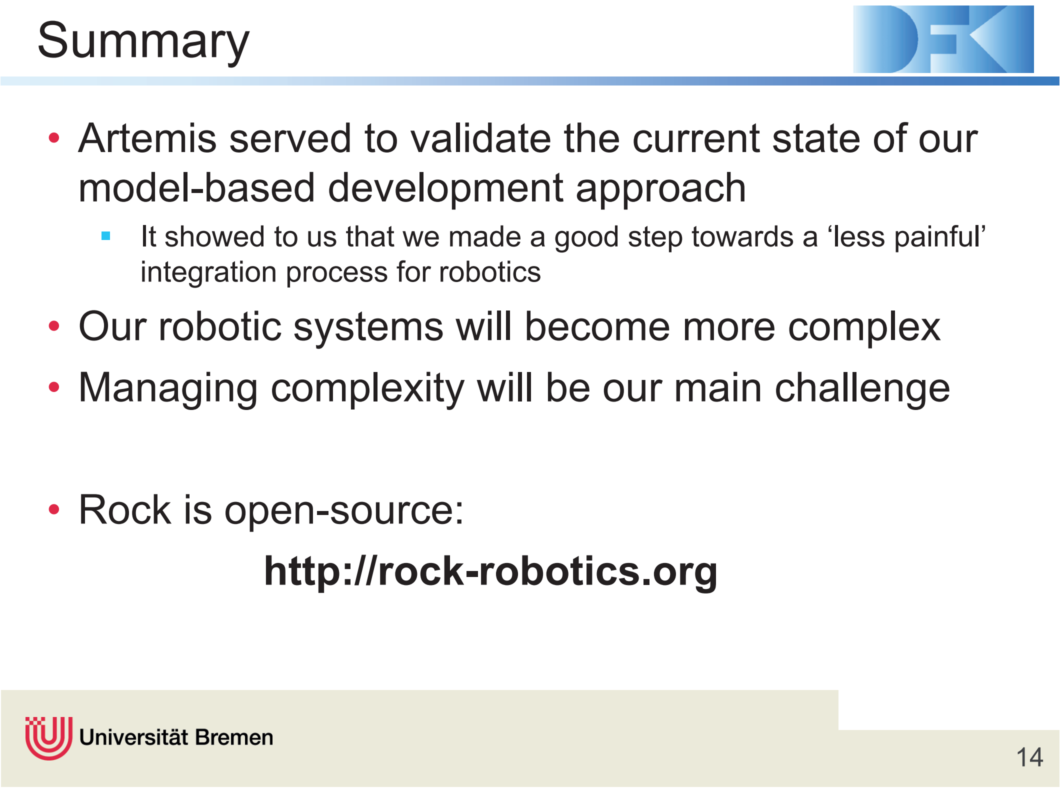
- **orocos.rb** Ruby-interface for components
- **Syskit** Model-based deployment and system *supervision*

Data analyis

- **data_logger** High-performance data logging
- **pocolog** Log file handling
- **vizkit** Data visualization and log replay

Universität Bremen

13



Summary

- Artemis served to validate the current state of our model-based development approach
  - It showed to us that we made a good step towards a 'less painful' integration process for robotics
- Our robotic systems will become more complex
- Managing complexity will be our main challenge

- Rock is open-source:

http://rock-robotics.org

Universität Bremen

Outlook

- DLR announced the next SpaceBot Cup in 2015!!
- Possibility to test the systems in a Mars like environment in October 2014 in Noordwijk (ESA's ESTEC technical centre)

Universität Bremen

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

2.4 ‘Distributed compilation using IceCC’ (FS-T-04)

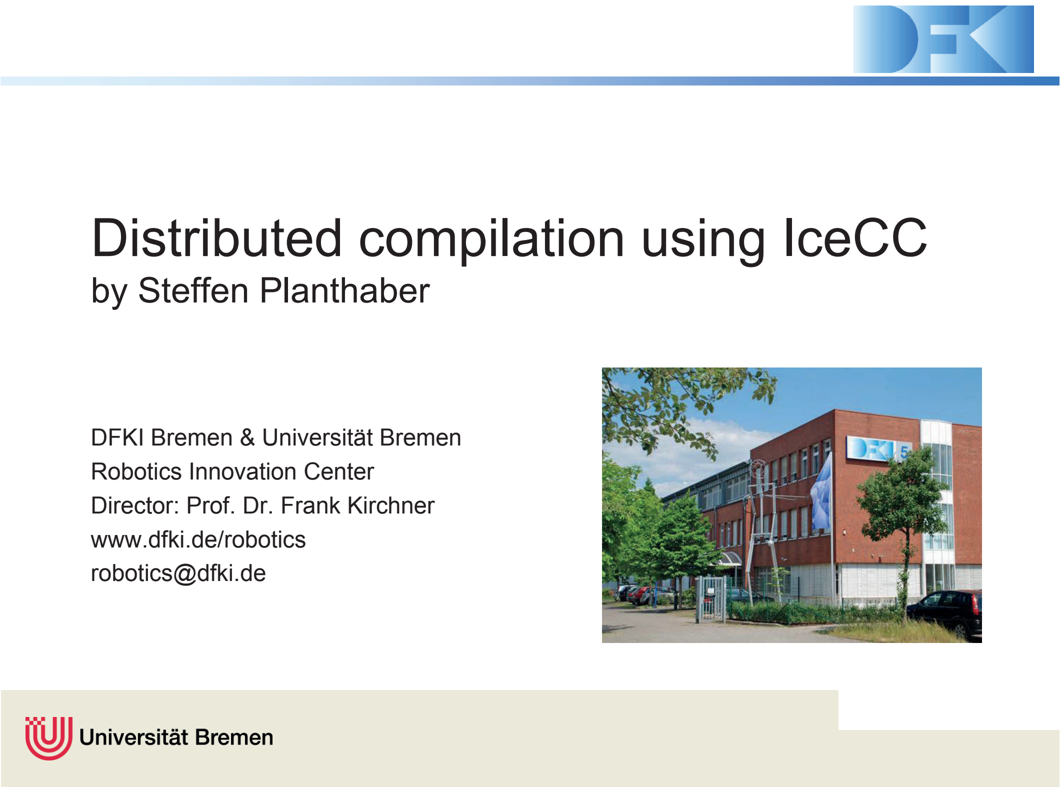
Steffen Planthaber⁽¹⁾

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `steffen.planthaber@dfki.de`

Abstract

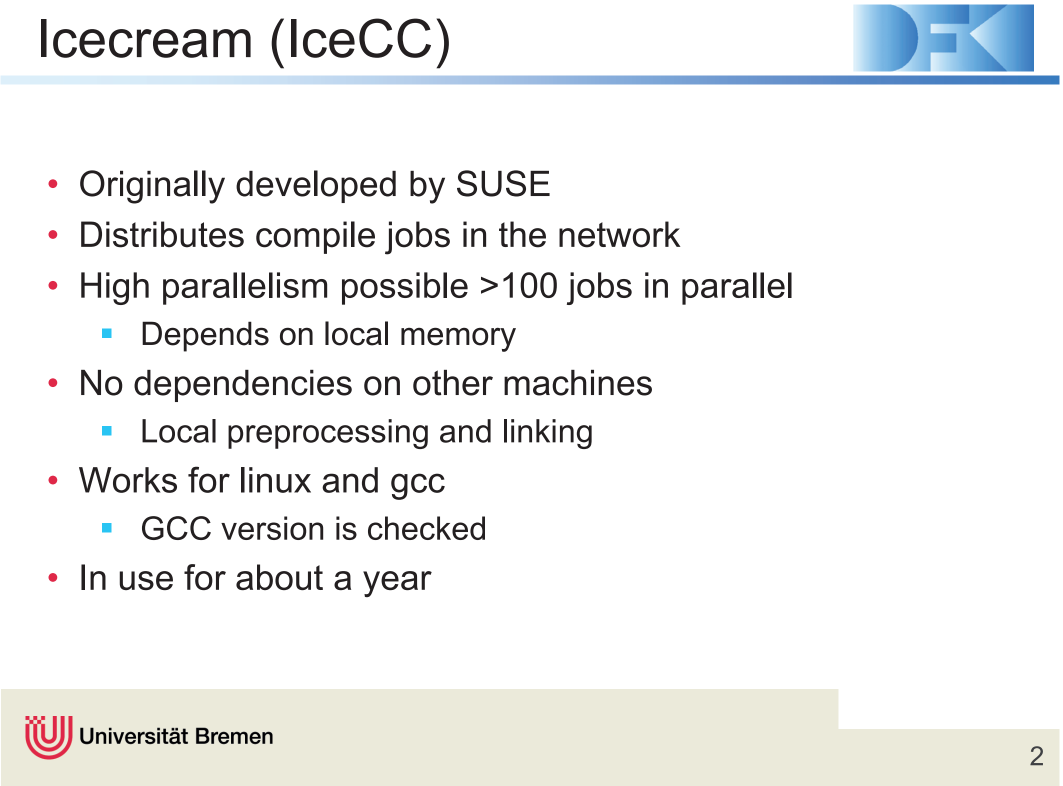
This presentation details the usage of IceCC. IceCC serve as distributed compilation platform and can speed up the compilation process using a cloud-like service. Details are provided on how to monitor IceCC which runs as a system service and how to tune it to maximize compilation speed.



Distributed compilation using IceCC

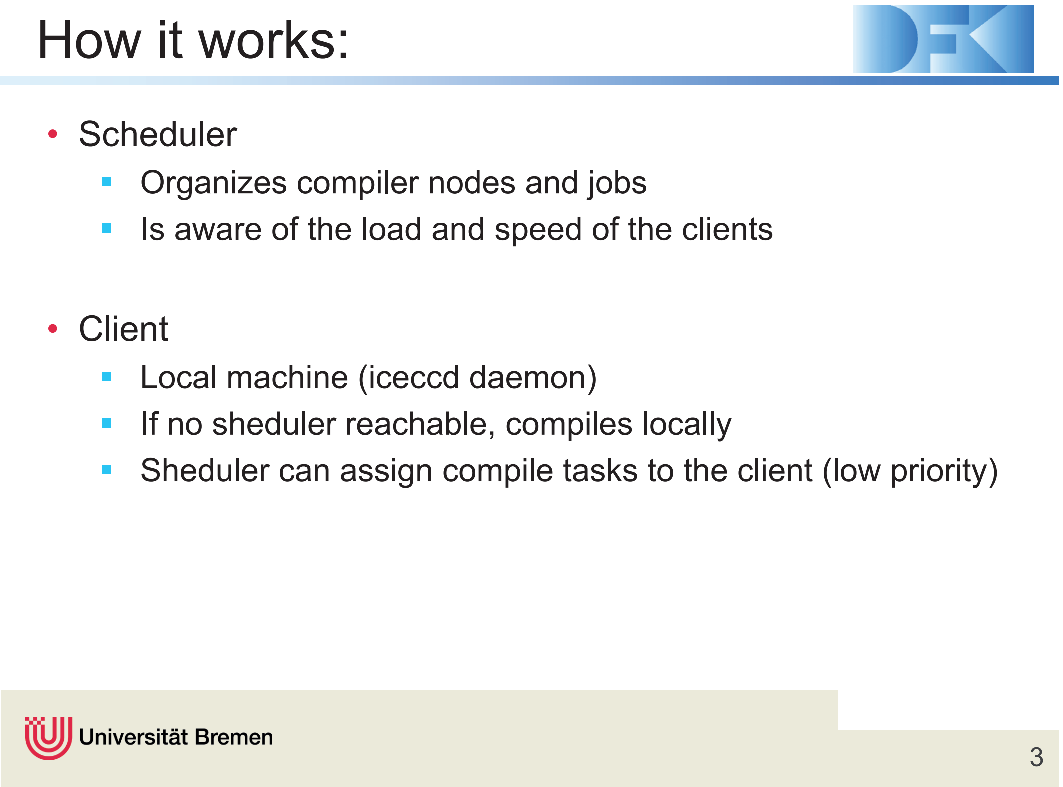
by Steffen Planthaber

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de



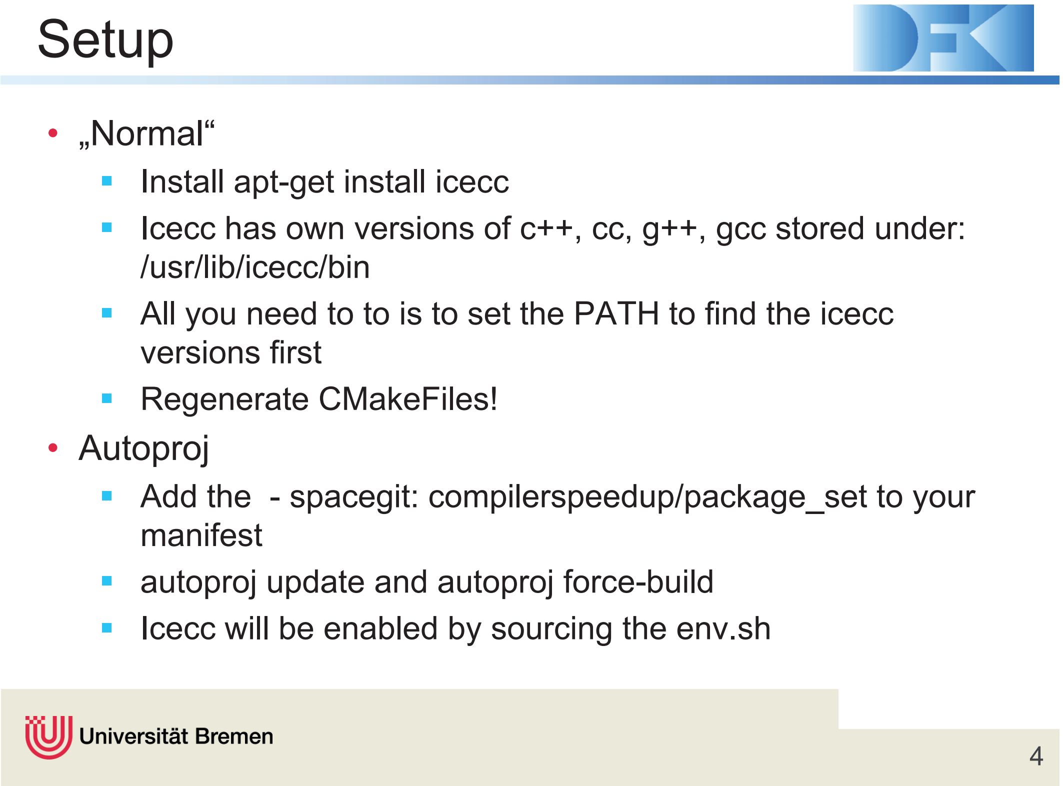
Icecream (IceCC)

- Originally developed by SUSE
- Distributes compile jobs in the network
- High parallelism possible >100 jobs in parallel
  - Depends on local memory
- No dependencies on other machines
  - Local preprocessing and linking
- Works for linux and gcc
  - GCC version is checked
- In use for about a year



How it works:

- Scheduler
  - Organizes compiler nodes and jobs
  - Is aware of the load and speed of the clients
- Client
  - Local machine (iceccd daemon)
  - If no sheduler reachable, compiles locally
  - Sheduler can assign compile tasks to the client (low priority)

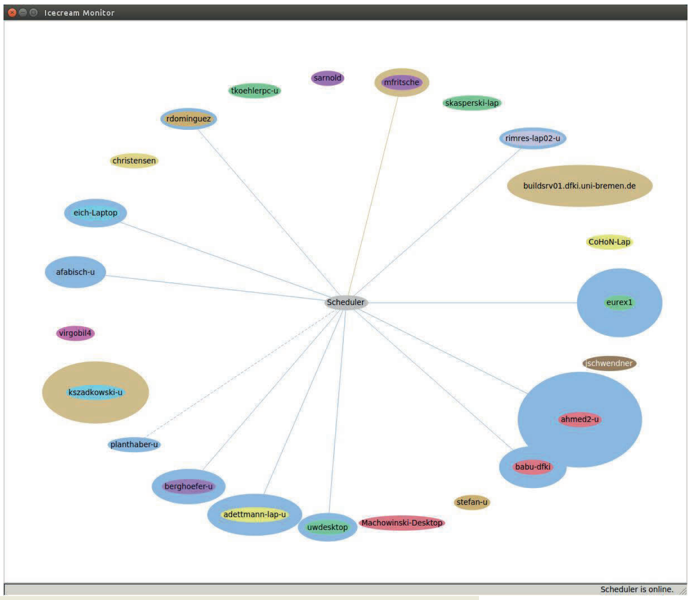


Setup

- „Normal“
  - Install apt-get install icecc
  - Icecc has own versions of c++, cc, g++, gcc stored under: /usr/lib/icecc/bin
  - All you need to to is to set the PATH to find the icecc versions first
  - Regenerate CMakeFiles!
- Autoproj
  - Add the - spacegit: compilerspeedup/package_set to your manifest
  - autoproj update and autoproj force-build
  - Icecc will be enabled by sourcing the env.sh

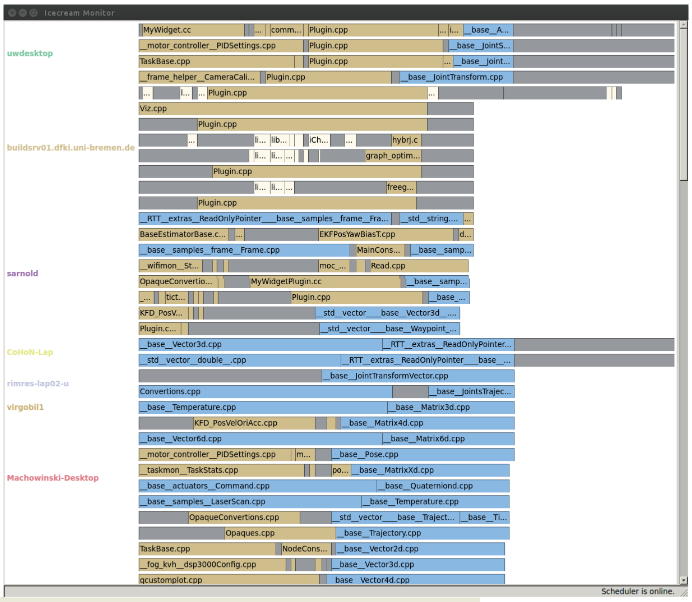
icemon

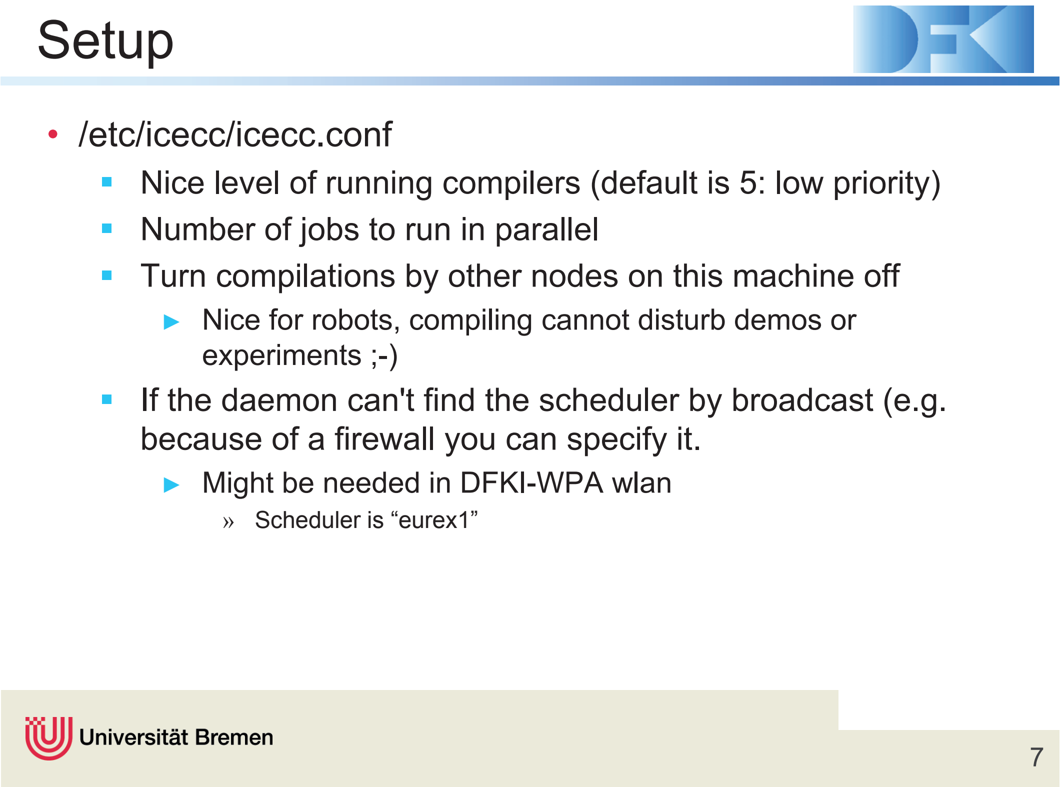
- Size defines the current load
- Color the origin



icemon

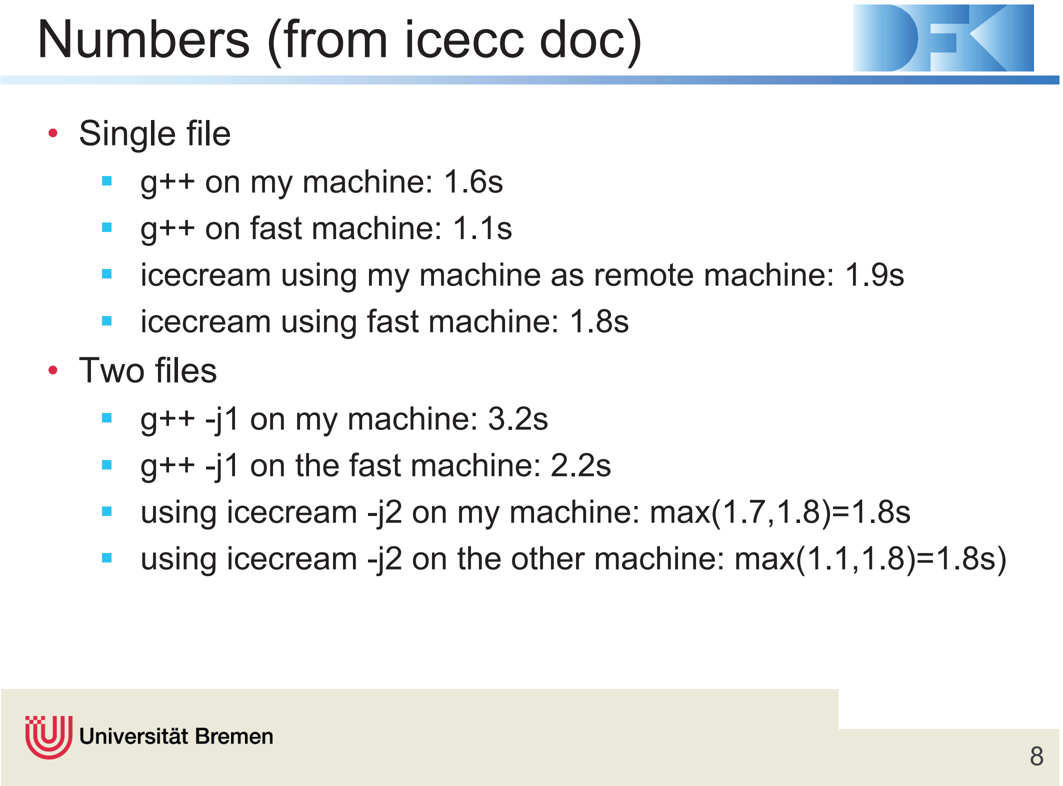
- Color: Origin of files compiled by node
- White: Linking





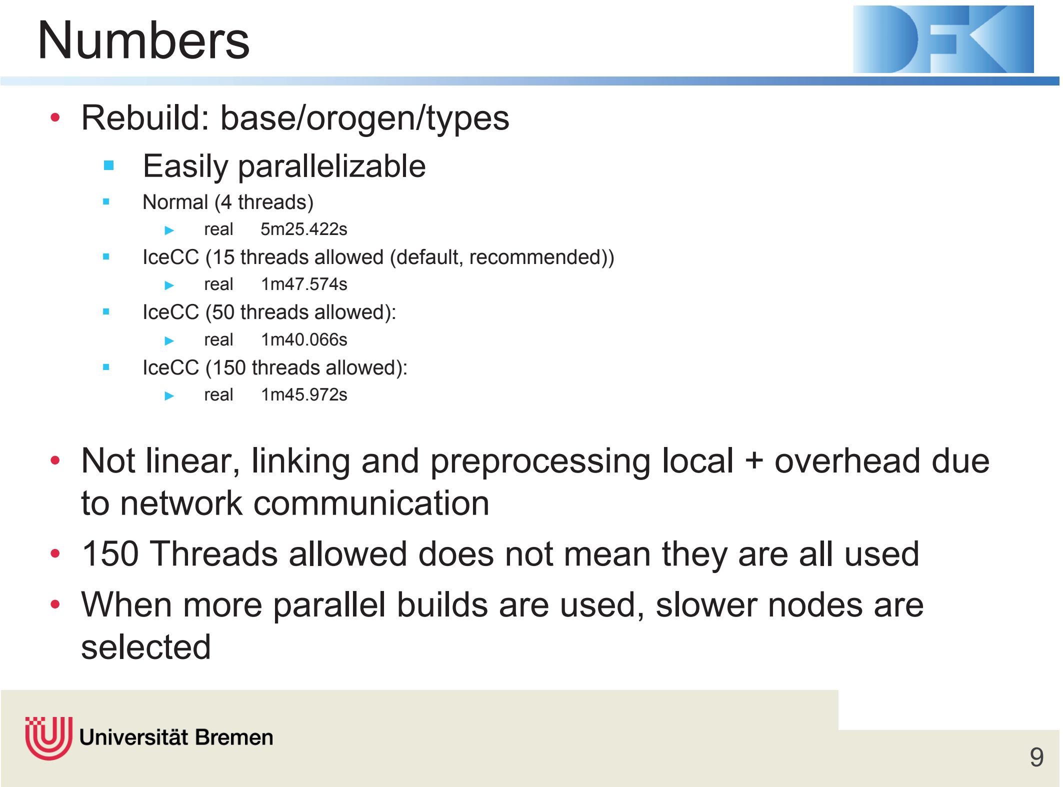
Setup

- /etc/icecc/icecc.conf
  - Nice level of running compilers (default is 5: low priority)
  - Number of jobs to run in parallel
  - Turn compilations by other nodes on this machine off
    - Nice for robots, compiling cannot disturb demos or experiments ;-)
  - If the daemon can't find the scheduler by broadcast (e.g. because of a firewall you can specify it.
    - Might be needed in DFKI-WPA wlan
      - Scheduler is “eurex1”



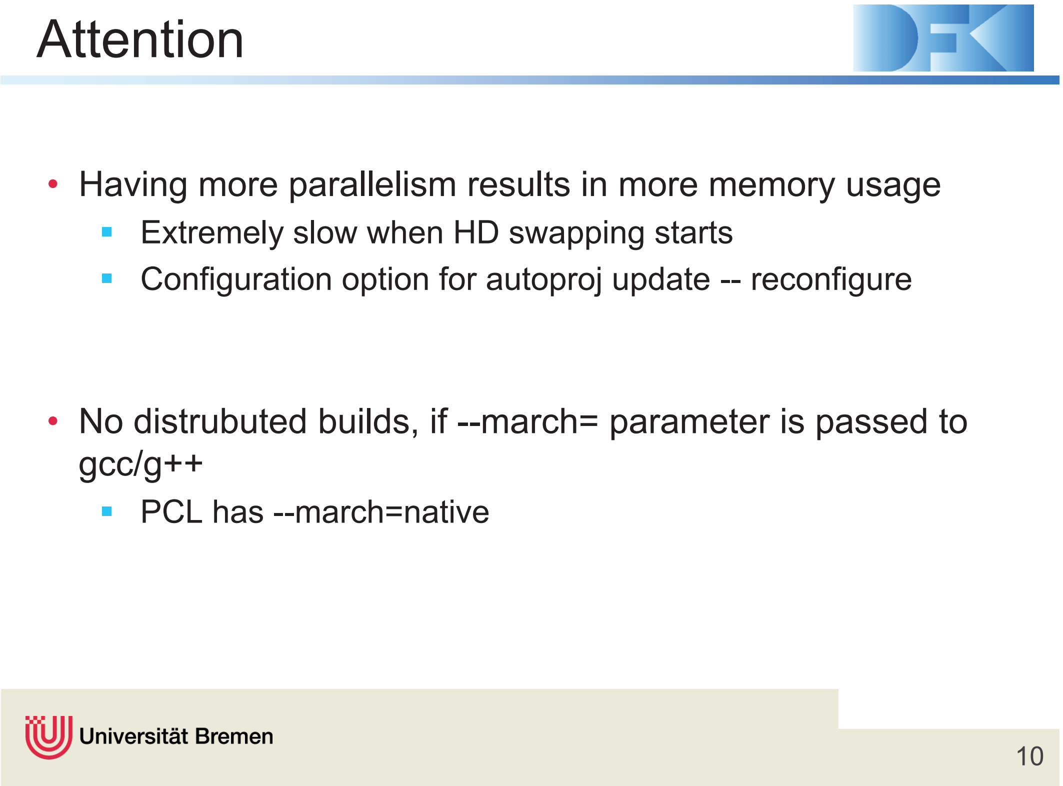
Numbers (from icecc doc)

- Single file
  - g++ on my machine: 1.6s
  - g++ on fast machine: 1.1s
  - icecream using my machine as remote machine: 1.9s
  - icecream using fast machine: 1.8s
- Two files
  - g++ -j1 on my machine: 3.2s
  - g++ -j1 on the fast machine: 2.2s
  - using icecream -j2 on my machine: max(1.7,1.8)=1.8s
  - using icecream -j2 on the other machine: max(1.1,1.8)=1.8s)



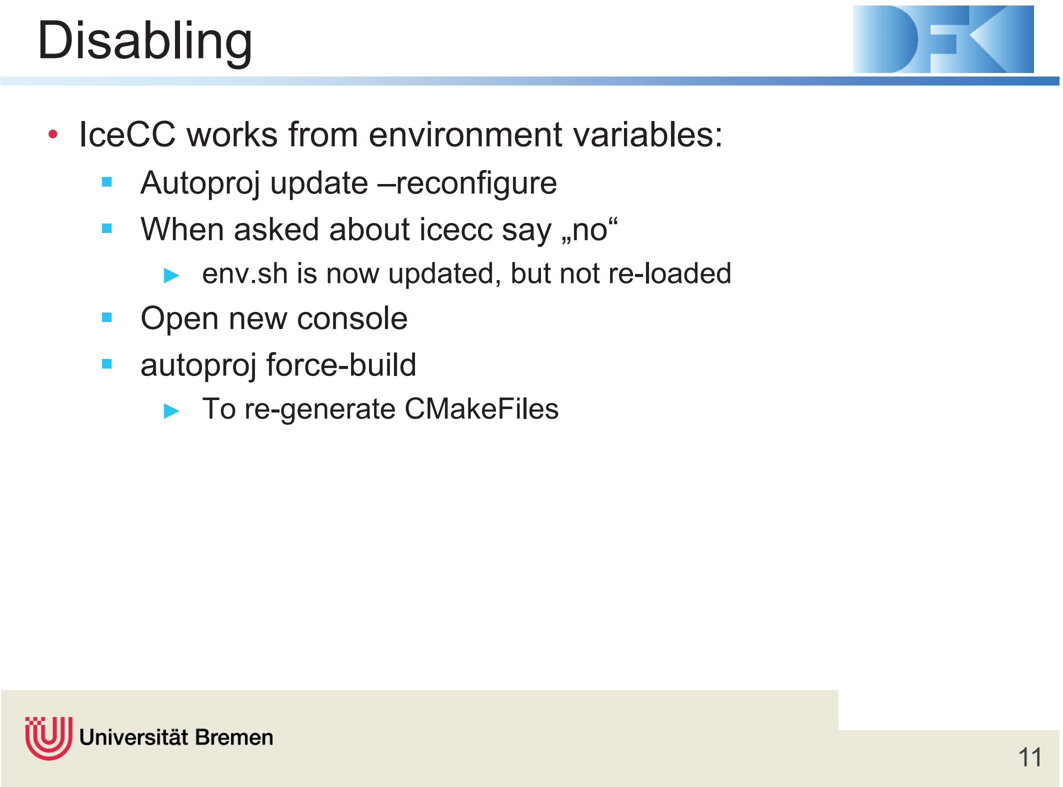
Numbers

- Rebuild: base/orogen/types
  - Easily parallelizable
  - Normal (4 threads)
    - real 5m25.422s
  - IceCC (15 threads allowed (default, recommended))
    - real 1m47.574s
  - IceCC (50 threads allowed):
    - real 1m40.066s
  - IceCC (150 threads allowed):
    - real 1m45.972s
- Not linear, linking and preprocessing local + overhead due to network communication
- 150 Threads allowed does not mean they are all used
- When more parallel builds are used, slower nodes are selected



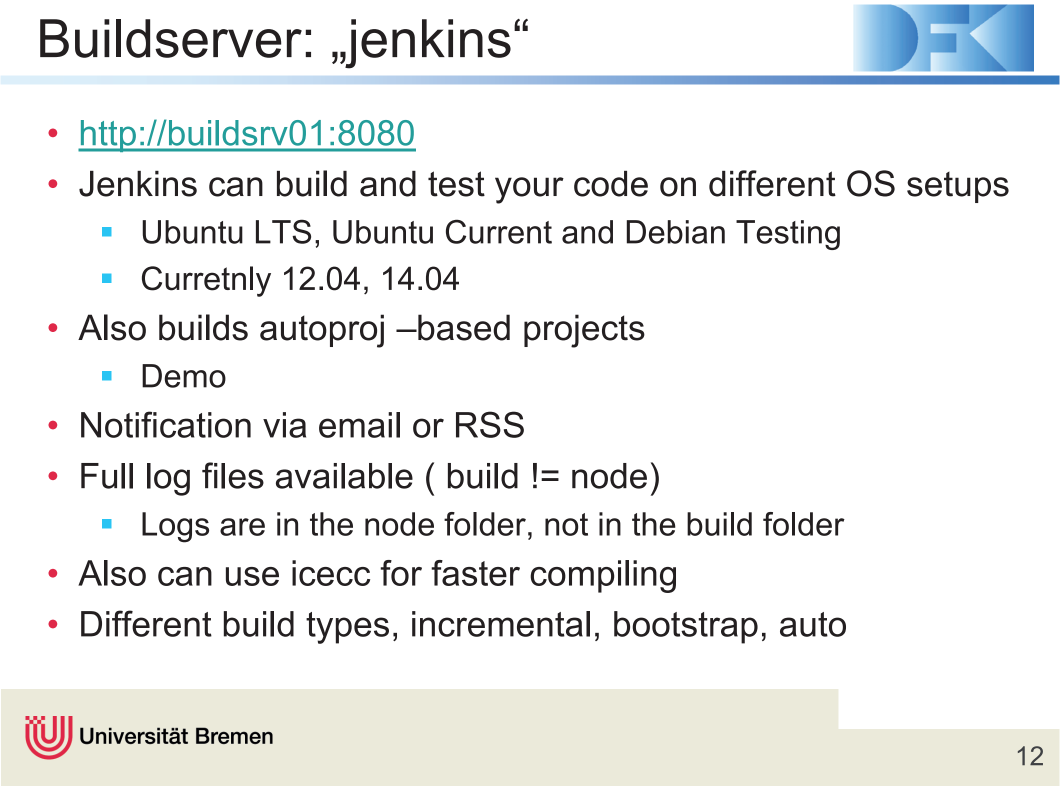
Attention

- Having more parallelism results in more memory usage
  - Extremely slow when HD swapping starts
  - Configuration option for autoproj update -- reconfigure
- No distrubuted builds, if --march= parameter is passed to gcc/g++
  - PCL has --march=native



Disabling

- IceCC works from environment variables:
  - Autoproj update –reconfigure
  - When asked about icecc say „no“
    - env.sh is now updated, but not re-loaded
  - Open new console
  - autoproj force-build
    - To re-generate CMakeFiles



Buildserver: „jenkins“

- http://buildsrv01:8080
- Jenkins can build and test your code on different OS setups
  - Ubuntu LTS, Ubuntu Current and Debian Testing
  - Curretnly 12.04, 14.04
- Also builds autoproj –based projects
  - Demo
- Notification via email or RSS
- Full log files available (build != node)
  - Logs are in the node folder, not in the build folder
- Also can use icecc for faster compiling
- Different build types, incremental, bootstrap, auto

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

2.5 'Data Distribution Service (DDS)' (FS-T-05)

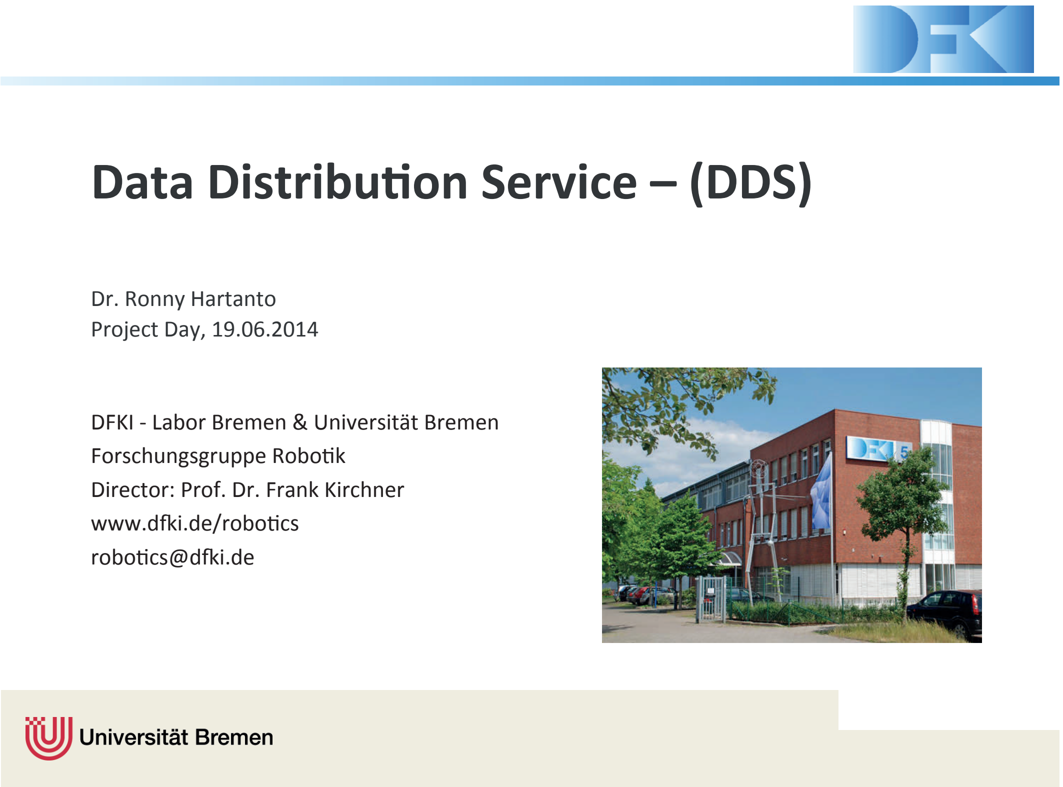
Ronny Hartanto (1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `ronny.hartanto@dfki.de`

Abstract

This presentation describes the application of the Data Distribution Service (DDS) to establish a multi-robot communication infrastructure. Experience with DDS has been gained in the project IMPERA where OpenSpliceDDS has been used as actual implementation for DDS.

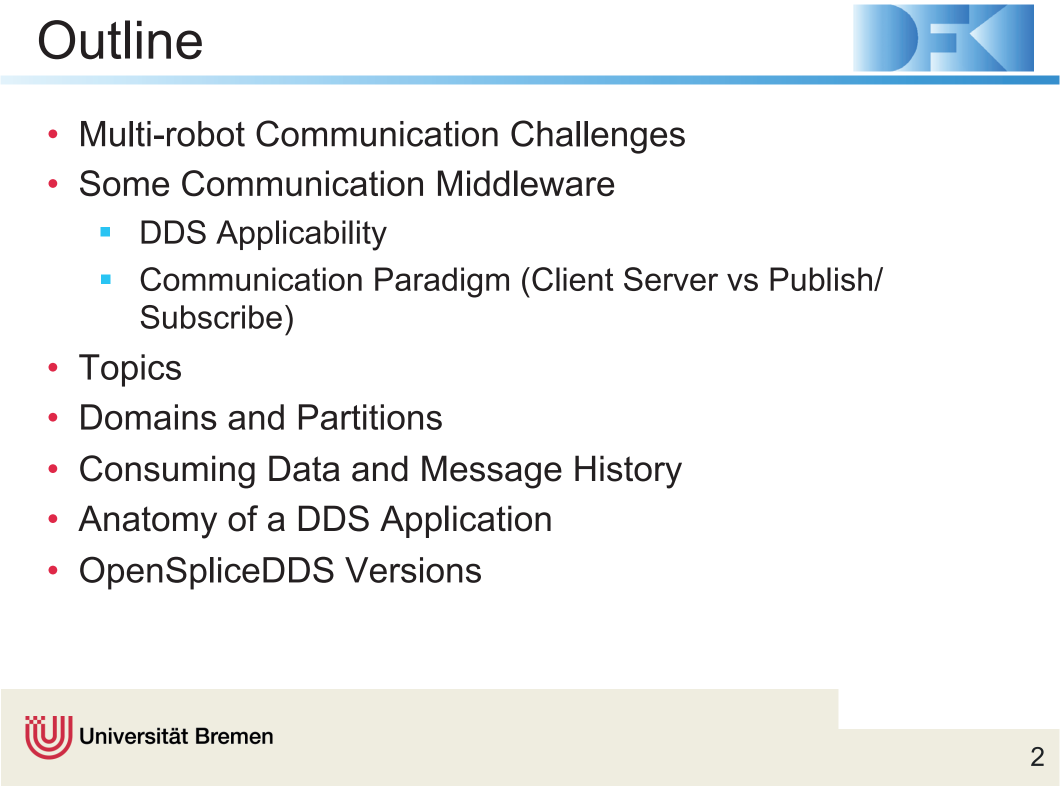


Data Distribution Service – (DDS)

Dr. Ronny Hartanto
Project Day, 19.06.2014

DFKI - Labor Bremen & Universität Bremen
Forschungsgruppe Robotik
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

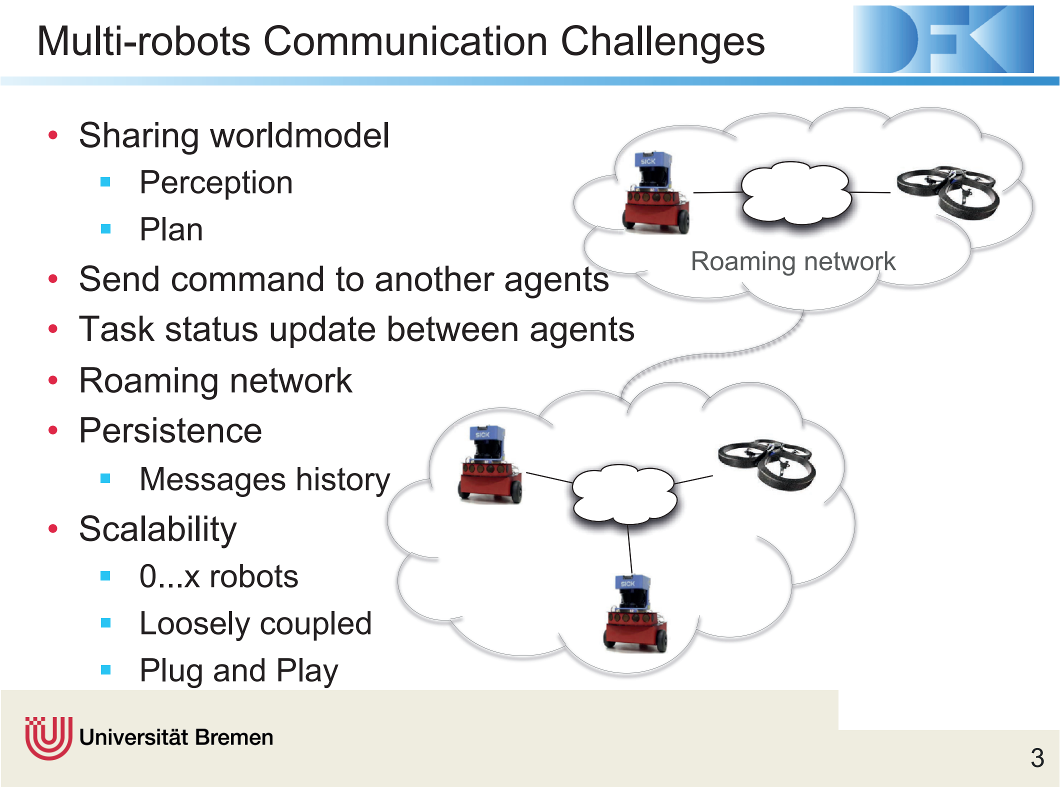
Universität Bremen



Outline

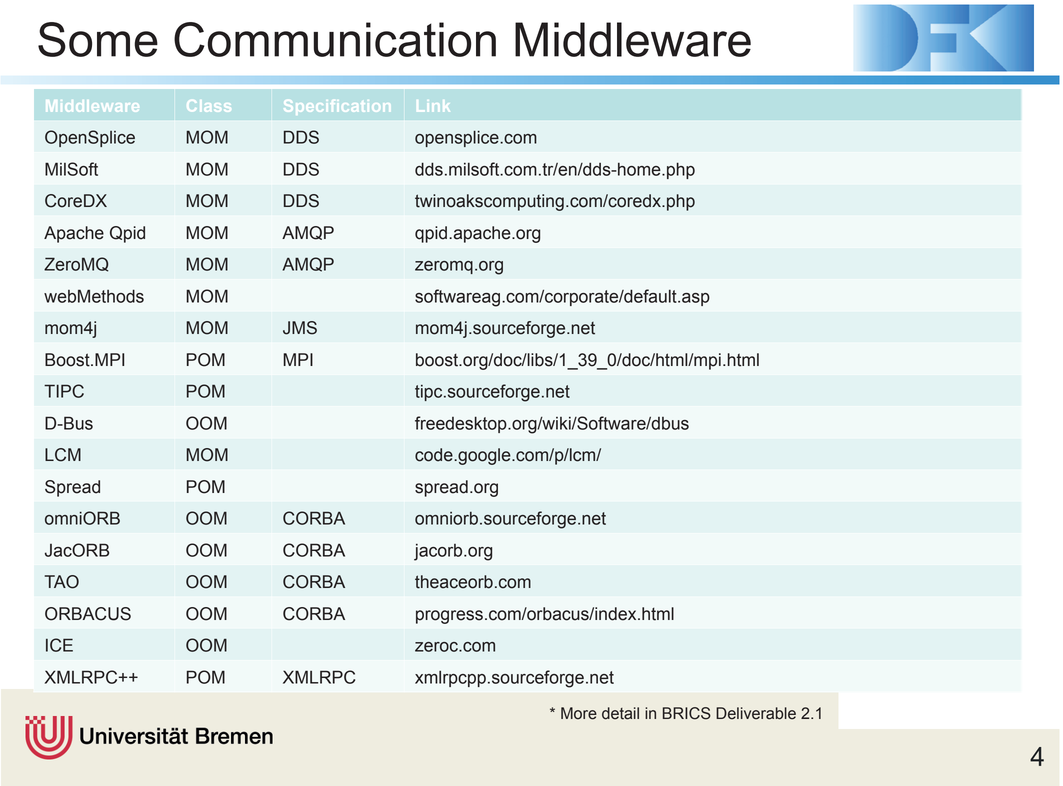
- Multi-robot Communication Challenges
- Some Communication Middleware
  - DDS Applicability
  - Communication Paradigm (Client Server vs Publish/ Subscribe)
- Topics
- Domains and Partitions
- Consuming Data and Message History
- Anatomy of a DDS Application
- OpenSpliceDDS Versions

Universität Bremen



Multi-robots Communication Challenges

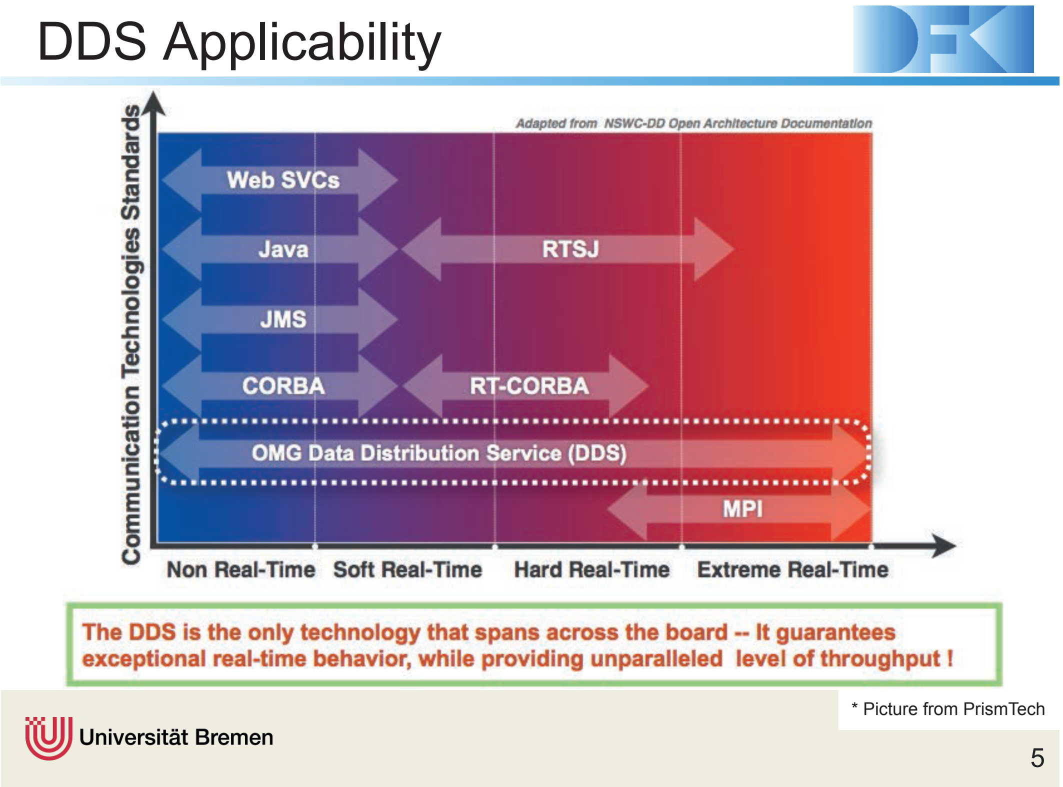
- Sharing worldmodel
  - Perception
  - Plan
- Send command to another agents
- Task status update between agents
- Roaming network
- Persistence
  - Messages history
- Scalability
  - 0...x robots
  - Loosely coupled
  - Plug and Play



Some Communication Middleware

Middleware	Class	Specification	Link
OpenSplice	MOM	DDS	opensplice.com
MilSoft	MOM	DDS	dds.milsoft.com.tr/en/dds-home.php
CoreDX	MOM	DDS	twinoakscomputing.com/coredx.php
Apache Qpid	MOM	AMQP	qpid.apache.org
ZeroMQ	MOM	AMQP	zeromq.org
webMethods	MOM		softwareag.com/corporate/default.asp
mom4j	MOM	JMS	mom4j.sourceforge.net
Boost.MPI	POM	MPI	boost.org/doc/libs/1_39_0/doc/html/mpi.html
TIPC	POM		tipc.sourceforge.net
D-Bus	OOM		freedesktop.org/wiki/Software/dbus
LCM	MOM		code.google.com/p/lcm/
Spread	POM		spread.org
omniORB	OOM	CORBA	omniorb.sourceforge.net
JacORB	OOM	CORBA	jacorb.org
TAO	OOM	CORBA	theaceorb.com
ORBACUS	OOM	CORBA	progress.com/orbacus/index.html
ICE	OOM		zeroc.com
XMLRPC++	POM	XMLRPC	xmlrpcpp.sourceforge.net

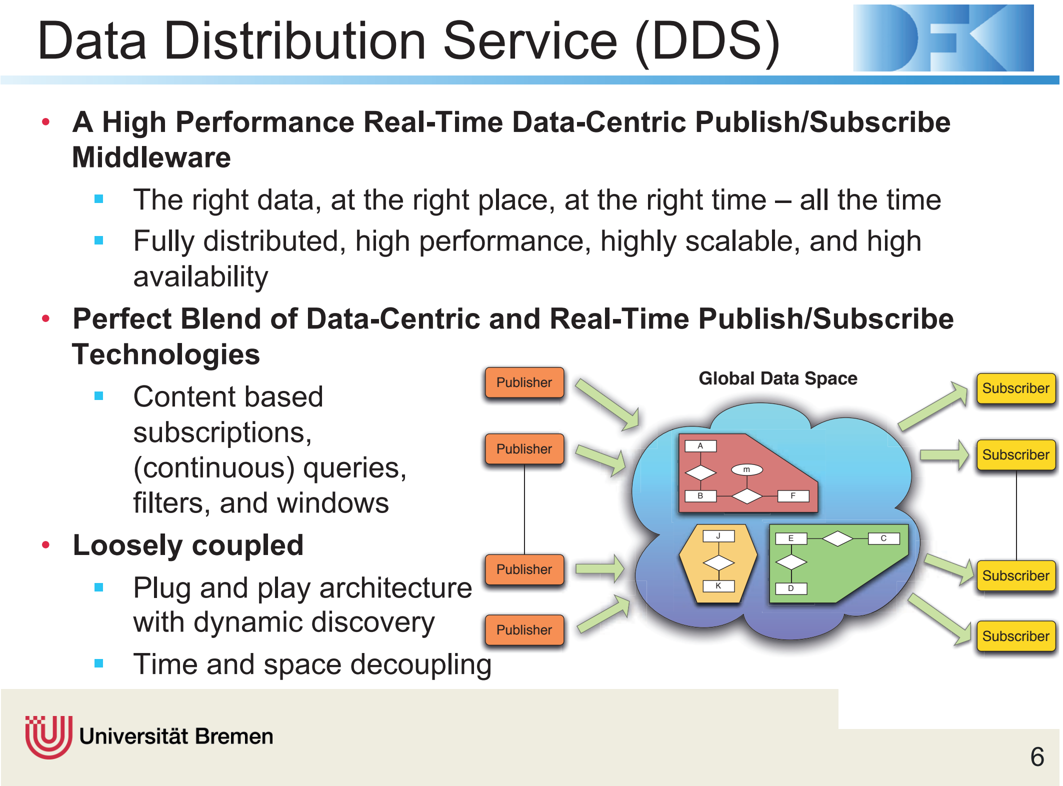
* More detail in BRICS Deliverable 2.1



DDS Applicability

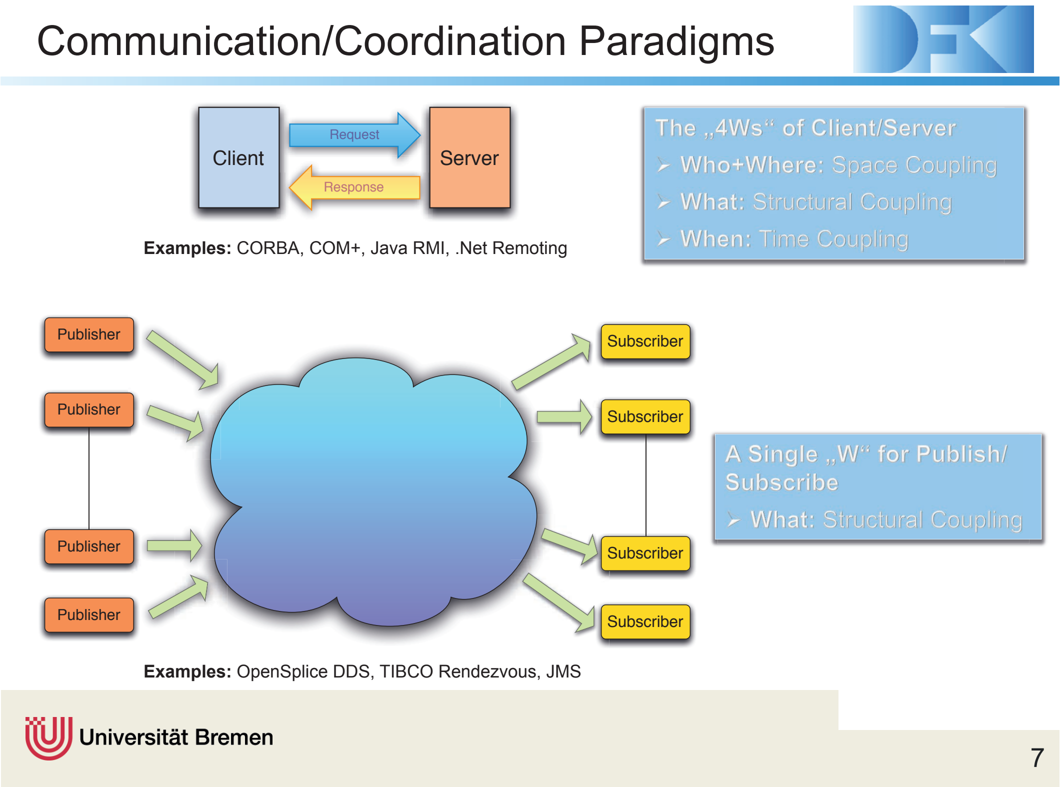
The DDS is the only technology that spans across the board -- It guarantees exceptional real-time behavior, while providing unparalleled level of throughput !

* Picture from PrismTech



Data Distribution Service (DDS)

- **A High Performance Real-Time Data-Centric Publish/Subscribe Middleware**
  - The right data, at the right place, at the right time – all the time
  - Fully distributed, high performance, highly scalable, and high availability
- **Perfect Blend of Data-Centric and Real-Time Publish/Subscribe Technologies**
  - Content based subscriptions, (continuous) queries, filters, and windows
- **Loosely coupled**
  - Plug and play architecture with dynamic discovery
  - Time and space decoupling



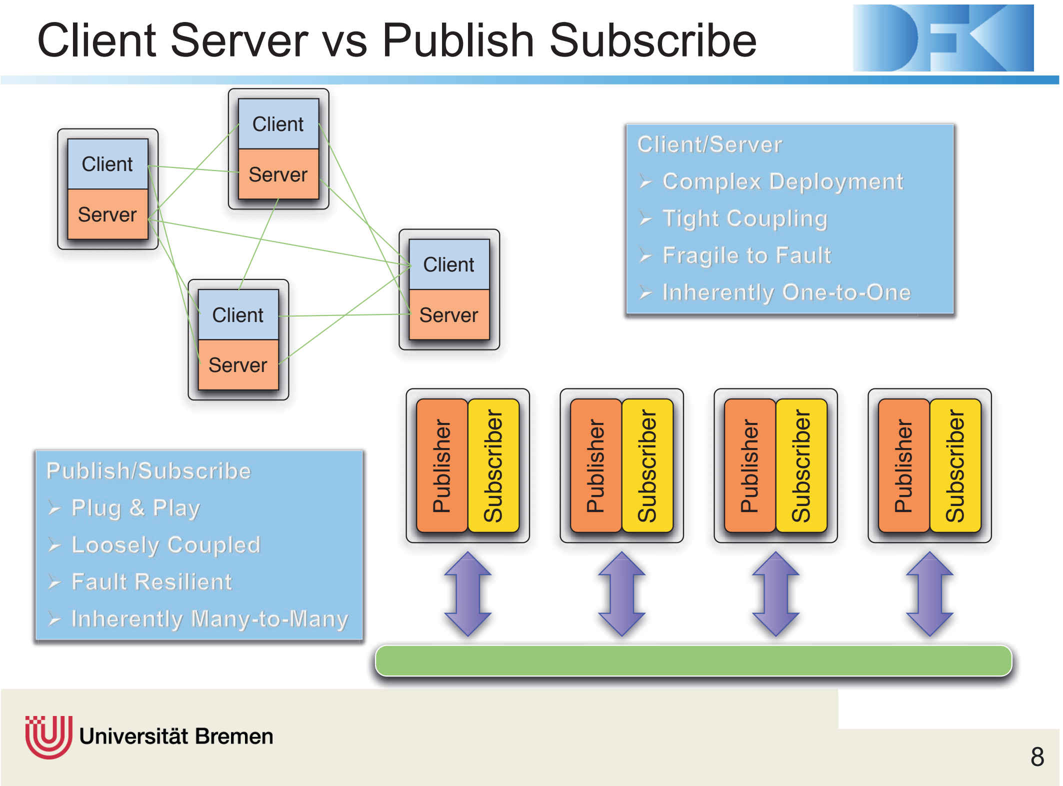
Communication/Coordination Paradigms

Examples: CORBA, COM+, Java RMI, .Net Remoting

The „4Ws“ of Client/Server
- Who+Where: Space Coupling
- What: Structural Coupling
- When: Time Coupling

A Single „W“ for Publish/ Subscribe
- What: Structural Coupling

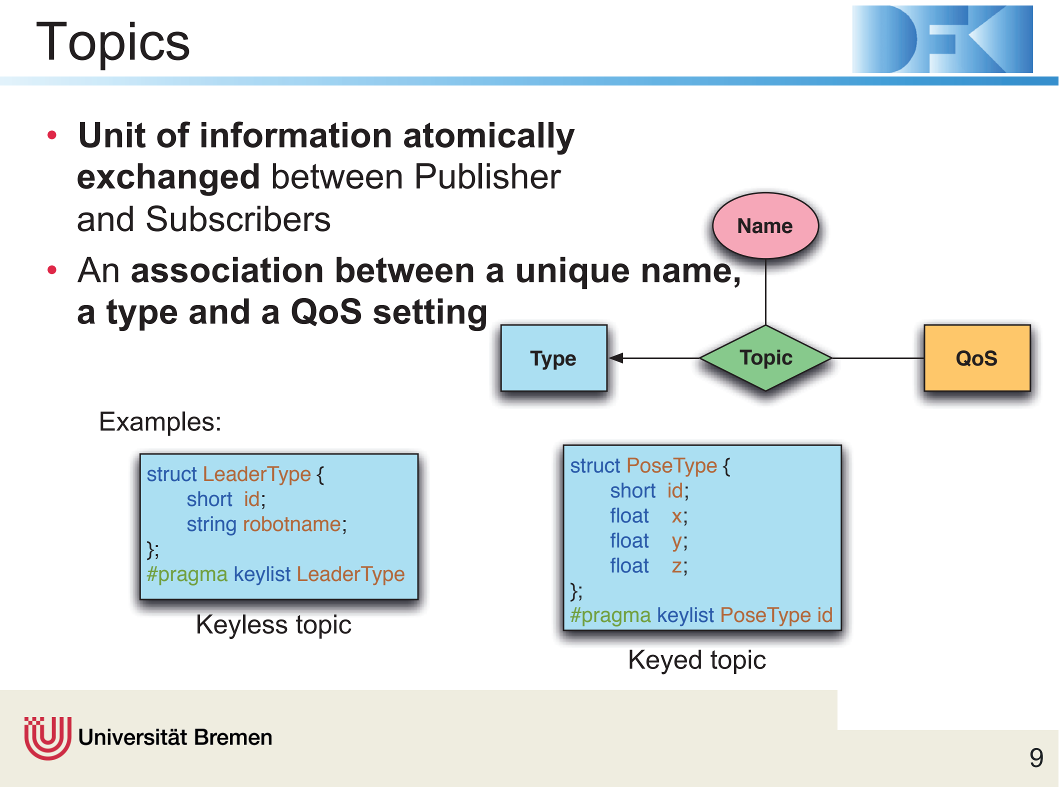
Examples: OpenSplice DDS, TIBCO Rendezvous, JMS



Client Server vs Publish Subscribe

Client/Server
- Complex Deployment
- Tight Coupling
- Fragile to Fault
- Inherently One-to-One

Publish/Subscribe
- Plug & Play
- Loosely Coupled
- Fault Resilient
- Inherently Many-to-Many



Topics

- **Unit of information atomically exchanged** between Publisher and Subscribers
- An **association between a unique name, a type and a QoS setting**

Name
Type
Topic
QoS

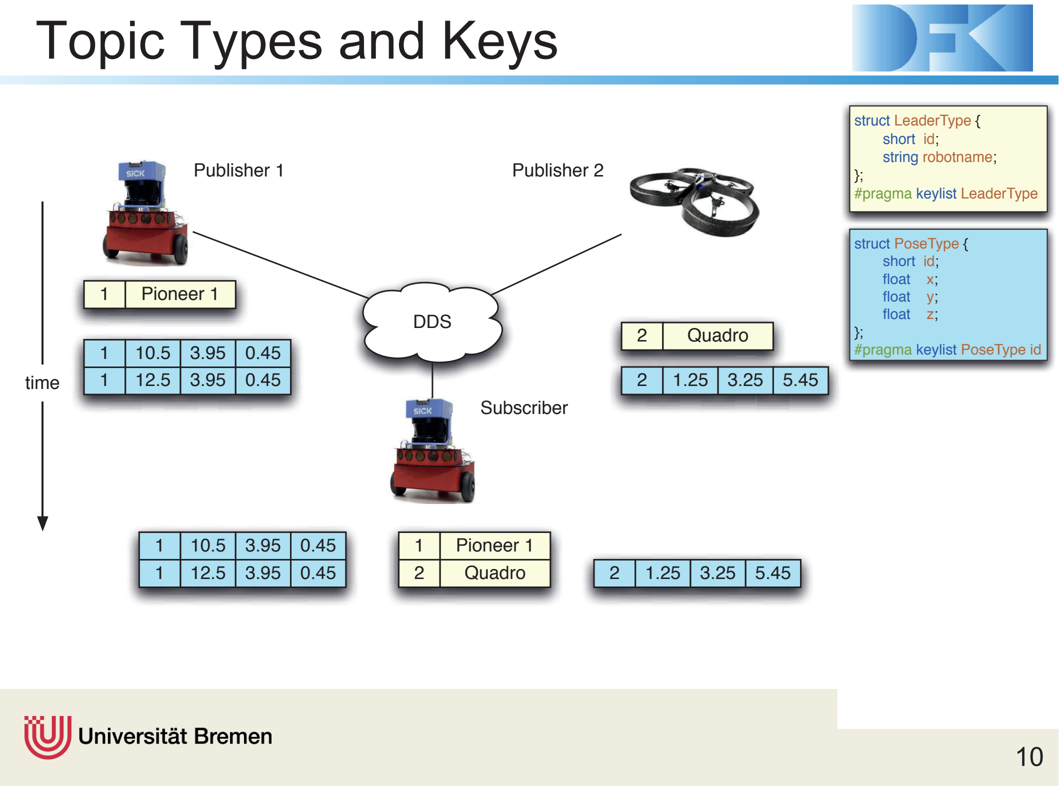
Examples:

```
struct LeaderType {
      short  id;
      string robotname;
};
#pragma keylist LeaderType
```

Keyless topic

```
struct PoseType {
      short  id;
      float   x;
      float   y;
      float   z;
};
#pragma keylist PoseType id
```

Keyed topic



Topic Types and Keys

Publisher 1

Publisher 2

DDS

Subscriber

time

1	Pioneer 1

1	10.5	3.95	0.45
1	12.5	3.95	0.45

2	Quadro

2	1.25	3.25	5.45

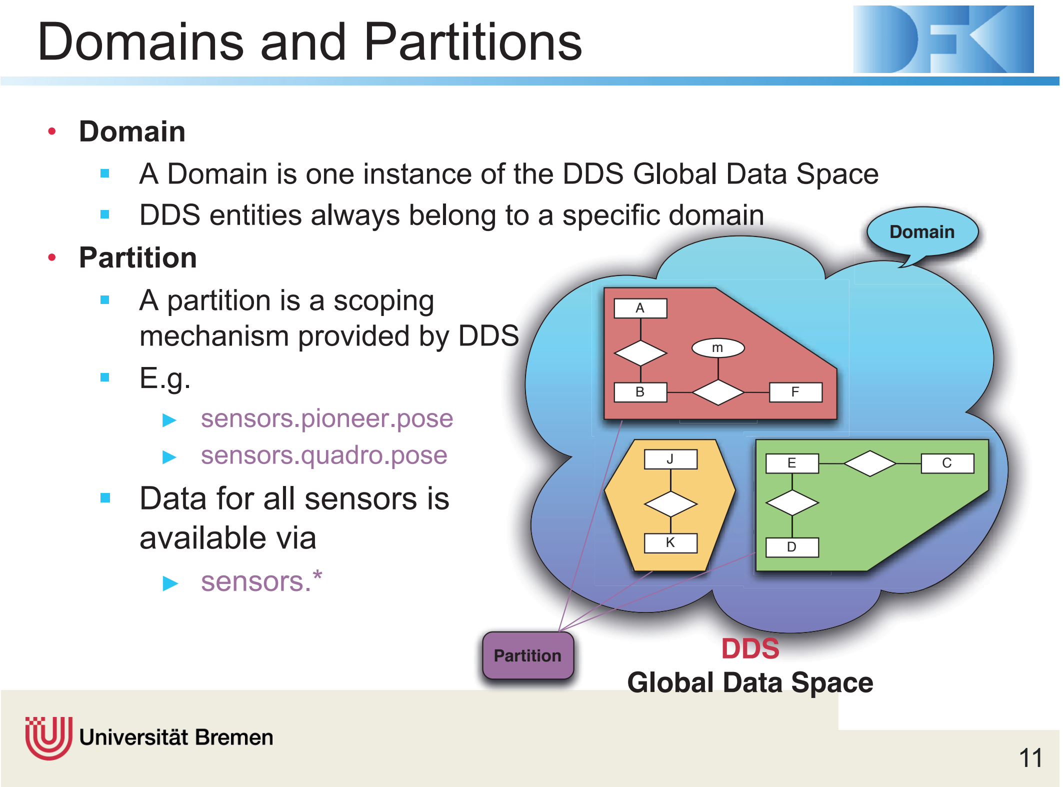
```
struct LeaderType {
      short  id;
      string robotname;
};
#pragma keylist LeaderType
```

```
struct PoseType {
      short  id;
      float   x;
      float   y;
      float   z;
};
#pragma keylist PoseType id
```

1	10.5	3.95	0.45
1	12.5	3.95	0.45

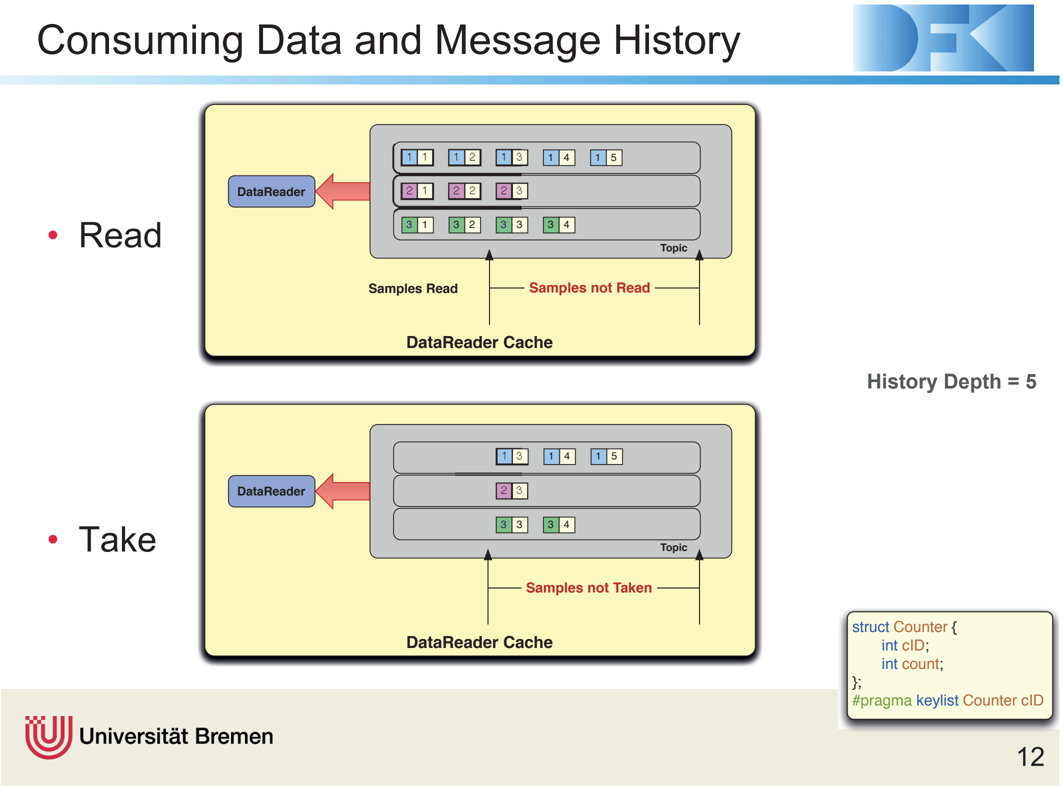
1	Pioneer 1
2	Quadro

2	1.25	3.25	5.45



Domains and Partitions

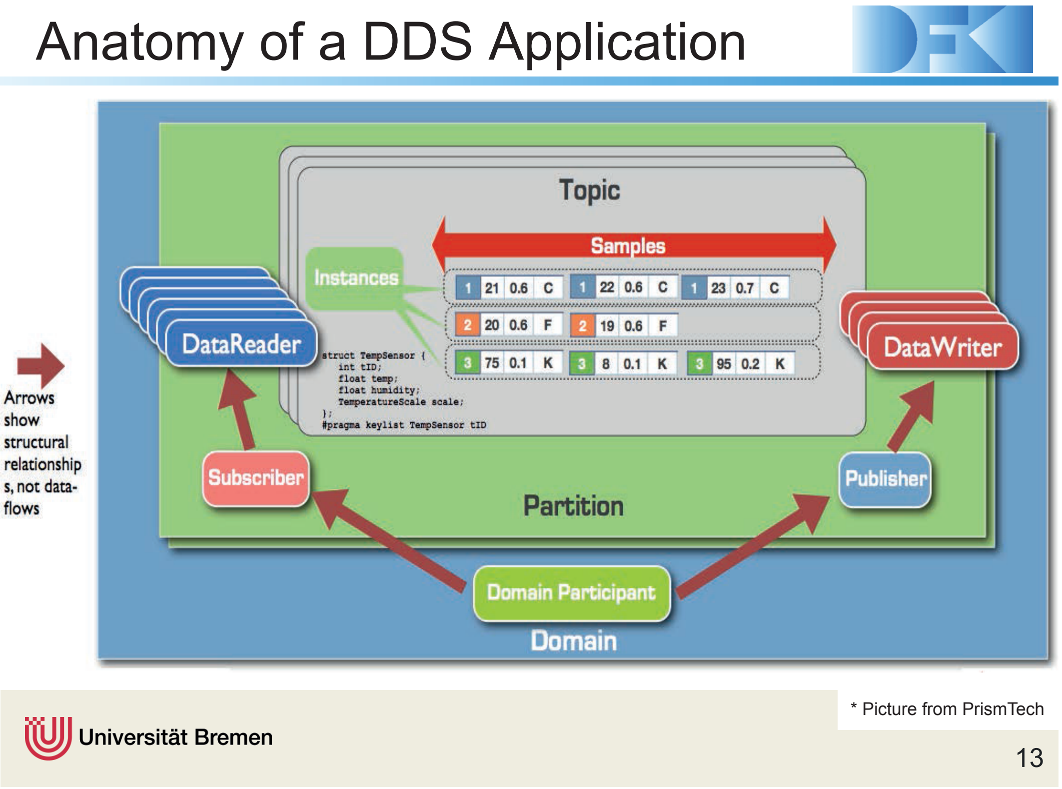
- **Domain**
  - A Domain is one instance of the DDS Global Data Space
  - DDS entities always belong to a specific domain
- **Partition**
  - A partition is a scoping mechanism provided by DDS
  - E.g.
    - sensors.pioneer.pose
    - sensors.quadro.pose
  - Data for all sensors is available via
    - sensors.*



Consuming Data and Message History

- Read
- Take

```
struct Counter {
      int cID;
      int count;
};
#pragma keylist Counter cID
```



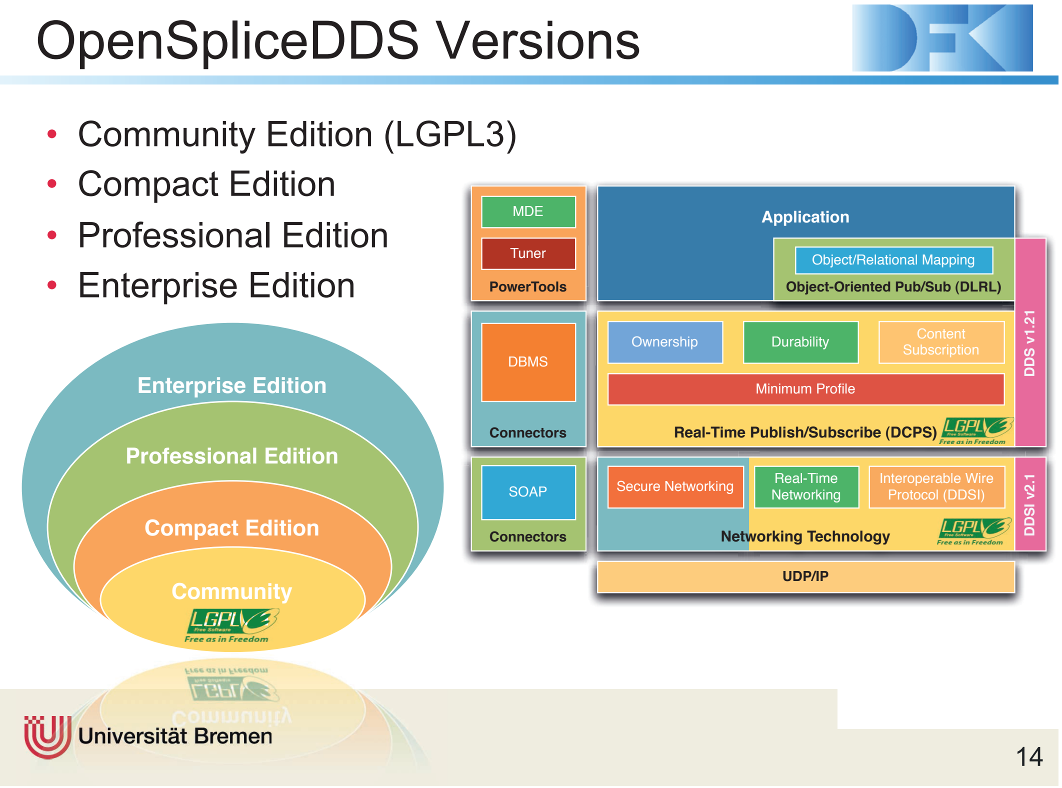
Anatomy of a DDS Application

```
struct TempSensor {
   int tID;
   float temp;
   float humidity;
   TemperatureScale scale;
};
#pragma keylist TempSensor tID
```

Arrows show structural relationships, not data-flows

* Picture from PrismTech

Universität Bremen



OpenSpliceDDS Versions

- Community Edition (LGPL3)
- Compact Edition
- Professional Edition
- Enterprise Edition

Universität Bremen

Resources

- The DDS Tutorial Part I & II (Angelo Corsaro, Ph.D., PrismTech)
- DDS: A Next-Generation Approach to Building Distributed Real-Time Systems (Gerardo Pardo-Castellote, Ph.D., Real-Time Innovations)
- Addressing the Data-Distribution Challenges of Next-Generation Business- and Mission-Critical Systems (Angelo Corsaro, Ph.D., PrismTech)
- BRICS: Best Practice in Robotics Deliverable D2.1 (BRSU)

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

2.6 ‘Planning with reconfigurable multi-robot systems’ (FS-P-01)

Thomas M. Roehr[1], *Ronny Hartanto*[1]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `thomas.roehr@dfki.de`, `ronny.hartanto@dfki.de`

Abstract

A number of space missions have proven the effectiveness of applying robots for planetary exploration. Those missions are usually handicapped by long distances and limited resources on the communication network which making such operations less efficient. One possible approach to overcome this problem is by increasing the level of autonomy of the deployed robotic systems. However, this is only cautiously being accepted as a tool for existing space missions and thus applied in a very limited fashion. Some reasons for these are limited experience with this technology in space missions, development costs, and a low to no risk tolerance in space missions. Meanwhile, the “pressure to reduce manpower during routine mission operation” is real, though thus such robotic missions are far from becoming routine missions and demanding further research on novel mission design concepts. Therefore, this poster presents an approach for applying reconfigurable multi-robot systems to allow for more and safer autonomy in upcoming missions.

German
Research Center
for Artificial
Intelligence

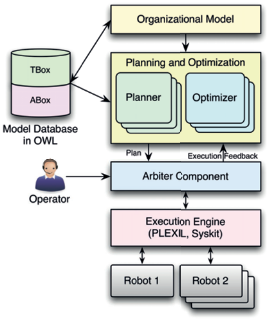
Planning with reconfigurable multi-robot systems

Thomas M. Roehr and Ronny Hartanto

A multi-robot architecture based on ESA’s Functional Reference Model

ESA’s **Functional Reference Model (FRM)** serves as main architectural template for the design of our multi-robot system. The FRM suggest the application of mission (C), task (B) and action (A) layer. To achieve full distribution of the (semi-) autonomous multi-robot system we:

- maintain a task layer and an action layer on each robot
- allow for horizontal extension of task and action layer by using a (FIPA-based) communication supporting distributed systems
- allow robot to robot communication



Plan Domain Definition Language

```
;Domain
(define (domain test_domain)
    (:requirements :strips :equality :typing :conditional-effects)
    (:types location physob_id physob_type)
    (:constants sherpa crex payload - physob_type)
    (:predicates ( at ?x - physob_id ?l - location)
        ( is_a ?x - physob_id ?r - physob_type)
        ( connected ?x ?y - physob_id)
       ( cannot_move ?x - physob_id)    )

(:action move
    :parameters (?obj - physob_id ?m ?l - location)
    :precondition ( and (at ?obj ?m) (not (= ?m ?l))
                                  (not (cannot_move ?obj) ))
    :effect (and (at ?obj ?l) (not (at ?obj ?m))
                 (forall (?z) (when
                         (and (connected ?z ?obj)
                         (not (= ?z ?obj)))
                         (and (at ?z ?l) (not (at ?z ?m))))))))
; Problem
(define (problem test_domain-1)
  (:domain test_domain)
  (:objects sherpa_0 crex_0 pl_0 - physob_id
                sherpa - physob_type
                location_s0  location_c0 location_p0 – location)
  (:init  (is_a sherpa_0 sherpa) (is_a crex_0 crex)  (is_a pl_0 payload)
    (at sherpa_0 location_s0) (at crex_0 location_c0) (at pl_0 location_p0)
    (can_move sherpa_0) (can_move crex_0))
  (:goal (at sherpa_0 location_c0))
```

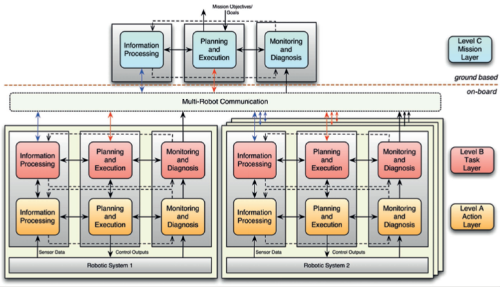
Planning architecture based on an organization model

Planning architecture based on organization modeling

To exploit the capabilities of a reconfigurable multi-robot system we suggest a planning architecture that consists of:

- an **organization model** that holds domain knowledge and allows inference of capabilities of recombined systems
- a component for **planning and optimization** which generates plan candidate and which is operates upon domain knowledge from the organization model and dynamic information about the system
- an **arbiter** component that evaluates the plan candidates based on predefined policies set by an operator
- an **execution engine** which manages the activities of a single robot and synchronizes with other robots using direct interaction based on predefined protocols

In order to maximize reuse of existing technologies the organization modeling uses Ontology Web Language (OWL) as its representation. Furthermore, planning domain and problem are generated in PDDL to interface with existing PDDL planner such LAMA.



Extended Functional Reference Model for multi-robot systems

Organization modeling

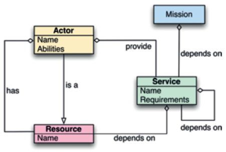
The capability of reconfiguration of a multi-robot system introduces new challenges for planning:

- considering permutations, i.e. new robotic actors resulting from combining two or more other actors
- handling effects of reconfiguration, e.g. operative and dormant actors

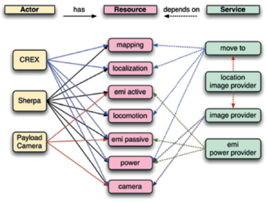
It also opens up new opportunities:

- increasing efficiency and robustness of the multi-robot system
- perform actor and resource management to optimize safety constraints

Modeling of the set of atomic actors and their respective capabilities allows to inference of resulting capabilities after reconfiguration. Furthermore, it provides the basis for performing analysis of organization properties, e.g., looking at resource redundancy and simulating the effects of the loss of resources.



Organization meta modeling



Modeling of resource dependencies

Supported by:

Federal Ministry
of Economics
and Technology

on the basis of a decision
by the German Bundestag

DLR

Grant Number 50RA1201

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

2.7 ‘FIPA-based multi-robot infrastructure for Rock’ (FS-P-02)

Satia Herfert(1), *Thomas M. Roehr*(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `satia.herfert@dfki.de`, `thomas.roehr@dfki.de`

Abstract

The application of teams of autonomous, cooperating robots in areas which are hazardous and inaccessible for humans is highly desirable. Furthermore, a decentralized approach allows for more fail-prune algorithms, but has its own challenges. To improve the capability of such multi-robot systems, we built an infrastructure that allows to share resources in a distributed system and to easily find and contact agents, even from other ecosystems. This poster provides a motivation for this research and illustrates parts of the architecture.

German Research Center for Artificial Intelligence

FIPA-based multi-agent infrastructure for Rock

Distributed locking algorithms and platform-interoperability in TransTerrA

Satia Herfert and Thomas M. Roehr

Multi-agent infrastructure

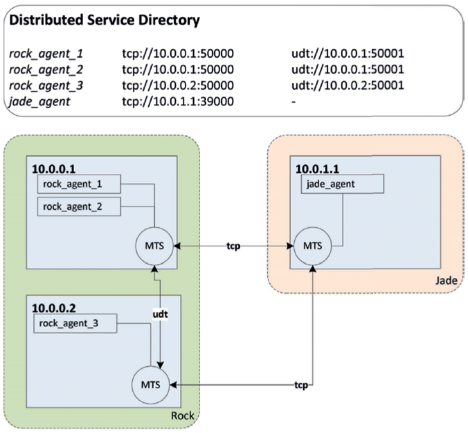
The application of teams of autonomous, cooperating robots in areas which are hazardous and inaccessible for humans is highly desirable. Furthermore, a decentralized approach allows for more fail-prune algorithms, but has its own challenges. To improve the capability of such multi-robot systems, we built an infrastructure that allows

- to share resources in a distributed system and
- to easily find and contact agents, even from other ecosystems.

Distributed locking for shared resources

Going for a decentralized solution, robots need to perform their resource sharing with distributed algorithms, i.e. Sherpa and Coyote will have to communicate about camera and battery usage. The challenge is that each software resource corresponds to a physical device belonging to exactly one agent – existing algorithms do not model this.

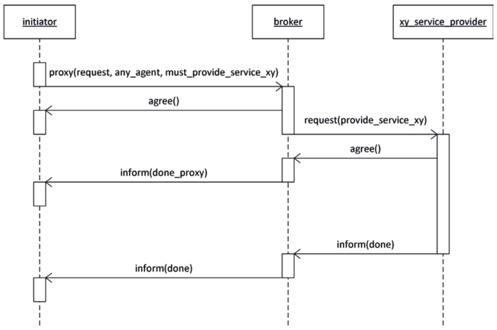
Good candidates for handling this problem are Ricart-Agrawala and Suzuki-Kasami. The first one uses a broadcast mechanism whereas the second one relies on a software-token granting exclusive access. Both have been implemented and extended. To guarantee that agents can continue their work, even if one agent fails for some reason, they have been equipped with agent-failure-detection and handling. This detection is realized sending heartbeat messages to resource holders in intervals. If no instant response is received, the agent is assumed to have failed/crashed.



Distributed Service Directory

rock_agent_1	tcp://10.0.0.1:50000	udt://10.0.0.1:50001
rock_agent_2	tcp://10.0.0.1:50000	udt://10.0.0.1:50001
rock_agent_3	tcp://10.0.0.2:50000	udt://10.0.0.2:50001
jade_agent	tcp://10.0.1.1:39000	-

Structure of the FIPA services. The message transport service (MTS) forwards messages to other MTS. They look up the addresses of agents in both ecosystems in the distributed service directory.



Typical message flow for the FIPA-brokering protocol. The initiator asks the broker to find an agent for him that provides a certain service. The broker finds one, and informs the initiator about the progress.

As an example Coyote could request exclusive access to a payload item camera, that Sherpa carries. If the communication is cut, which is detected, Coyote will consequently have to modify its behavior.

FIPA and JADE-interoperability

The before mentioned algorithms exchange messages that are compliant to the FIPA standards for inter-operation of heterogeneous agents. FIPA's reference implementation, JADE, is written in Java. Rock was extended to be able to connect to JADE systems. Being able to cross-platform communicate with JADE enables developers to take advantage of the broad JADE infrastructure. JADE has a built-in GUI that facilitates testing and external developers wishing to use Rock's functionality are able to remain in JADE's ecosystem.

The agents are able to easily find each other with a distributed service directory based on zeroconf/avahi network entries. This is platform independent. They continue to communicate via a set of reusable TCP/UDP sockets, over which they send XML encoded data.

The given infrastructure provides a basis for implementing advanced cooperation strategies such as auction-based algorithms that solve task assignment problems. In the long run, this enables us to use reconfigurable multi-robot teams to a maximum extent.

Supported by:

Federal Ministry
of Economics
and Technology

on the basis of a decision
by the German Bundestag

Grant Number 50RA1201

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

3 ‘Manipulation & Control’

3.1 ‘Work Group Activities’ (MC-T-01)

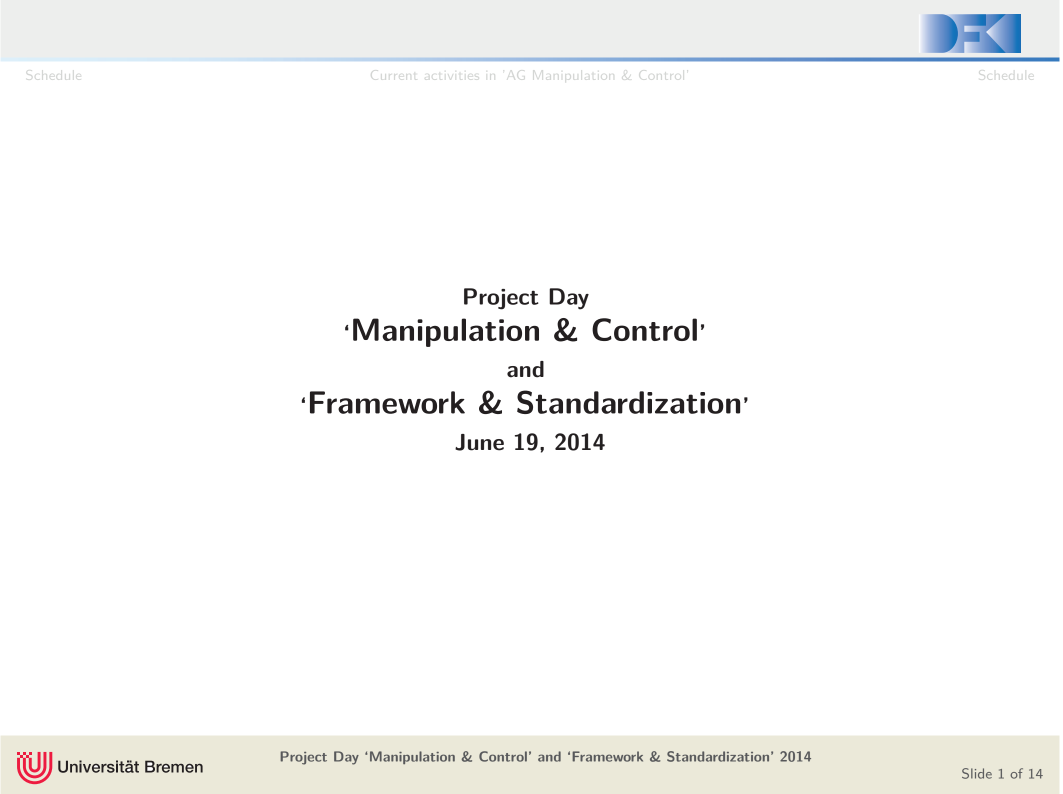
Bertold Bongardt⁽¹⁾*, Benjamin Girault*⁽¹⁾*, et al.*

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

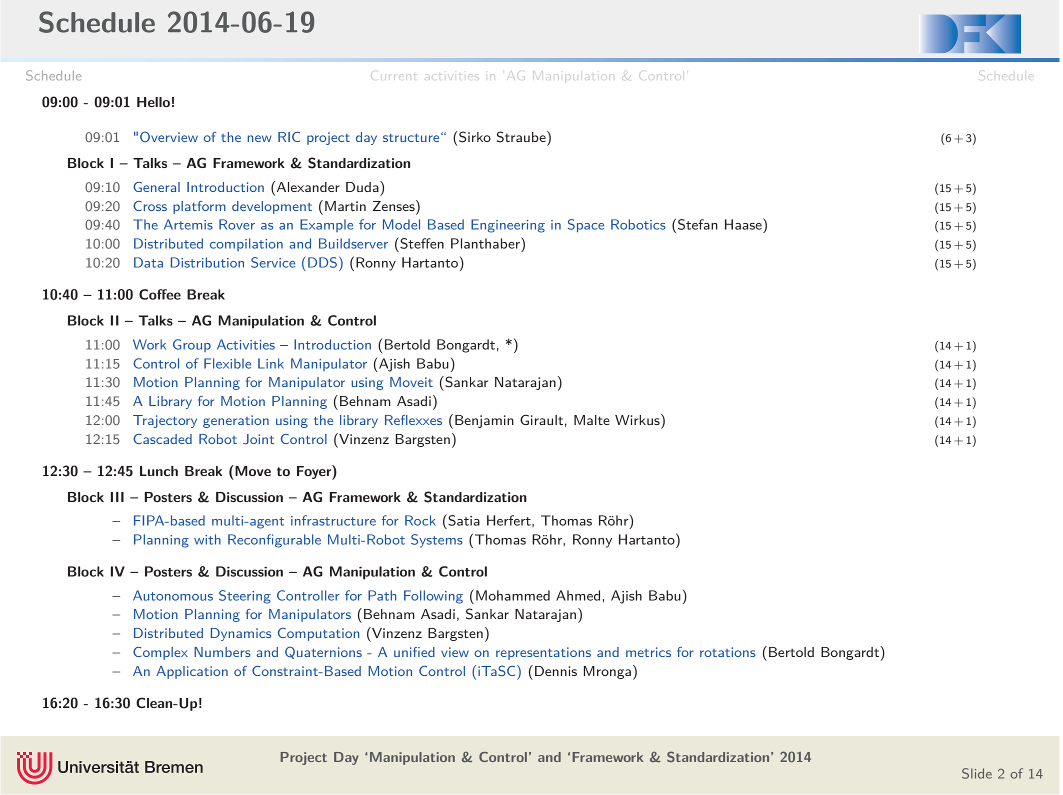
Contact: `bertold.bongardt@dfki.de`, `benjamin.girault@dfki.de`

Abstract

In this introduction, a compact outline of the actitivities of the work group during the last year is given. Further, the posters of the afternoon session are introduced briefly.



Project Day
'Manipulation & Control'
and
'Framework & Standardization'
June 19, 2014



Schedule 2014-06-19

09:00 - 09:01 Hello!

09:01 "Overview of the new RIC project day structure" (Sirko Straube) (6 + 3)

Block I – Talks – AG Framework & Standardization

09:10 General Introduction (Alexander Duda) (15 + 5)
09:20 Cross platform development (Martin Zenses) (15 + 5)
09:40 The Artemis Rover as an Example for Model Based Engineering in Space Robotics (Stefan Haase) (15 + 5)
10:00 Distributed compilation and Buildserver (Steffen Planthaber) (15 + 5)
10:20 Data Distribution Service (DDS) (Ronny Hartanto) (15 + 5)

10:40 – 11:00 Coffee Break

Block II – Talks – AG Manipulation & Control

11:00 Work Group Activities – Introduction (Bertold Bongardt, *) (14 + 1)
11:15 Control of Flexible Link Manipulator (Ajish Babu) (14 + 1)
11:30 Motion Planning for Manipulator using Moveit (Sankar Natarajan) (14 + 1)
11:45 A Library for Motion Planning (Behnam Asadi) (14 + 1)
12:00 Trajectory generation using the library Reflexxes (Benjamin Girault, Malte Wirkus) (14 + 1)
12:15 Cascaded Robot Joint Control (Vinzenz Bargsten) (14 + 1)

12:30 – 12:45 Lunch Break (Move to Foyer)

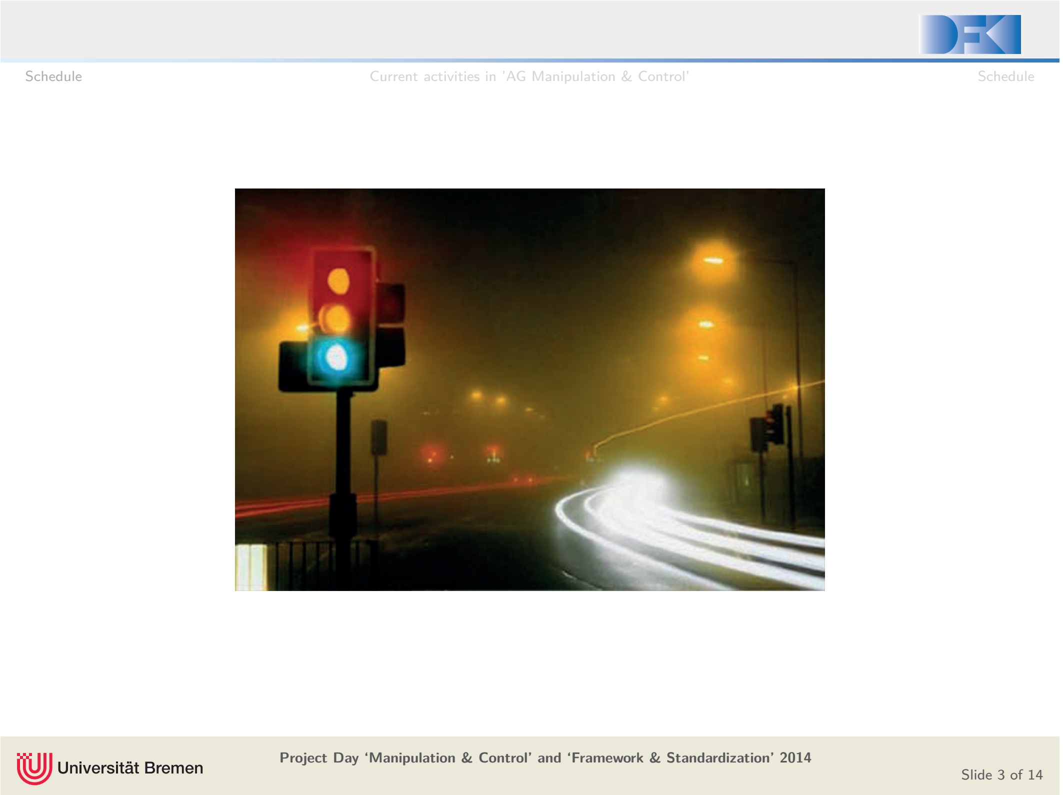
Block III – Posters & Discussion – AG Framework & Standardization

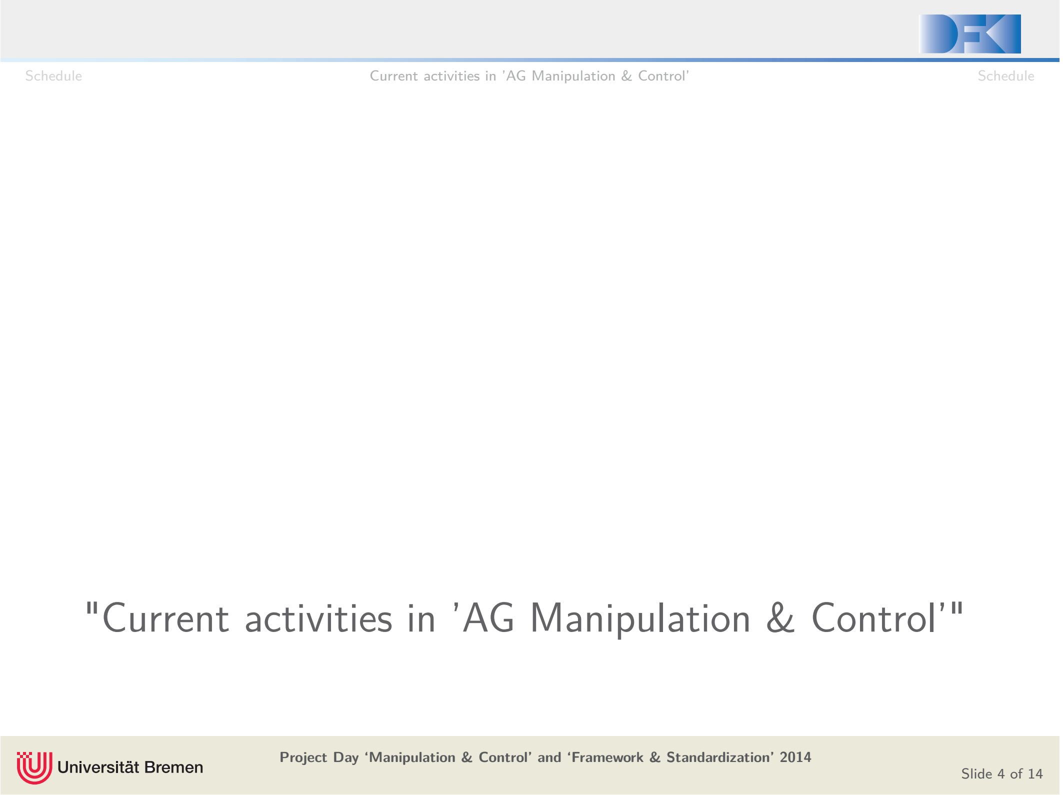
- FIPA-based multi-agent infrastructure for Rock (Satia Herfert, Thomas Röhr)
- Planning with Reconfigurable Multi-Robot Systems (Thomas Röhr, Ronny Hartanto)

Block IV – Posters & Discussion – AG Manipulation & Control

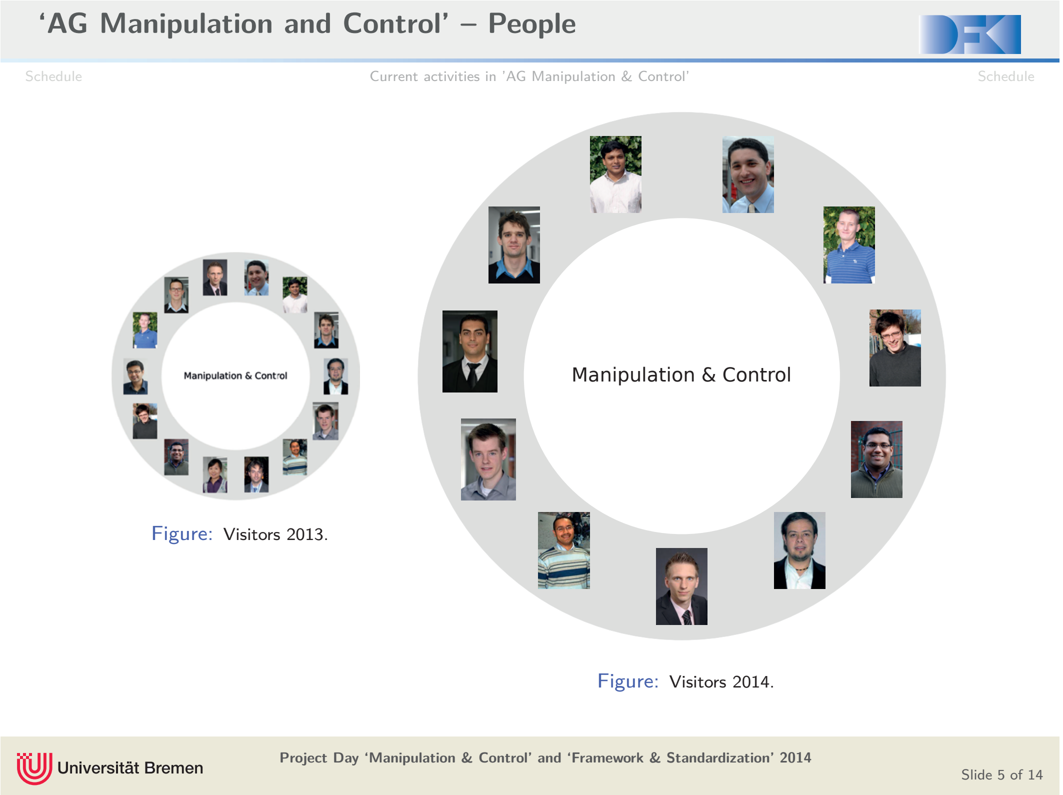
- Autonomous Steering Controller for Path Following (Mohammed Ahmed, Ajish Babu)
- Motion Planning for Manipulators (Behnam Asadi, Sankar Natarajan)
- Distributed Dynamics Computation (Vinzenz Bargsten)
- Complex Numbers and Quaternions - A unified view on representations and metrics for rotations (Bertold Bongardt)
- An Application of Constraint-Based Motion Control (iTaSC) (Dennis Mronga)

16:20 - 16:30 Clean-Up!





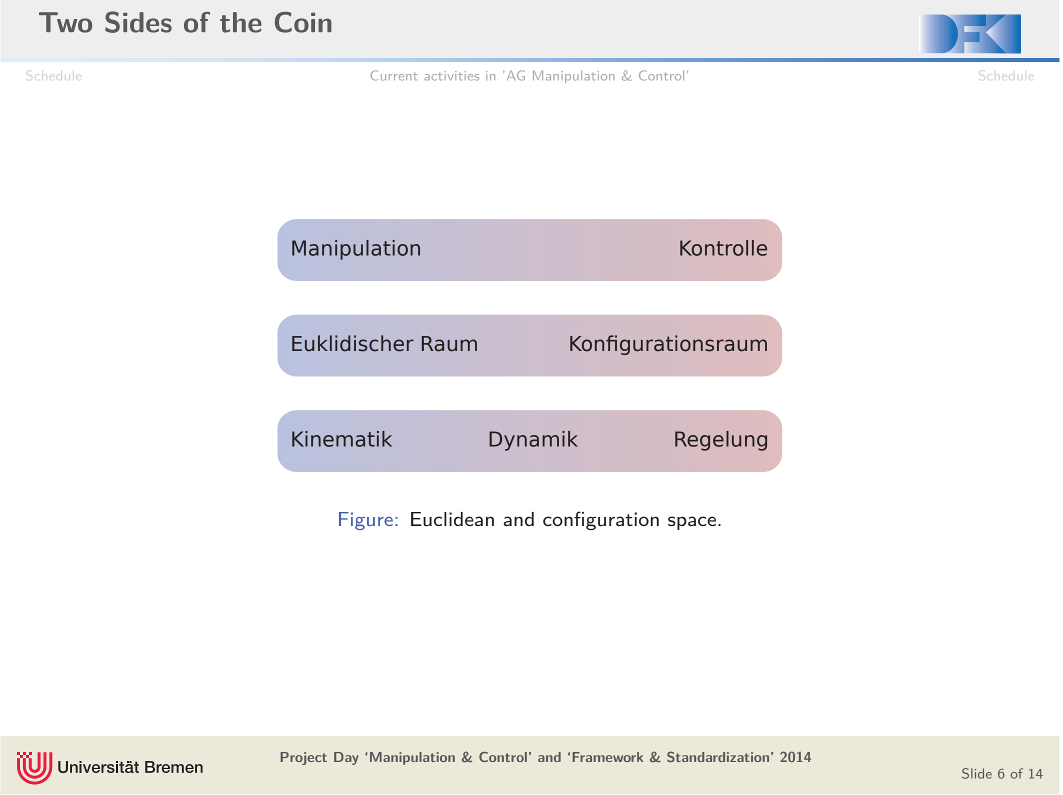
"Current activities in ’AG Manipulation & Control’"



‘AG Manipulation and Control’ – People

Figure: Visitors 2013.

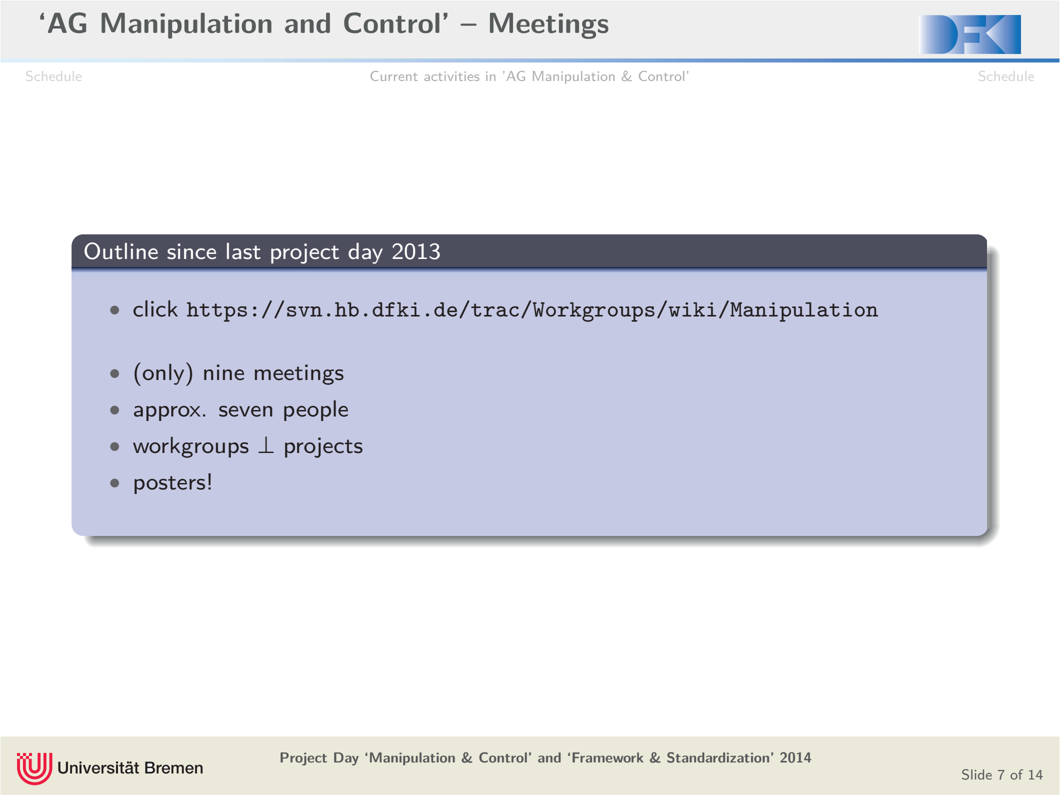
Figure: Visitors 2014.



Two Sides of the Coin

Manipulation		Kontrolle
Euklidischer Raum		Konfigurationsraum
Kinematik	Dynamik	Regelung

Figure: Euclidean and configuration space.

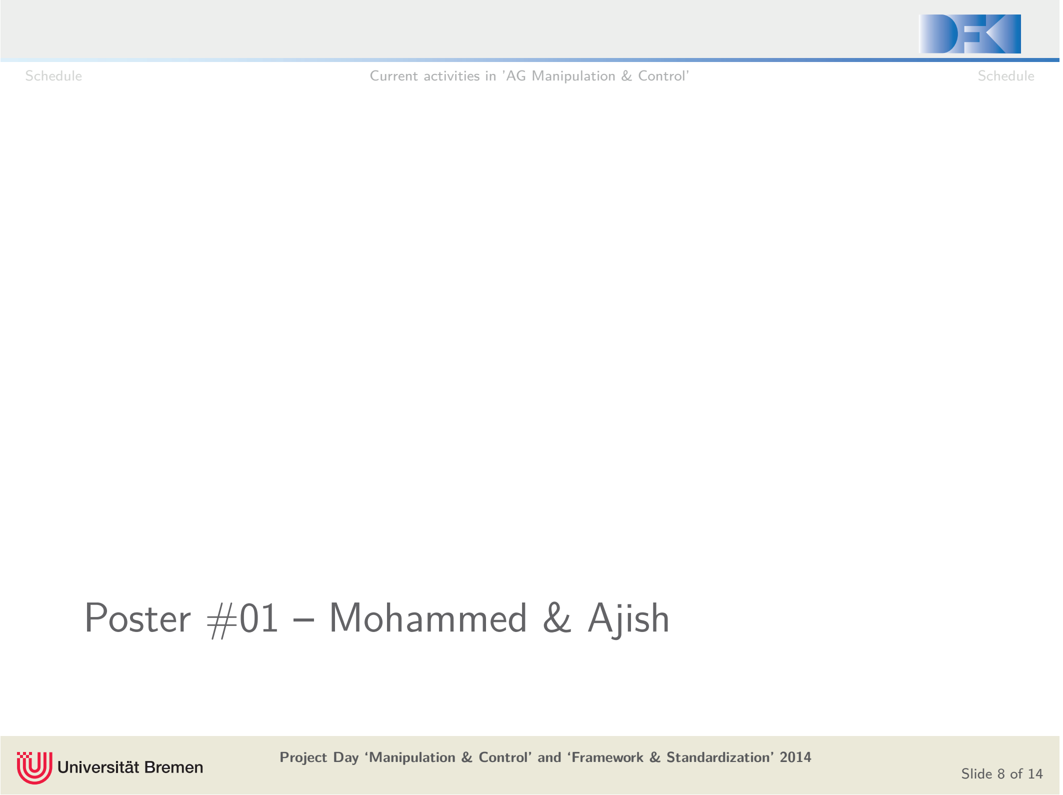


‘AG Manipulation and Control’ – Meetings

Outline since last project day 2013

- click `https://svn.hb.dfki.de/trac/Workgroups/wiki/Manipulation`

- (only) nine meetings
- approx. seven people
- workgroups $\perp$ projects
- posters!



Poster #01 – Mohammed & Ajish

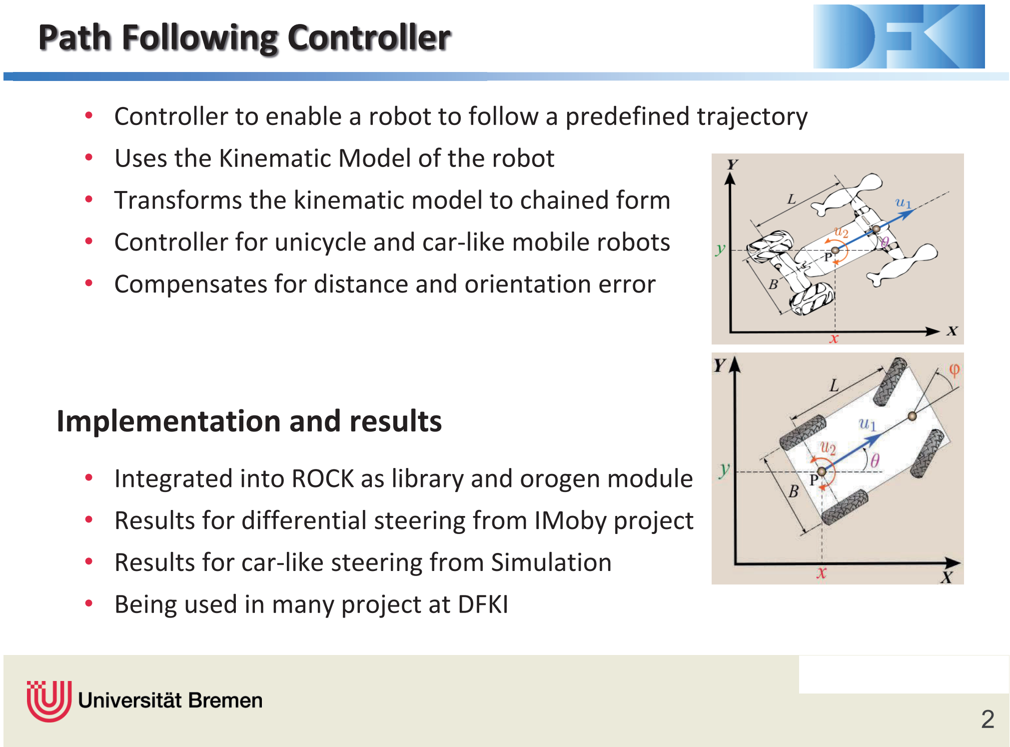
Autonomous Steering Controller for Path Following

Mohammed Ahmed and Ajish Babu

Project-day AG Manipulation / AG Framework & Standardization, 19th June 2014

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



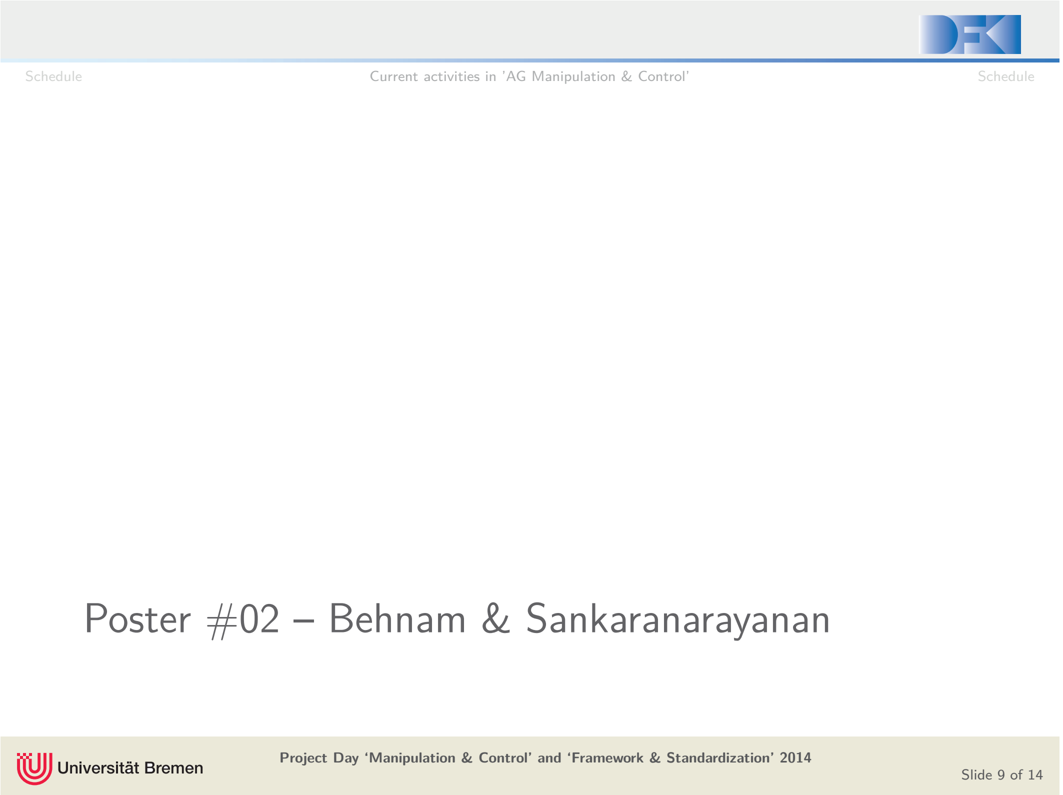
Path Following Controller

- Controller to enable a robot to follow a predefined trajectory
- Uses the Kinematic Model of the robot
- Transforms the kinematic model to chained form
- Controller for unicycle and car-like mobile robots
- Compensates for distance and orientation error

Implementation and results

- Integrated into ROCK as library and orogen module
- Results for differential steering from IMoby project
- Results for car-like steering from Simulation
- Being used in many project at DFKI

Universität Bremen



Schedule Current activities in 'AG Manipulation & Control' Schedule

Poster #02 – Behnam & Sankaranarayanan

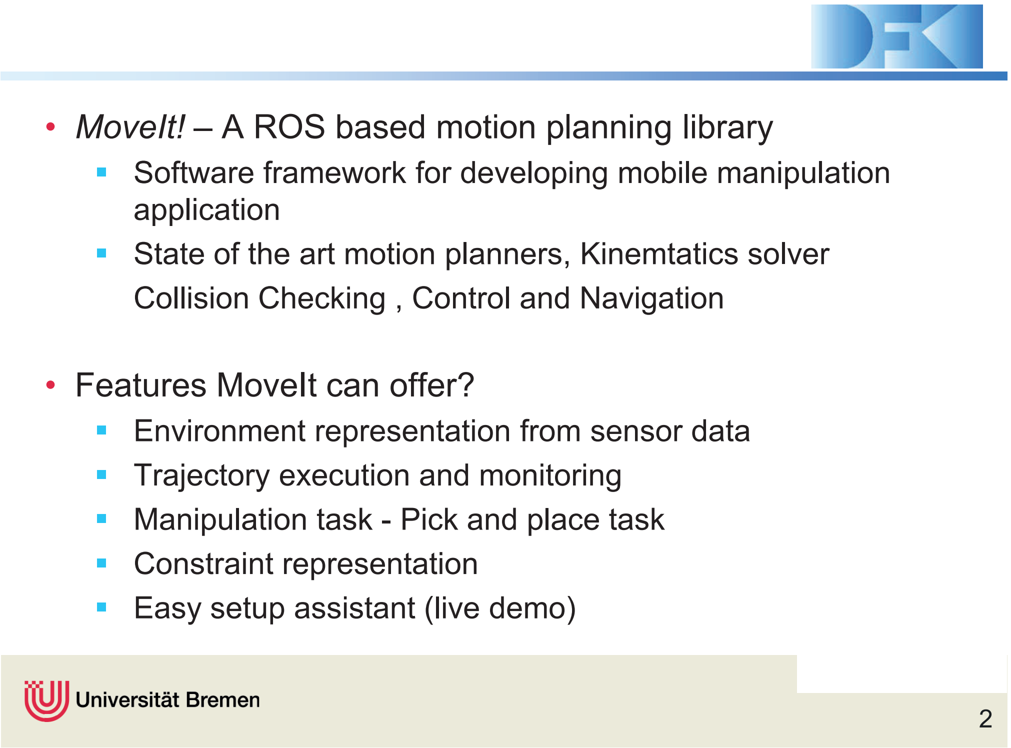
Universität Bremen Project Day 'Manipulation & Control' and 'Framework & Standardization' 2014 Slide 9 of 14

Motion Planning for Manipulator

Behnam Asadi and Sankaranarayanan Natarajan

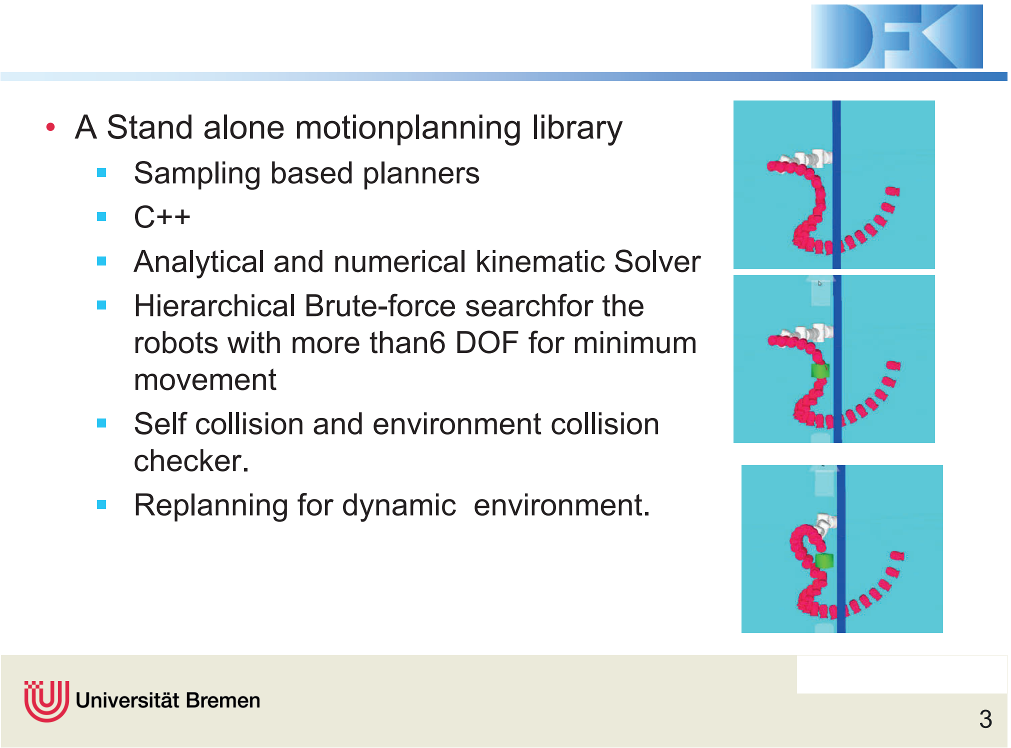
DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



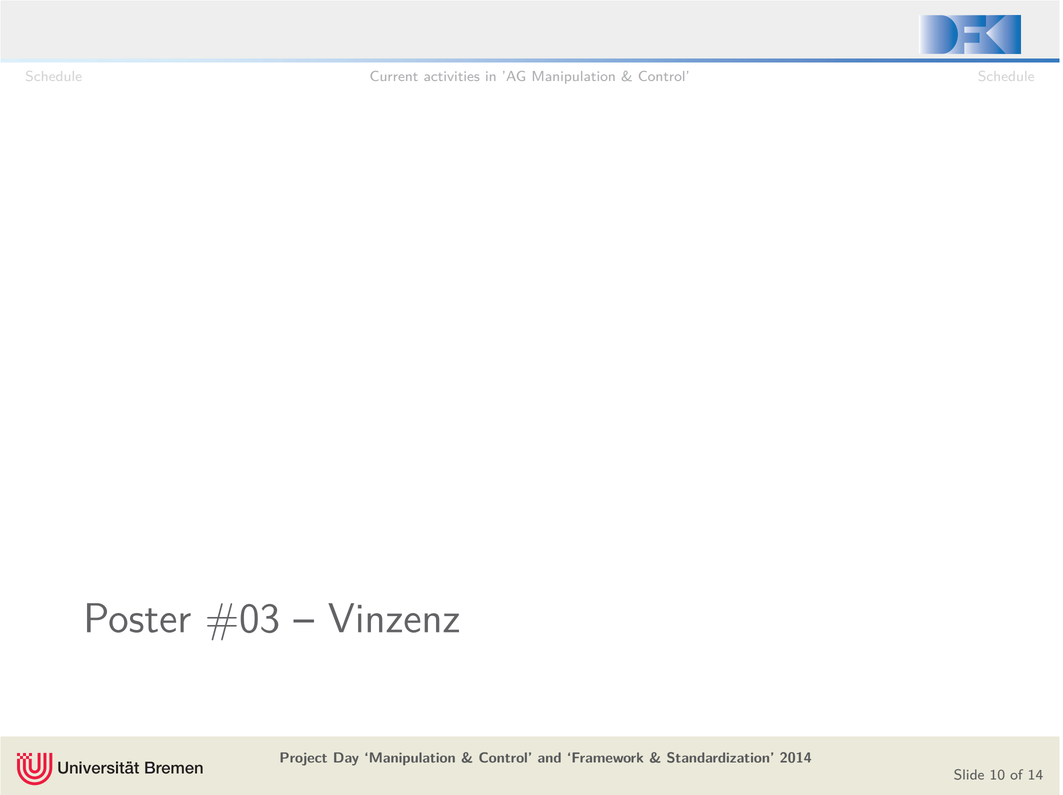
- *MoveIt!* – A ROS based motion planning library
  - Software framework for developing mobile manipulation application
  - State of the art motion planners, Kinemtatics solver Collision Checking , Control and Navigation
- Features MoveIt can offer?
  - Environment representation from sensor data
  - Trajectory execution and monitoring
  - Manipulation task - Pick and place task
  - Constraint representation
  - Easy setup assistant (live demo)

Universität Bremen

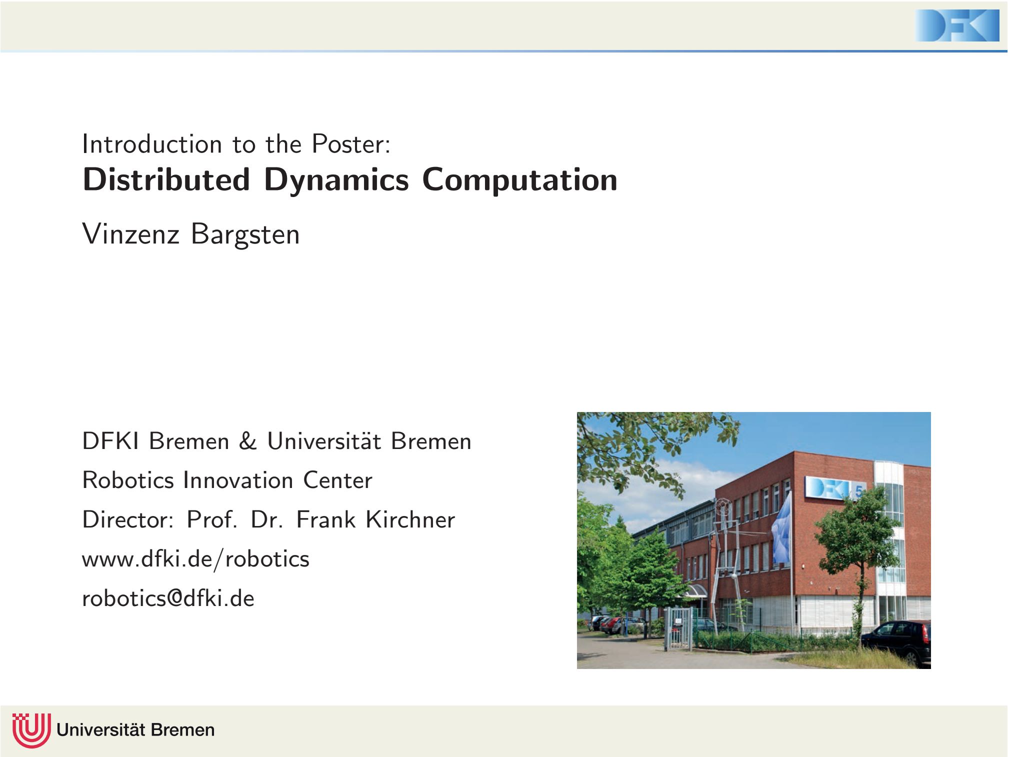


- A Stand alone motionplanning library
  - Sampling based planners
  - C++
  - Analytical and numerical kinematic Solver
  - Hierarchical Brute-force searchfor the robots with more than6 DOF for minimum movement
  - Self collision and environment collision checker.
  - Replanning for dynamic environment.

Universität Bremen



Poster #03 – Vinzenz



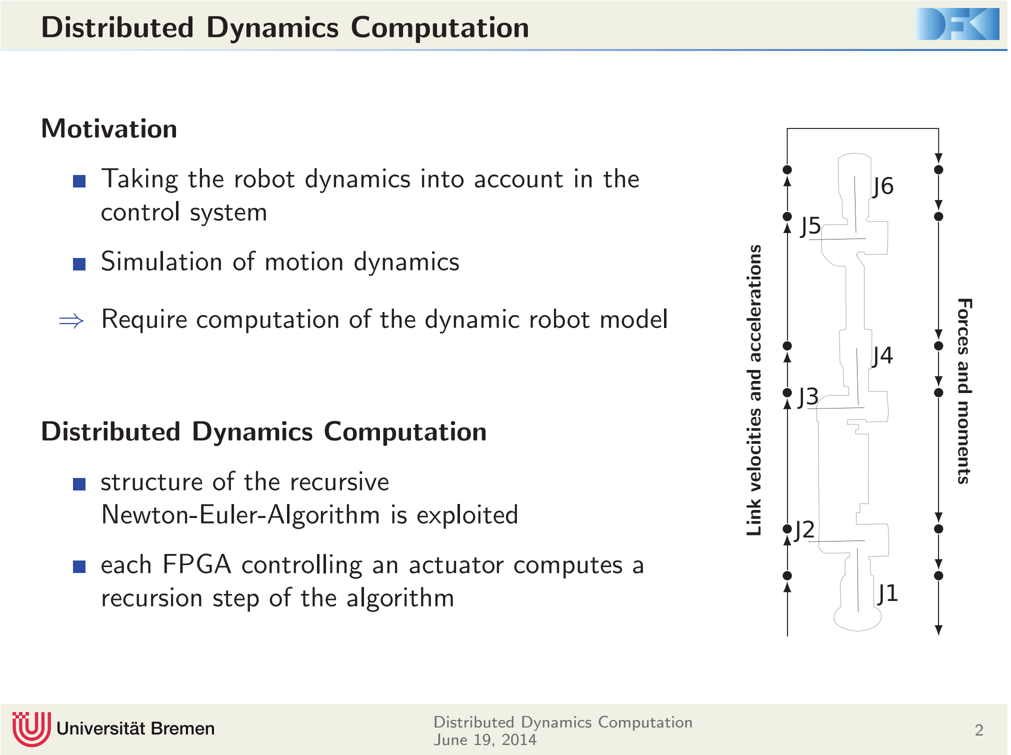
Introduction to the Poster:

Distributed Dynamics Computation

Vinzenz Bargsten

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



Distributed Dynamics Computation

Motivation

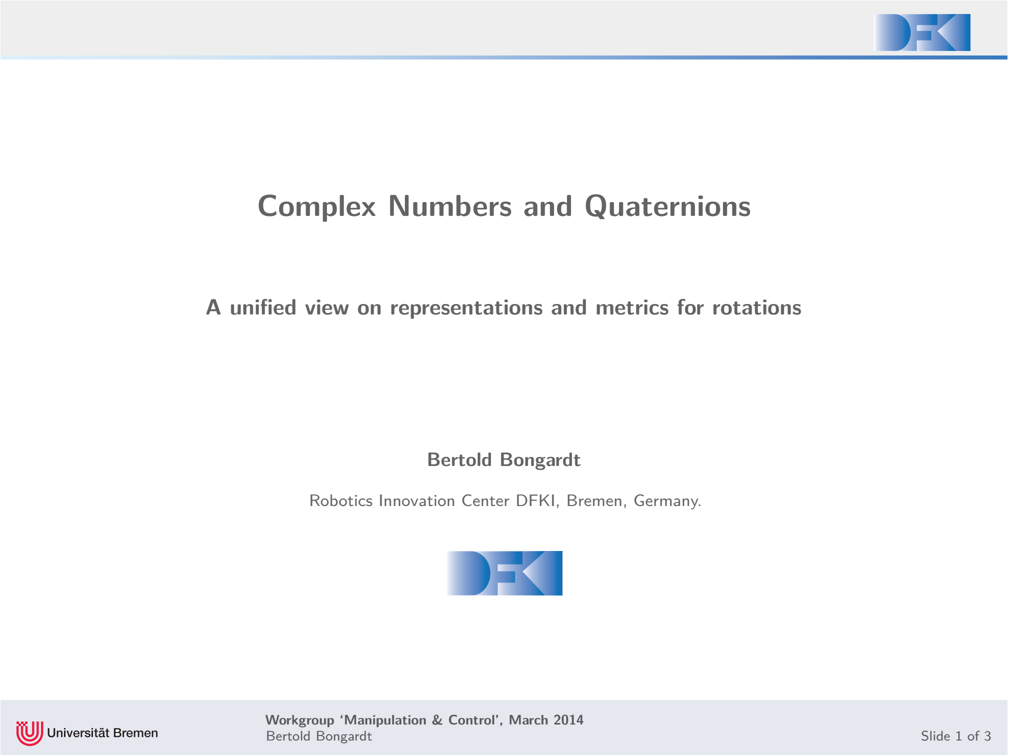
- Taking the robot dynamics into account in the control system
- Simulation of motion dynamics

⇒ Require computation of the dynamic robot model

Distributed Dynamics Computation

- structure of the recursive Newton-Euler-Algorithm is exploited
- each FPGA controlling an actuator computes a recursion step of the algorithm

Poster #04 – Bertold

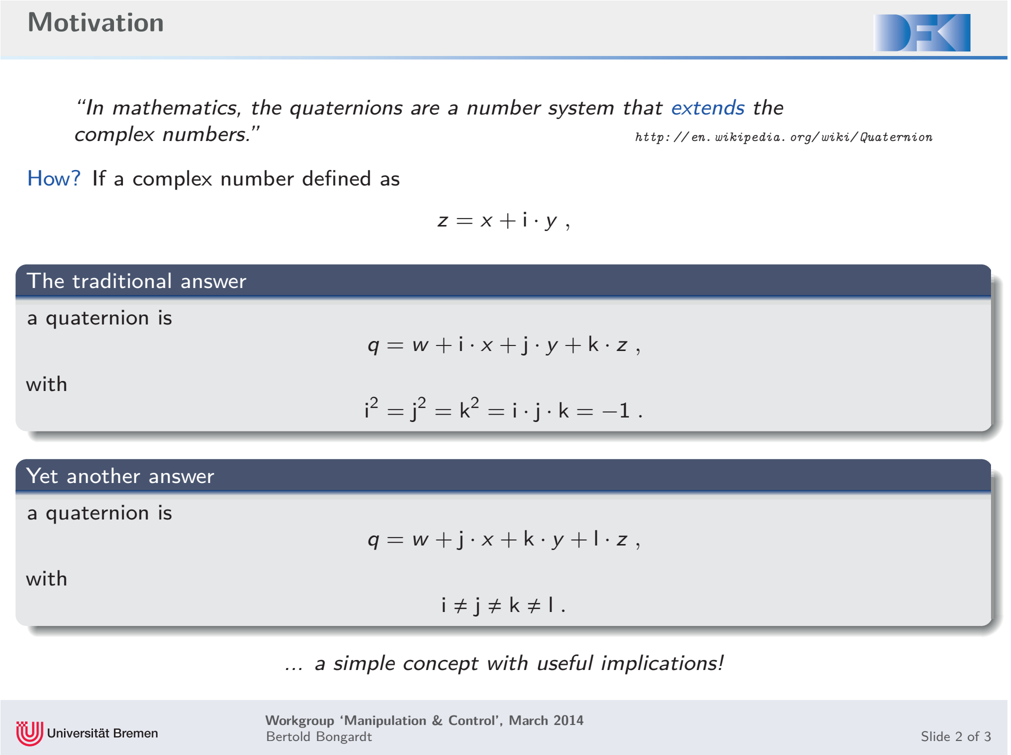


Complex Numbers and Quaternions

A unified view on representations and metrics for rotations

Bertold Bongardt

Robotics Innovation Center DFKI, Bremen, Germany.



Motivation

> *“In mathematics, the quaternions are a number system that extends the complex numbers.”* http://en.wikipedia.org/wiki/Quaternion

How? If a complex number defined as

$$z = x + \mathrm{i} \cdot y \, ,$$

The traditional answer

a quaternion is

$$q = w + \mathrm{i} \cdot x + \mathrm{j} \cdot y + \mathrm{k} \cdot z \, ,$$

with

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{i} \cdot \mathrm{j} \cdot \mathrm{k} = -1 \, .$$

Yet another answer

a quaternion is

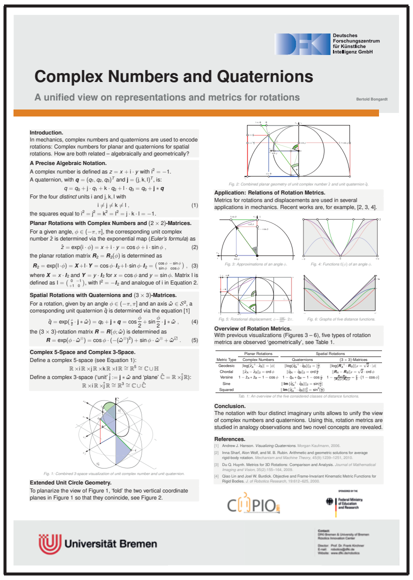
$$q = w + \mathrm{j} \cdot x + \mathrm{k} \cdot y + \mathrm{l} \cdot z \, ,$$

with

$$\mathrm{i} \neq \mathrm{j} \neq \mathrm{k} \neq \mathrm{l} \, .$$

... a simple concept with useful implications!

More Information



Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Complex Numbers and Quaternions

A unified view on representations and metrics for rotations

Bertold Bongardt

Introduction.
In mechanics, complex numbers and quaternions are used to encode rotations: Complex numbers for planar and quaternions for spatial rotations. How are both related – algebraically and geometrically?

A Precise Algebraic Notation.
A complex number is defined as $z = x + \mathrm{i} \cdot y$ with $\mathrm{i}^2 = -1$.
A quaternion, with $\boldsymbol{q} = (q_1, q_2, q_3)^T$ and $\mathbf{j} = (\mathrm{j}, \mathrm{k}, \mathrm{l})^T$, is:

$$q = q_0 + \mathrm{j} \cdot q_1 + \mathrm{k} \cdot q_2 + \mathrm{l} \cdot q_3 = q_0 + \mathbf{j} * \boldsymbol{q}$$

For the four *distinct* units i and j, k, l with

$$\mathrm{i} \neq \mathrm{j} \neq \mathrm{k} \neq \mathrm{l}\,, \qquad (1)$$

the squares equal to $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{l}^2 = \mathrm{j} \cdot \mathrm{k} \cdot \mathrm{l} = -1$.

Planar Rotations with Complex Numbers and (2×2)-Matrices.
For a given angle, $\phi \in (-\pi, \pi]$, the corresponding unit complex number $\hat{z}$ is determined via the exponential map (*Euler's formula*) as

$$\hat{z} = \exp(\mathrm{i} \cdot \phi) = x + \mathrm{i} \cdot y = \cos\phi + \mathrm{i} \cdot \sin\phi\,, \qquad (2)$$

the planar rotation matrix $\boldsymbol{R}_2 = \boldsymbol{R}_2(\phi)$ is determined as

$$\boldsymbol{R}_2 = \exp(\mathrm{I} \cdot \phi) = \boldsymbol{X} + \mathrm{I} \cdot \boldsymbol{Y} = \cos\phi \cdot \boldsymbol{I}_2 + \mathrm{I} \cdot \sin\phi \cdot \boldsymbol{I}_2 = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}, \qquad (3)$$

where $\boldsymbol{X} = x \cdot \boldsymbol{I}_2$ and $\boldsymbol{Y} = y \cdot \boldsymbol{I}_2$ for $x = \cos\phi$ and $y = \sin\phi$. Matrix I is defined as $\mathrm{I} = \begin{pmatrix} 0 & -1 \\ +1 & 0 \end{pmatrix}$, with $\mathrm{I}^2 = -\boldsymbol{I}_2$ and analogue of i in Equation 2.

Spatial Rotations with Quaternions and (3×3)-Matrices.
For a rotation, given by an angle $\phi \in (-\pi, \pi]$ and an axis $\hat{\omega} \in \mathcal{S}^2$, a corresponding unit quaternion $\hat{q}$ is determined via the equation [1]

$$\hat{q} = \exp(\tfrac{\phi}{2} \cdot \mathbf{j} * \hat{\omega}) = q_0 + \mathbf{j} * \boldsymbol{q} = \cos\frac{\phi}{2} + \sin\frac{\phi}{2} \cdot \mathbf{j} * \hat{\omega}\,, \qquad (4)$$

the (3×3)-rotation matrix $\boldsymbol{R} = \boldsymbol{R}(\phi; \hat{\omega})$ is determined as

$$\boldsymbol{R} = \exp(\phi \cdot \hat{\omega}^{\otimes}) = \cos\phi \cdot (-(\hat{\omega}^{\otimes})^2) + \sin\phi \cdot \hat{\omega}^{\otimes} + \hat{\omega}^{\boxtimes}\,. \qquad (5)$$

Complex 5-Space and Complex 3-Space.
Define a complex 5-space (see Equation 1):

$$\mathbb{R} \times \mathrm{i}\mathbb{R} \times \mathrm{j}\mathbb{R} \times \mathrm{k}\mathbb{R} \times \mathrm{l}\mathbb{R} \cong \mathbb{R}^5 \cong \mathbb{C} \cup \mathbb{H}$$

Define a complex 3-space ('unit' $\hat{\mathrm{j}} := \mathbf{j} * \hat{\omega}$ and 'plane' $\hat{\mathbb{C}} = \mathbb{R} \times \hat{\mathrm{j}}\mathbb{R}$):

$$\mathbb{R} \times \mathrm{i}\mathbb{R} \times \hat{\mathrm{j}}\mathbb{R} \cong \mathbb{R}^3 \cong \mathbb{C} \cup \hat{\mathbb{C}}$$

Fig. 1: Combined 3-space visualization of unit complex number and unit quaternion.

Extended Unit Circle Geometry.
To planarize the view of Figure 1, 'fold' the two vertical coordinate planes in Figure 1 so that they conincide, see Figure 2.

Fig. 2: Combined planar geometry of unit complex number $\hat{z}$ and unit quaternion $\hat{q}$.

Application: Relations of Rotation Metrics.
Metrics for rotations and displacements are used in several applications in mechanics. Recent works are, for example, [2, 3, 4].

Fig. 3: Approximations of an angle ϕ. *Fig. 4: Functions $f_i(\phi)$ of an angle ϕ.*

Fig. 5: Rotational displacement, $\phi = \frac{108°}{360°} \cdot 2\pi$. *Fig. 6: Graphs of five distance functions.*

Overview of Rotation Metrics.
With previous visualizations (Figures 3 – 6), five types of rotation metrics are observed 'geometrically', see Table 1.

	Planar Rotations	Spatial Rotations	
Metric Type	Complex Numbers	Quaternions	(3×3)-Matrices
Geodesic	$\lVert\log(\hat{z}_A^{-1} \cdot \hat{z}_B)\rVert = \lvert\phi\rvert$	$\lVert\log(\hat{q}_A^{-1} \cdot \hat{q}_B)\rVert_2 = \lvert\frac{\phi}{2}\rvert$	$\lVert\log(\boldsymbol{R}_A^{-1} \cdot \boldsymbol{R}_B)\rVert_F = \sqrt{2} \cdot \lvert\phi\rvert$
Chordal	$\lVert\hat{z}_A - \hat{z}_B\rVert_2 = \mathrm{crd}\,\phi$	$\lVert\hat{q}_A - \hat{q}_B\rVert_2 = \mathrm{crd}\,\frac{\phi}{2}$	$\lVert\boldsymbol{R}_A - \boldsymbol{R}_B\rVert_F = \sqrt{2} \cdot \mathrm{crd}\,\phi$
Versine	$1 - \hat{z}_A * \hat{z}_B = 1 - \cos\phi$	$1 - \hat{q}_A * \hat{q}_B = 1 - \cos\frac{\phi}{2}$	$1 - \frac{\boldsymbol{R}_A \cdot \boldsymbol{R}_B}{\lVert\boldsymbol{R}_A\rVert_F \lVert\boldsymbol{R}_A\rVert_F} = \frac{2}{3} \cdot (1 - \cos\phi)$
Sine		$\lVert\mathbf{Im}\,(\hat{q}_A^{-1} \cdot \hat{q}_B)\rVert_2 = \sin\lvert\frac{\phi}{2}\rvert$	
Squared		$\lVert\mathbf{Im}\,(\hat{q}_A^{-1} \cdot \hat{q}_B)\rVert_2^2 = \sin^2(\frac{\phi}{2})$	

Tab. 1: An overview of the five considered classes of distance functions.

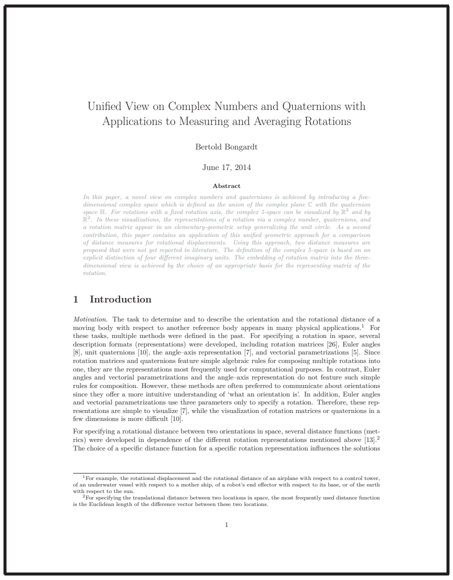
Conclusion.
The notation with four distinct imaginary units allows to unify the view of complex numbers and quaternions. Using this, rotation metrics are studied in analogy observations and two novel concepts are revealed.

References.

[1] Andrew J. Hanson. *Visualizing Quaternions.* Morgan Kaufmann, 2006.
[2] Inna Sharf, Alon Wolf, and M. B. Rubin. Arithmetic and geometric solutions for average rigid-body rotation. *Mechanism and Machine Theory*, 45(9):1239–1251, 2010.
[3] Du Q. Huynh. Metrics for 3D Rotations: Comparison and Analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.
[4] Qiao Lin and Joel W. Burdick. Objective and Frame-Invariant Kinematic Metric Functions for Rigid Bodies. *J. of Robotics Research*, 19:612–625, 2000.

SPONSORED BY THE Federal Ministry of Education and Research

Universität Bremen



Unified View on Complex Numbers and Quaternions with Applications to Measuring and Averaging Rotations

Bertold Bongardt

June 17, 2014

Abstract

In this paper, a novel view on complex numbers and quaternions is achieved by introducing a five-dimensional complex space which is defined as the union of the complex plane $\mathbb{C}$ with the quaternion space $\mathbb{H}$. For rotations with a fixed rotation axis, the complex 5-space can be visualized by $\mathbb{R}^3$ and by $\mathbb{R}^2$. In these visualizations, the representations of a rotation via a complex number, quaternions, and a rotation matrix appear in an elementary-geometric setup generalizing the unit circle. As a second contribution, this paper contains an application of this unified geometric approach for a comparison of distance measures for rotational displacements. Using this approach, two distance measures are proposed that were not yet reported in literature. The definition of the complex 5-space is based on an explicit distinction of four different imaginary units. The embedding of rotation matrix into the three-dimensional view is achieved by the choice of an appropriate basis for the representing matrix of the rotation.

1 Introduction

Motivation. The task to determine and to describe the orientation and the rotational distance of a moving body with respect to another reference body appears in many physical applications.[1] For these tasks, multiple methods were defined in the past. For specifying a rotation in space, several description formats (representations) were developed, including rotation matrices [26], Euler angles [8], unit quaternions [10], the angle–axis representation [7], and vectorial parametrizations [5]. Since rotation matrices and quaternions feature simple algebraic rules for composing multiple rotations into one, they are the representations most frequently used for computational purposes. In contrast, Euler angles and vectorial parametrizations and the angle–axis representation do not feature such simple rules for composition. However, these methods are often preferred to communicate about orientations since they offer a more intuitive understanding of 'what an orientation is'. In addition, Euler angles and vectorial parametrizations use three parameters only to specify a rotation. Therefore, these representations are simple to visualize [7], while the visualization of rotation matrices or quaternions in a few dimensions is more difficult [10].

For specifying a rotational distance between two orientations in space, several distance functions (metrics) were developed in dependence of the different rotation representations mentioned above [13].[2] The choice of a specific distance function for a specific rotation representation influences the solutions

---

[1] For example, the rotational displacement and the rotational distance of an airplane with respect to a control tower, of an underwater vessel with respect to a mother ship, of a robot's end effector with respect to its base, or of the earth with respect to the sun.

[2] For specifying the translational distance between two locations in space, the most frequently used distance function is the Euclidean length of the difference vector between these two locations.

1

Poster #05 – Dennis

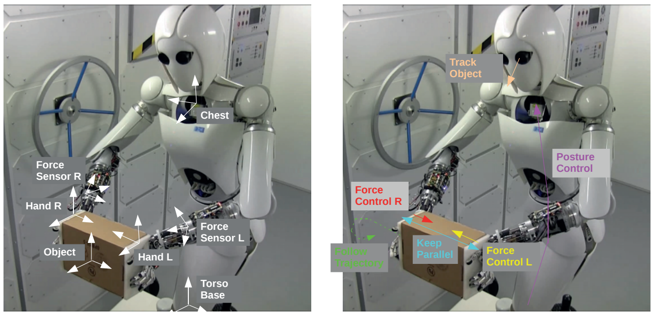
iTaSC: Application and Rock Integration

Specification and Control of Complex Sensor-Based Tasks

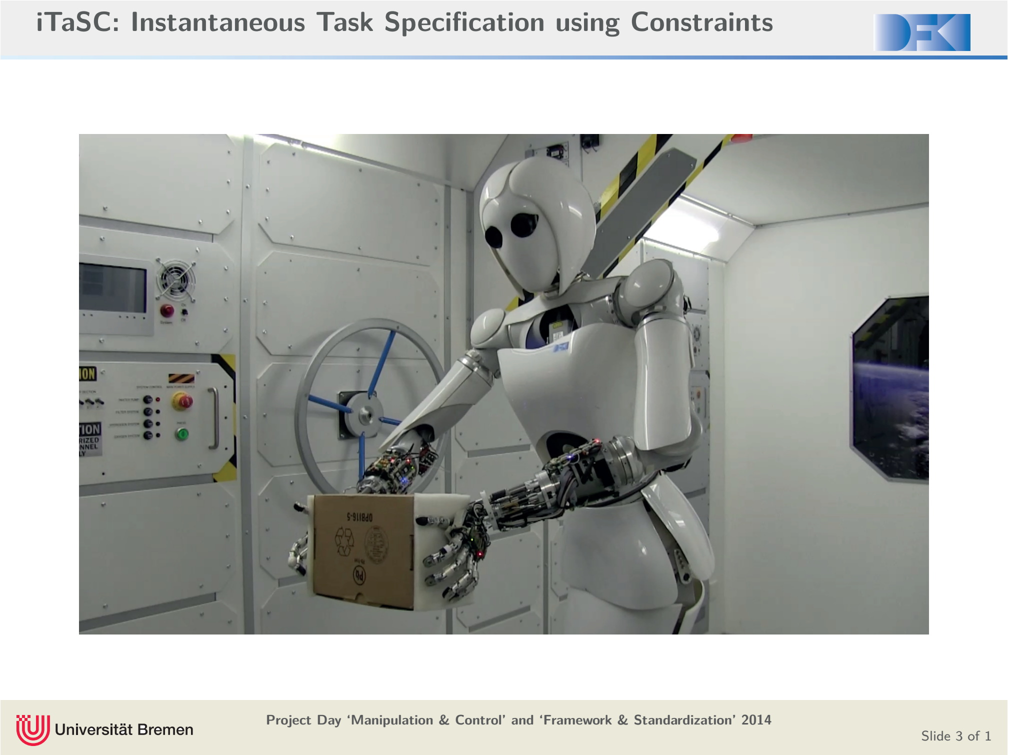
Dennis Mronga

Robotics Innovation Center DFKI, Bremen, Germany.

iTaSC: Instantaneous Task Specification using Constraints



- What is iTaSC and what are its capabilities?
- How is it integrated in Rock?



iTaSC: Instantaneous Task Specification using Constraints

Schedule

Schedule 2014-06-19

Schedule Current activities in ‘AG Manipulation & Control’ Schedule

09:00 - 09:01 Hello!

09:01	"Overview of the new RIC project day structure" (Sirko Straube)	(6 + 3)

Block I – Talks – AG Framework & Standardization

09:10	General Introduction (Alexander Duda)	(15 + 5)
09:20	Cross platform development (Martin Zenses)	(15 + 5)
09:40	The Artemis Rover as an Example for Model Based Engineering in Space Robotics (Stefan Haase)	(15 + 5)
10:00	Distributed compilation and Buildserver (Steffen Planthaber)	(15 + 5)
10:20	Data Distribution Service (DDS) (Ronny Hartanto)	(15 + 5)

10:40 – 11:00 Coffee Break

Block II – Talks – AG Manipulation & Control

11:00	Work Group Activities – Introduction (Bertold Bongardt, *)	(14 + 1)
11:15	Control of Flexible Link Manipulator (Ajish Babu)	(14 + 1)
11:30	Motion Planning for Manipulator using Moveit (Sankar Natarajan)	(14 + 1)
11:45	A Library for Motion Planning (Behnam Asadi)	(14 + 1)
12:00	Trajectory generation using the library Reflexxes (Benjamin Girault, Malte Wirkus)	(14 + 1)
12:15	Cascaded Robot Joint Control (Vinzenz Bargsten)	(14 + 1)

12:30 – 12:45 Lunch Break (Move to Foyer)

Block III – Posters & Discussion – AG Framework & Standardization

- FIPA-based multi-agent infrastructure for Rock (Satia Herfert, Thomas Röhr)
- Planning with Reconfigurable Multi-Robot Systems (Thomas Röhr, Ronny Hartanto)

Block IV – Posters & Discussion – AG Manipulation & Control

- Autonomous Steering Controller for Path Following (Mohammed Ahmed, Ajish Babu)
- Motion Planning for Manipulators (Behnam Asadi, Sankar Natarajan)
- Distributed Dynamics Computation (Vinzenz Bargsten)
- Complex Numbers and Quaternions - A unified view on representations and metrics for rotations (Bertold Bongardt)
- An Application of Constraint-Based Motion Control (iTaSC) (Dennis Mronga)

16:20 - 16:30 Clean-Up!

Universität Bremen

Project Day ‘Manipulation & Control’ and ‘Framework & Standardization’ 2014

Slide 14 of 14

3.2 'Control of Flexible Link Manipulator' (MC-T-02)

Ajish Babu(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `ajish.babu@dfki.de`

Abstract

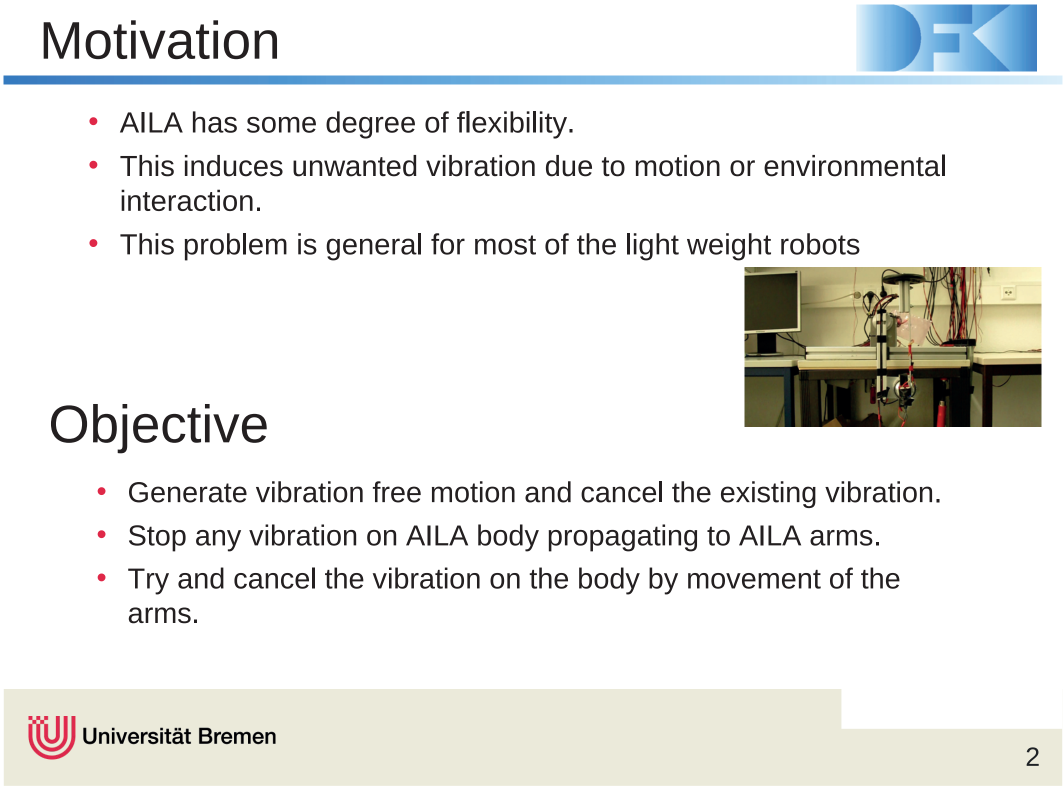
Light-weight robot design have a certain degree of flexibility, which in turn introduces unwanted vibrations. This talk introduces different methods of vibration attenuation, combining feed-forward and feedback techniques. Feed-forward compensation based on Input Shaping and feedback control based on Strain gauges and Inertial Measurement Unit are used. The effectiveness of these controllers are studied using a two-link manipulator test setup.

Control of Flexible Link Manipulator

by Ajish Babu

Project-day AG Manipulation / AG Framework & Standardization
19th June 2014

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

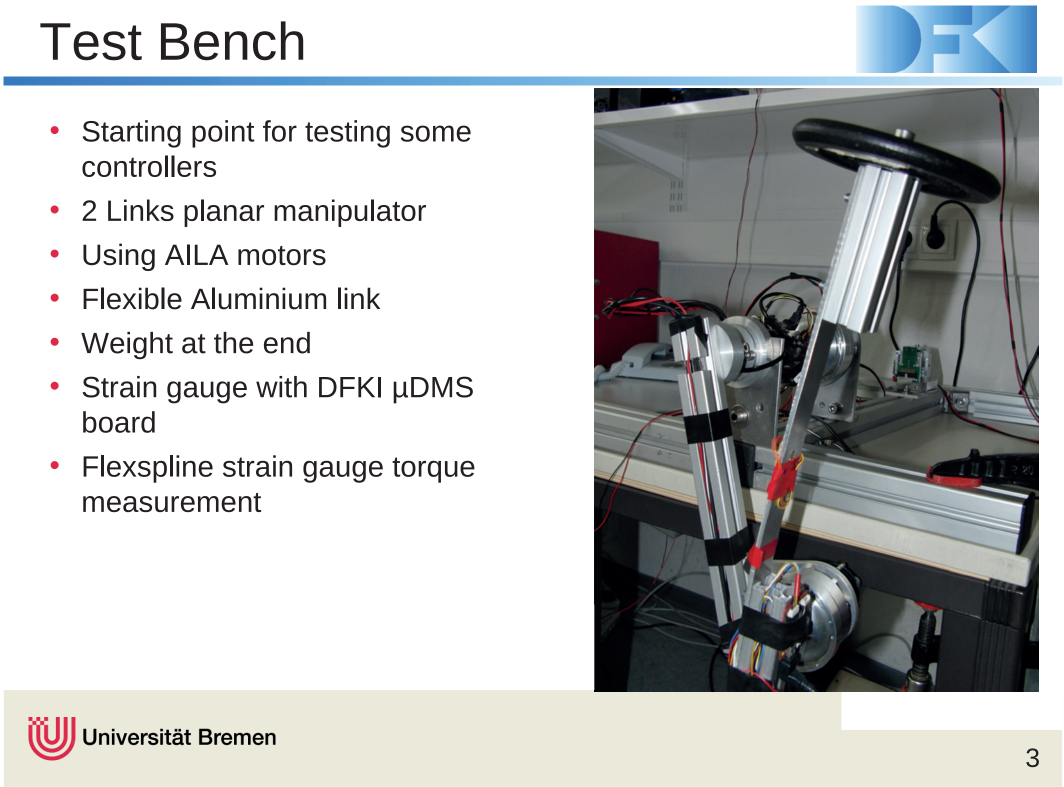


Motivation

- AILA has some degree of flexibility.
- This induces unwanted vibration due to motion or environmental interaction.
- This problem is general for most of the light weight robots

Objective

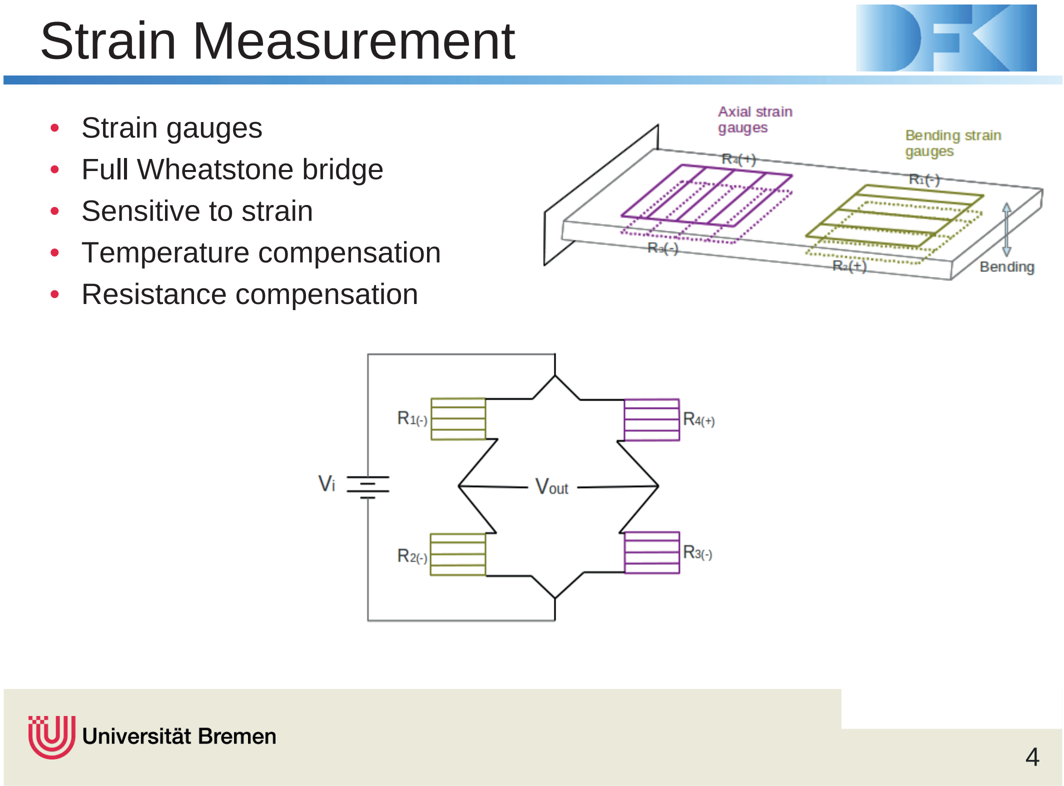
- Generate vibration free motion and cancel the existing vibration.
- Stop any vibration on AILA body propagating to AILA arms.
- Try and cancel the vibration on the body by movement of the arms.



Test Bench

- Starting point for testing some controllers
- 2 Links planar manipulator
- Using AILA motors
- Flexible Aluminium link
- Weight at the end
- Strain gauge with DFKI µDMS board
- Flexspline strain gauge torque measurement

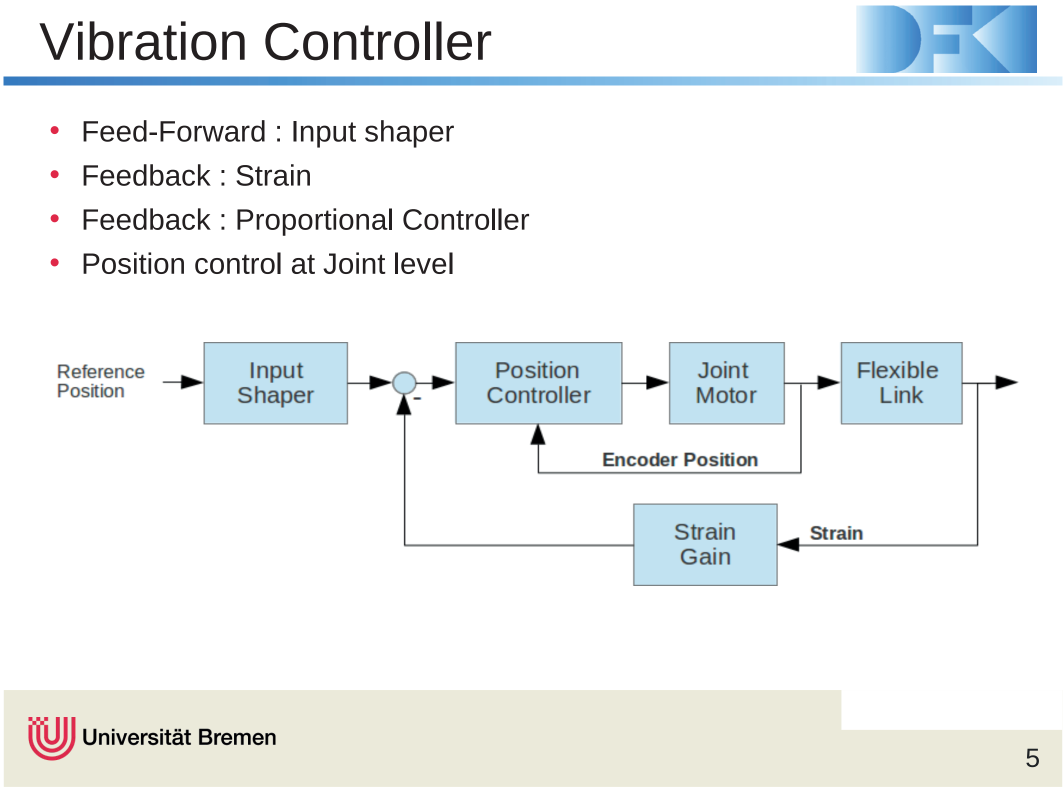
Universität Bremen



Strain Measurement

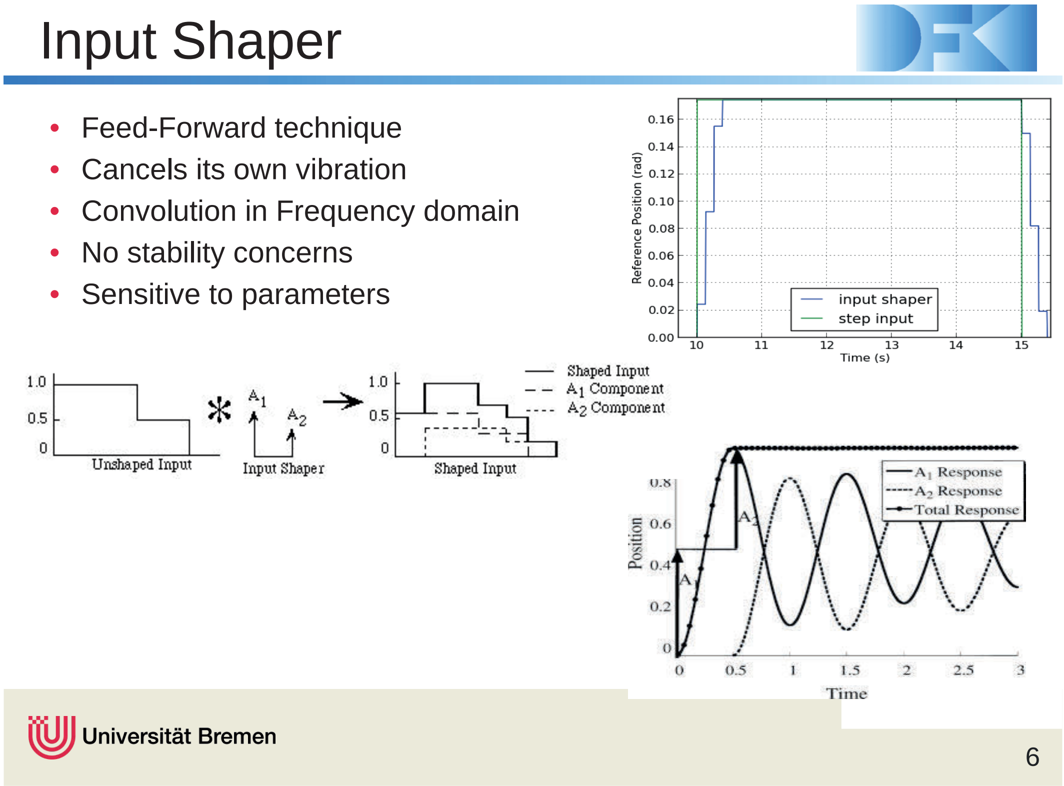
- Strain gauges
- Full Wheatstone bridge
- Sensitive to strain
- Temperature compensation
- Resistance compensation

Universität Bremen



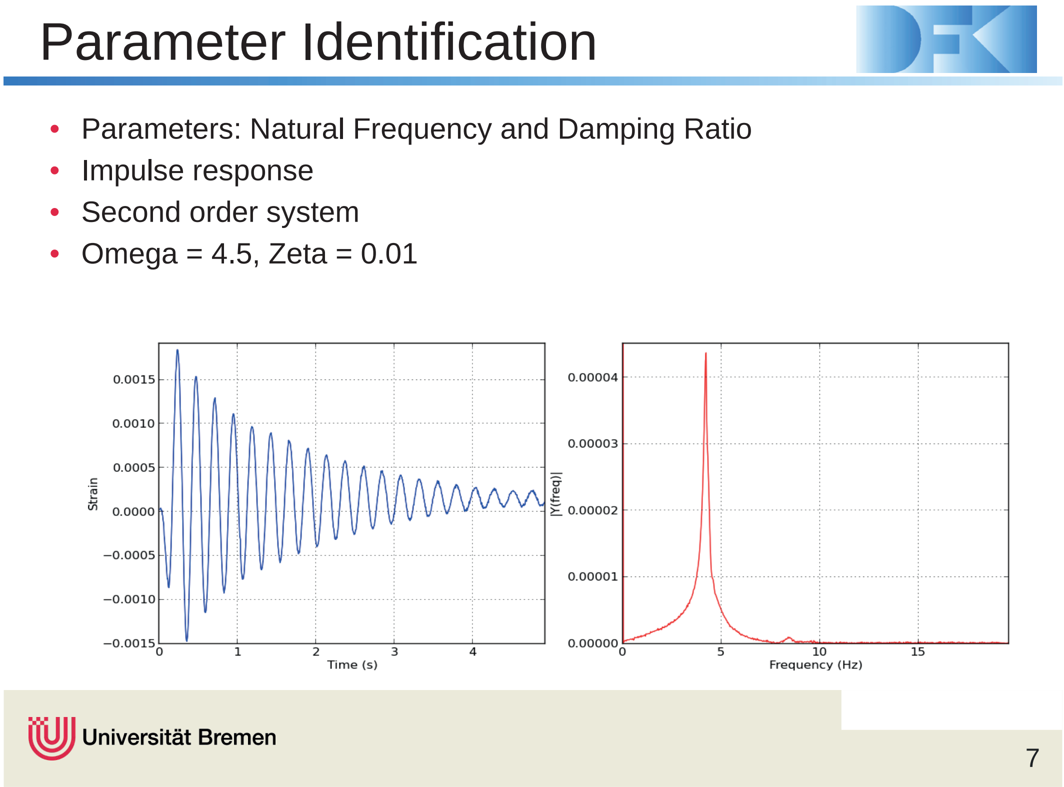
Vibration Controller

- Feed-Forward : Input shaper
- Feedback : Strain
- Feedback : Proportional Controller
- Position control at Joint level



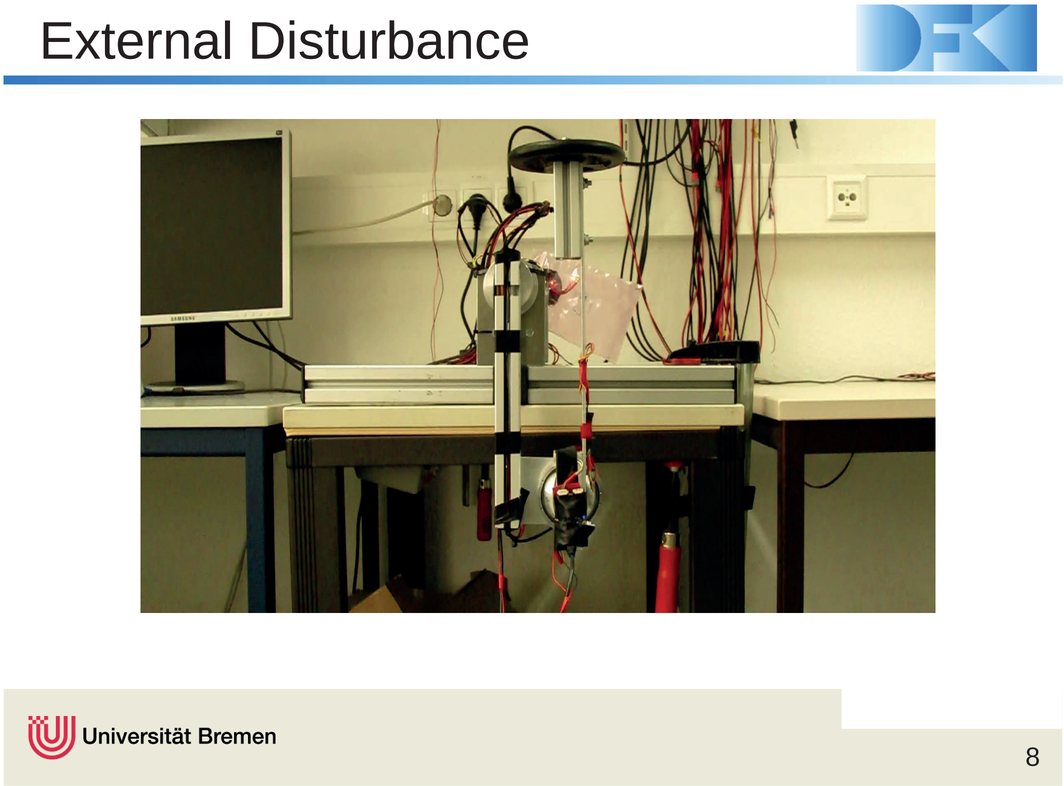
Input Shaper

- Feed-Forward technique
- Cancels its own vibration
- Convolution in Frequency domain
- No stability concerns
- Sensitive to parameters

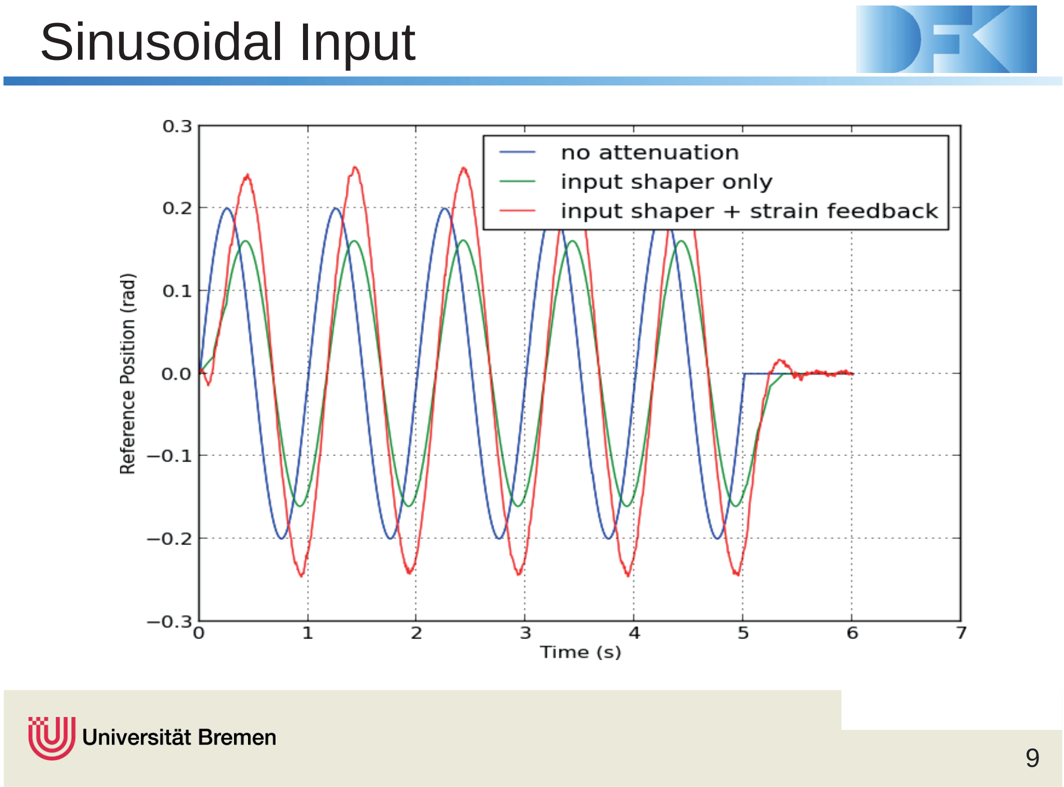


Parameter Identification

- Parameters: Natural Frequency and Damping Ratio
- Impulse response
- Second order system
- Omega = 4.5, Zeta = 0.01

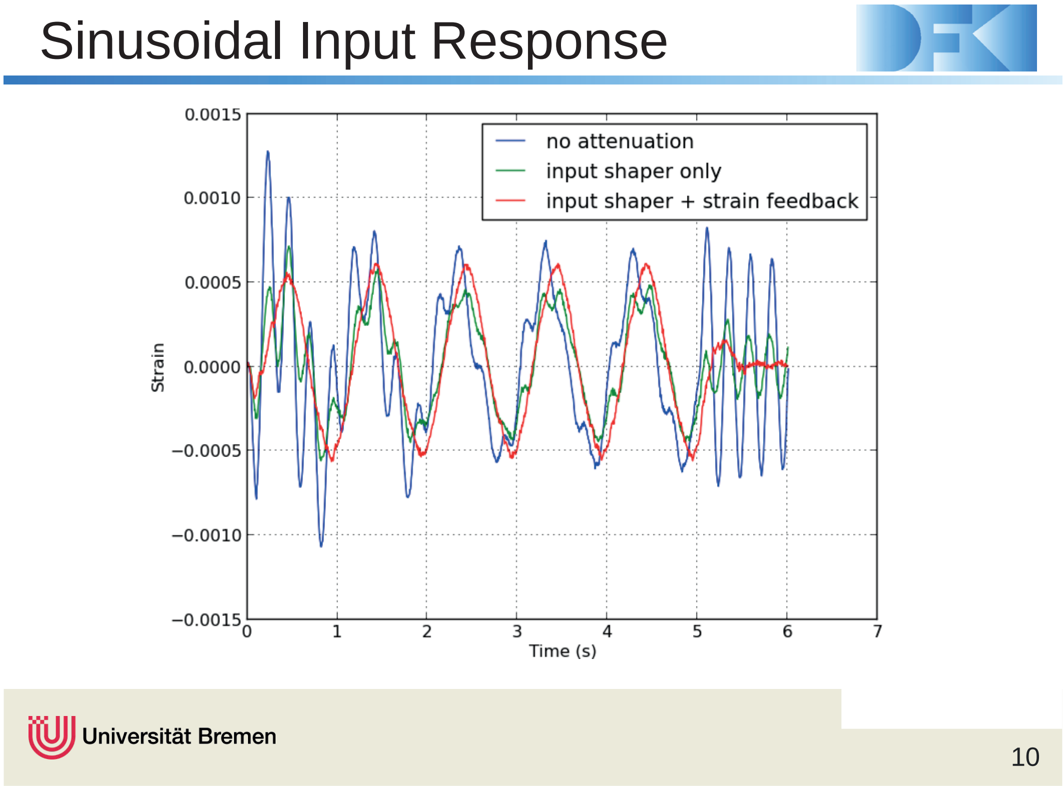


External Disturbance



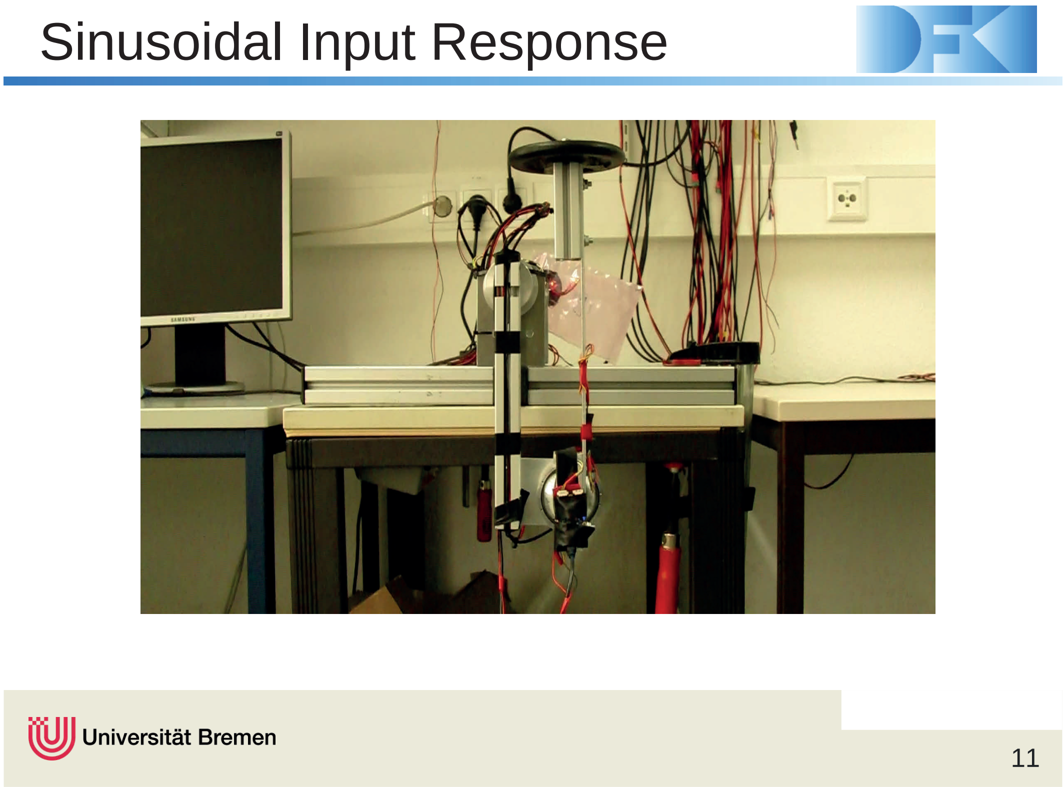
Sinusoidal Input

Universität Bremen

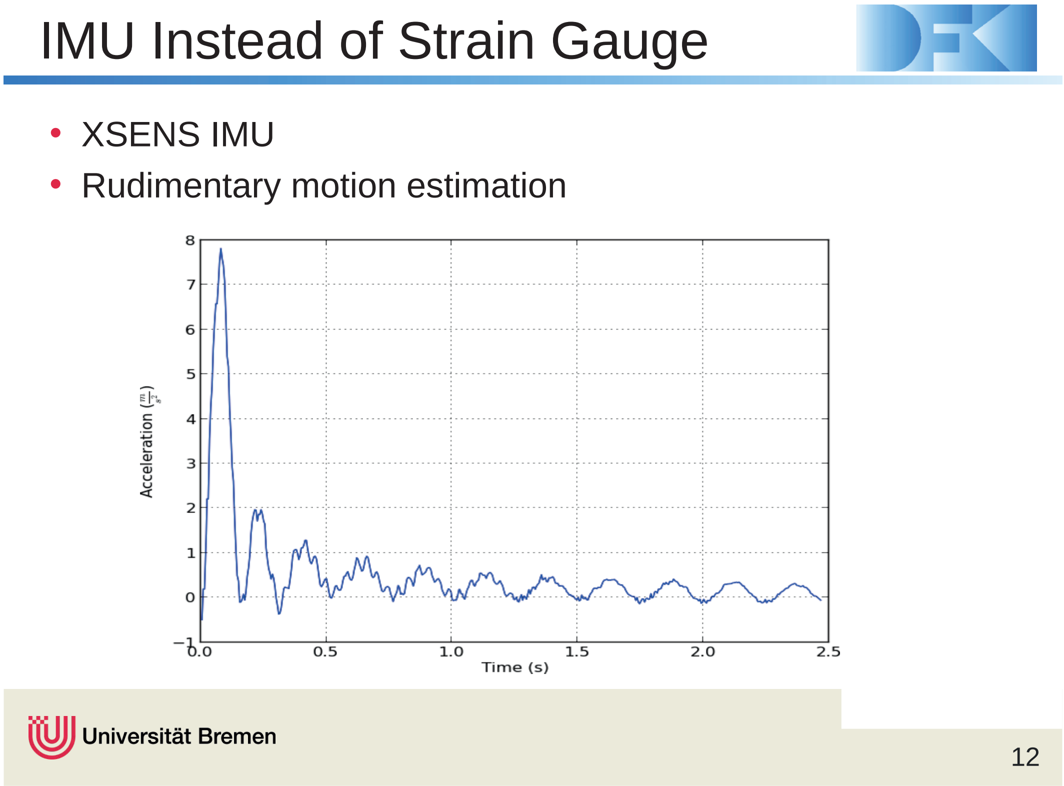


Sinusoidal Input Response

Universität Bremen

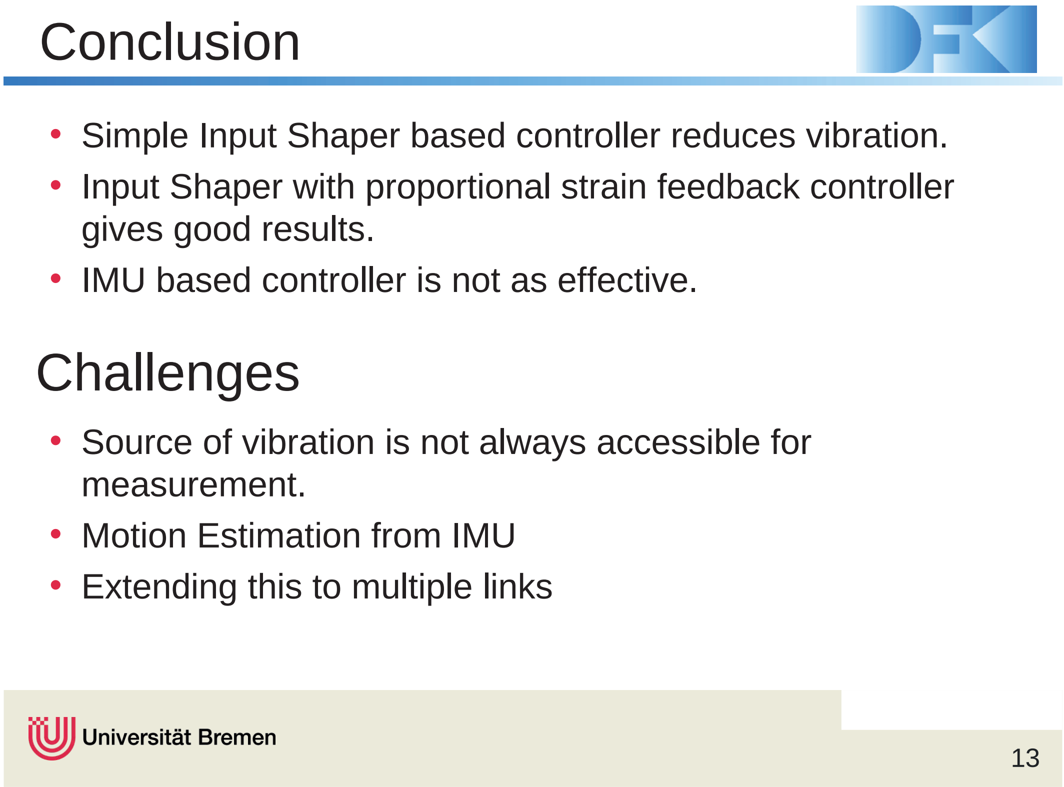


Sinusoidal Input Response



IMU Instead of Strain Gauge

- XSENS IMU
- Rudimentary motion estimation

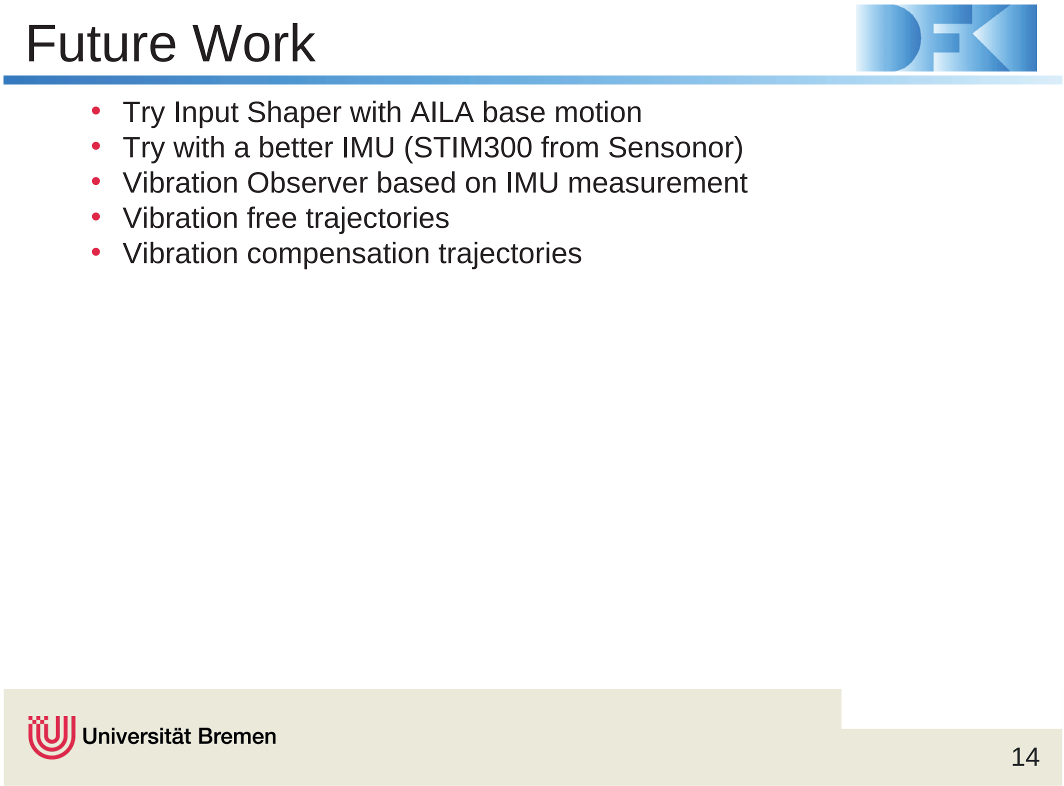


Conclusion

- Simple Input Shaper based controller reduces vibration.
- Input Shaper with proportional strain feedback controller gives good results.
- IMU based controller is not as effective.

Challenges

- Source of vibration is not always accessible for measurement.
- Motion Estimation from IMU
- Extending this to multiple links



Future Work

- Try Input Shaper with AILA base motion
- Try with a better IMU (STIM300 from Sensonor)
- Vibration Observer based on IMU measurement
- Vibration free trajectories
- Vibration compensation trajectories

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

3.3 'Motion Planning for Manipulator – *MoveIt!*' (MC-T-03)

Sankaranarayanan Natarajan(1)

(1) *DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany*

Contact: `sankaranarayanan.natarajan@dfki.de`

Abstract

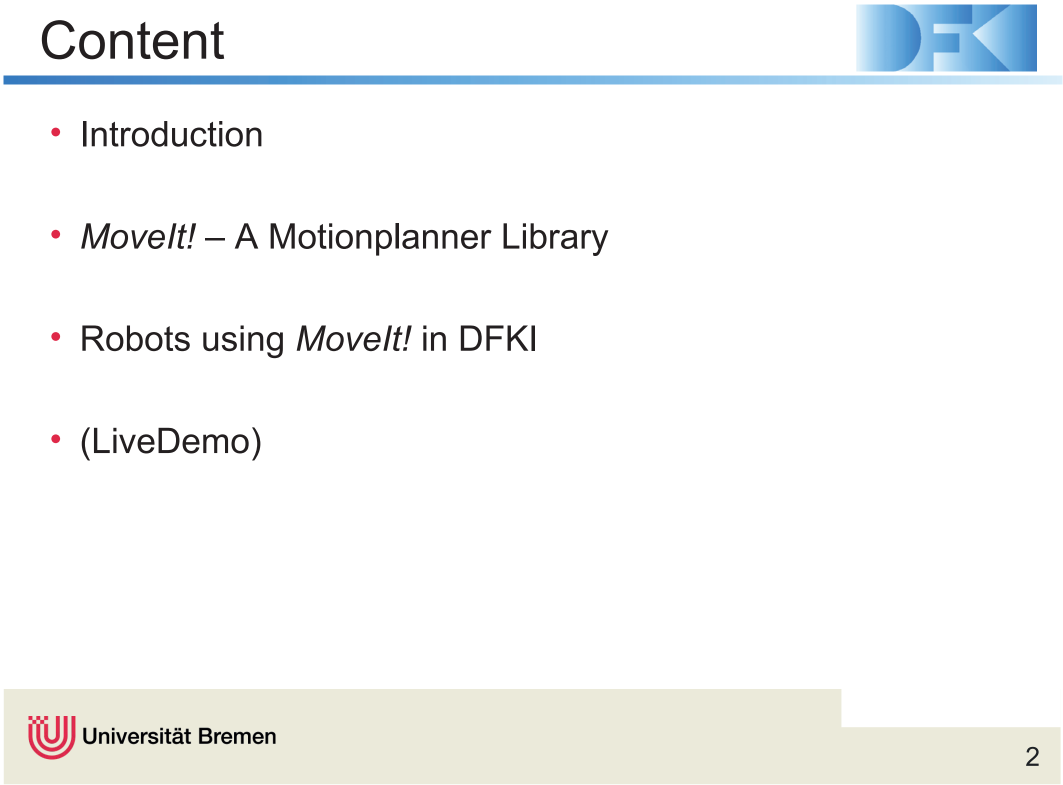
This talk gives a brief introduction on a motion planner framework *MoveIt!* . The *MoveIt!* framework provides the state of the art motion planners, kinematic solver, collision checker, control and navigation software modules. The key features of the *MoveIt!* like environment representation and set up assistant are discussed.

Motion Planning for Manipulator

- *MoveIt!* -

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



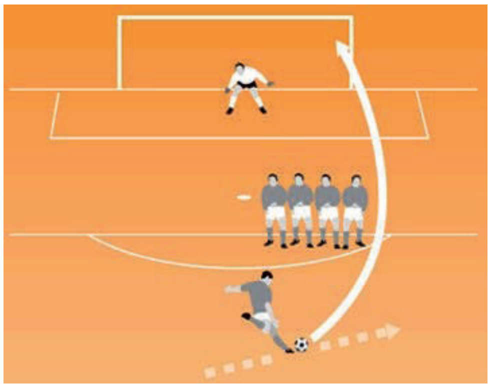
Content

- Introduction
- *MoveIt!* – A Motionplanner Library
- Robots using *MoveIt!* in DFKI
- (LiveDemo)

Universität Bremen

Introduction

- MotionPlanning
  - Ability of a robot to plan its own motion considering the given constraints
    - Constraints: obstacles in the environment, kinematic constraints.
  - Motion planner is a collection of software modules
    - Object modeling
    - Collision detection
    - Search algorithm

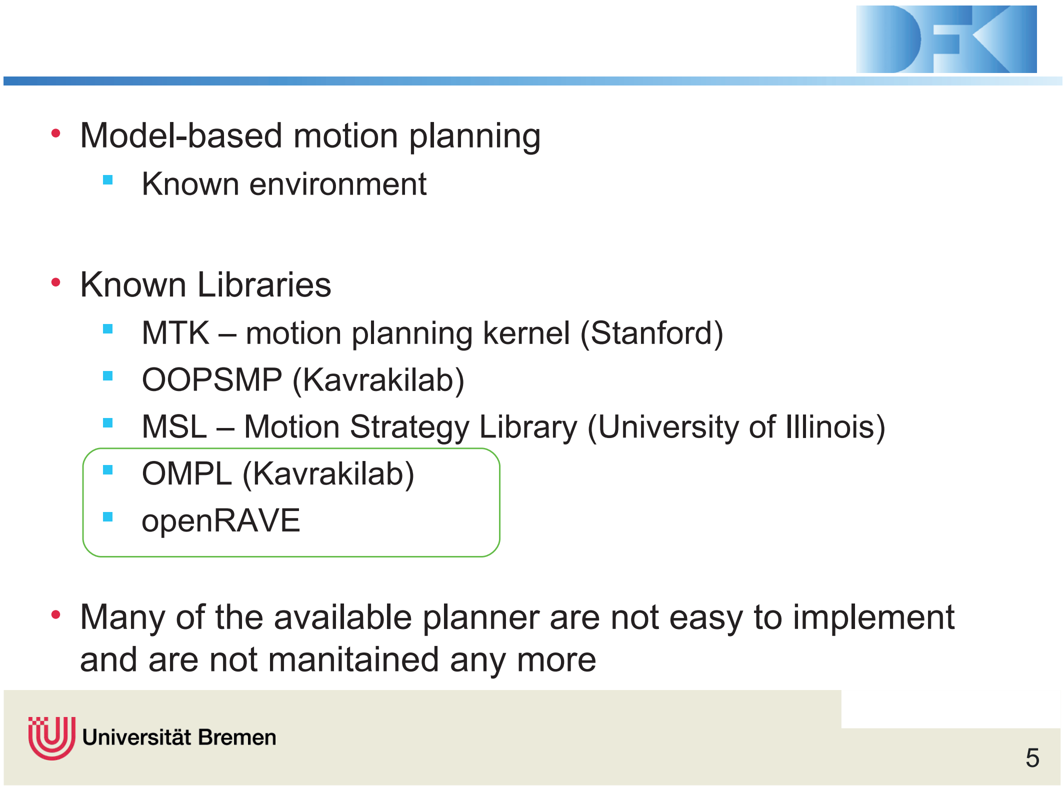


Developing a motion planner is not an easy task

- Model-based motion planning
  - Known environment

- Known Libraries
  - MTK – motion planning kernel (Stanford)
  - OOPSMP (Kavrakilab)
  - MSL – Motion Strategy Library (University of Illinois)
  - OMPL (Kavrakilab)
  - openRAVE

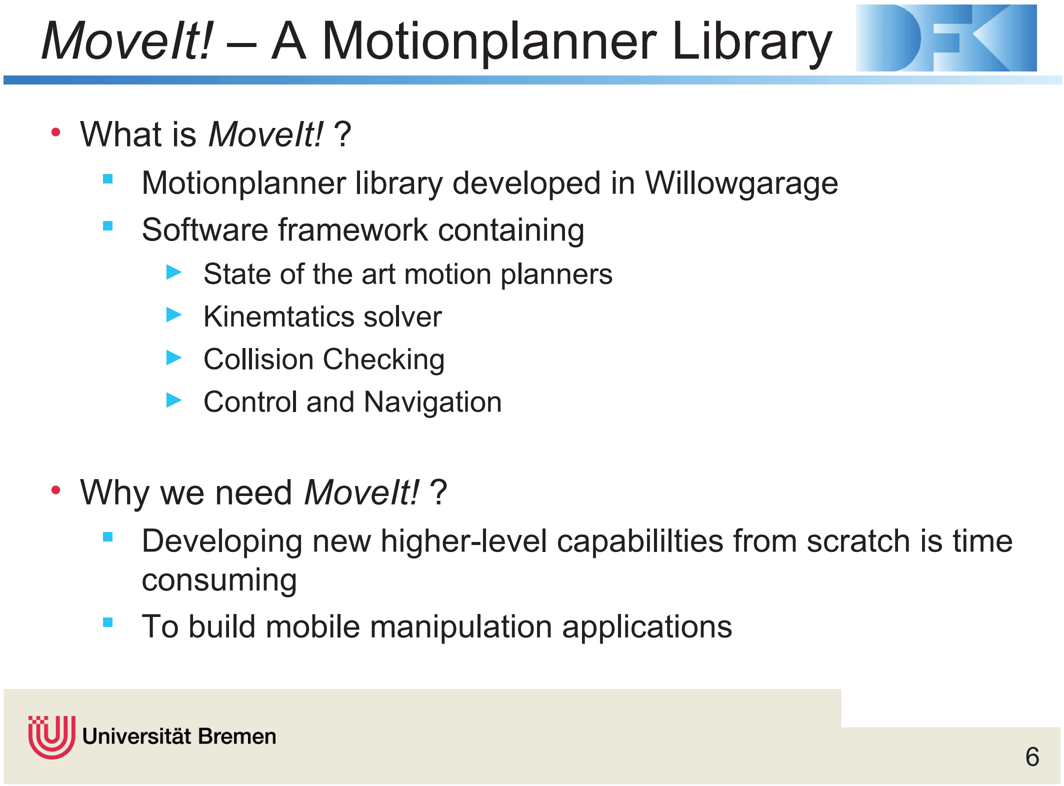
- Many of the available planner are not easy to implement and are not manitained any more



- Model-based motion planning
  - Known environment

- Known Libraries
  - MTK – motion planning kernel (Stanford)
  - OOPSMP (Kavrakilab)
  - MSL – Motion Strategy Library (University of Illinois)
  - OMPL (Kavrakilab)
  - openRAVE

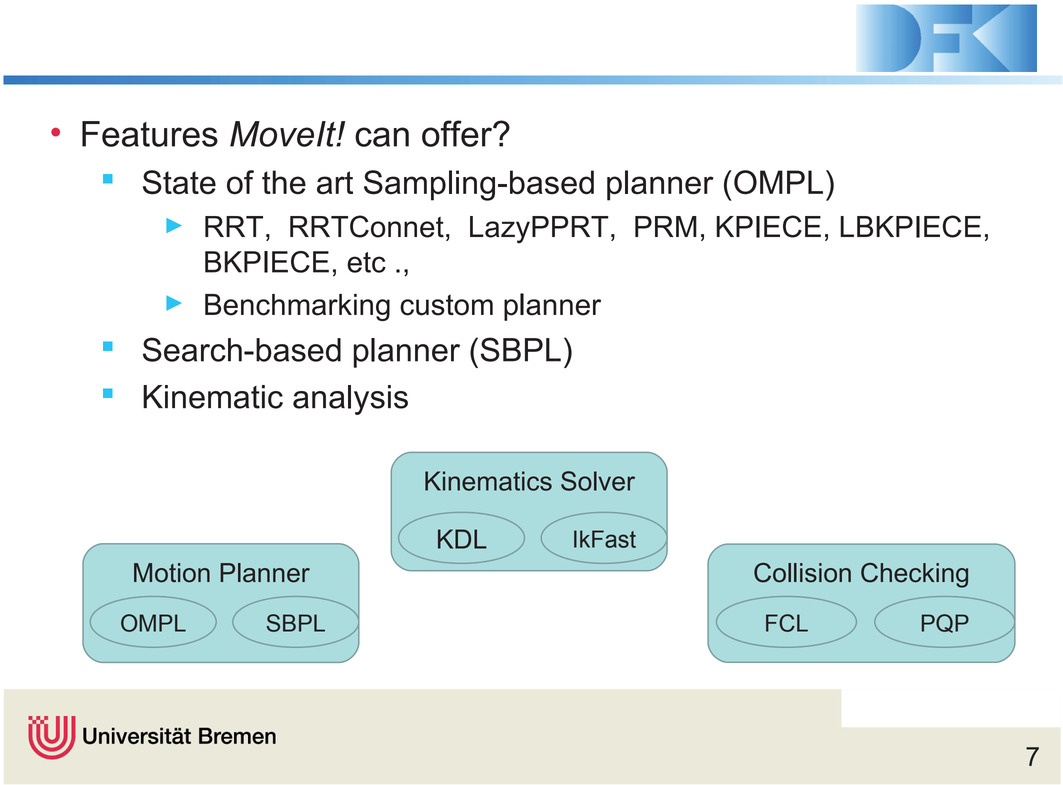
- Many of the available planner are not easy to implement and are not manitained any more



MoveIt! – A Motionplanner Library

- What is *MoveIt!* ?
  - Motionplanner library developed in Willowgarage
  - Software framework containing
    - State of the art motion planners
    - Kinemtatics solver
    - Collision Checking
    - Control and Navigation

- Why we need *MoveIt!* ?
  - Developing new higher-level capabililties from scratch is time consuming
  - To build mobile manipulation applications



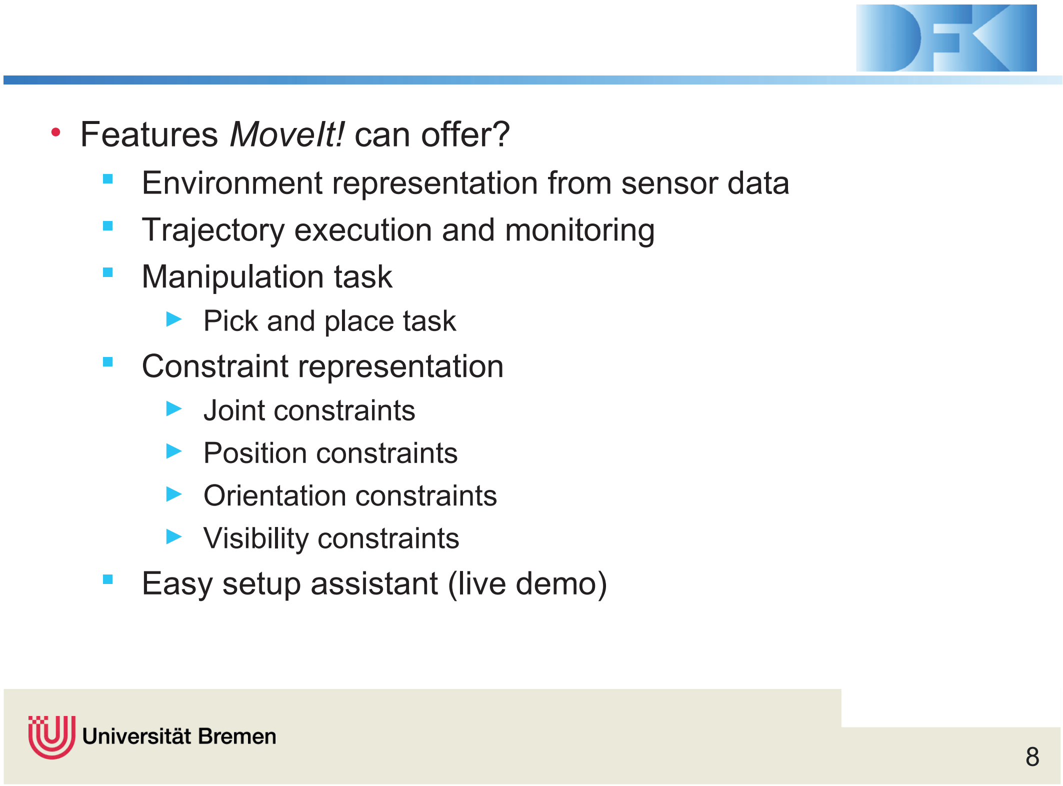
- Features *MoveIt!* can offer?
  - State of the art Sampling-based planner (OMPL)
    - RRT, RRTConnet, LazyPPRT, PRM, KPIECE, LBKPIECE, BKPIECE, etc .,
    - Benchmarking custom planner
  - Search-based planner (SBPL)
  - Kinematic analysis

Kinematics Solver
KDL
IkFast

Motion Planner
OMPL
SBPL

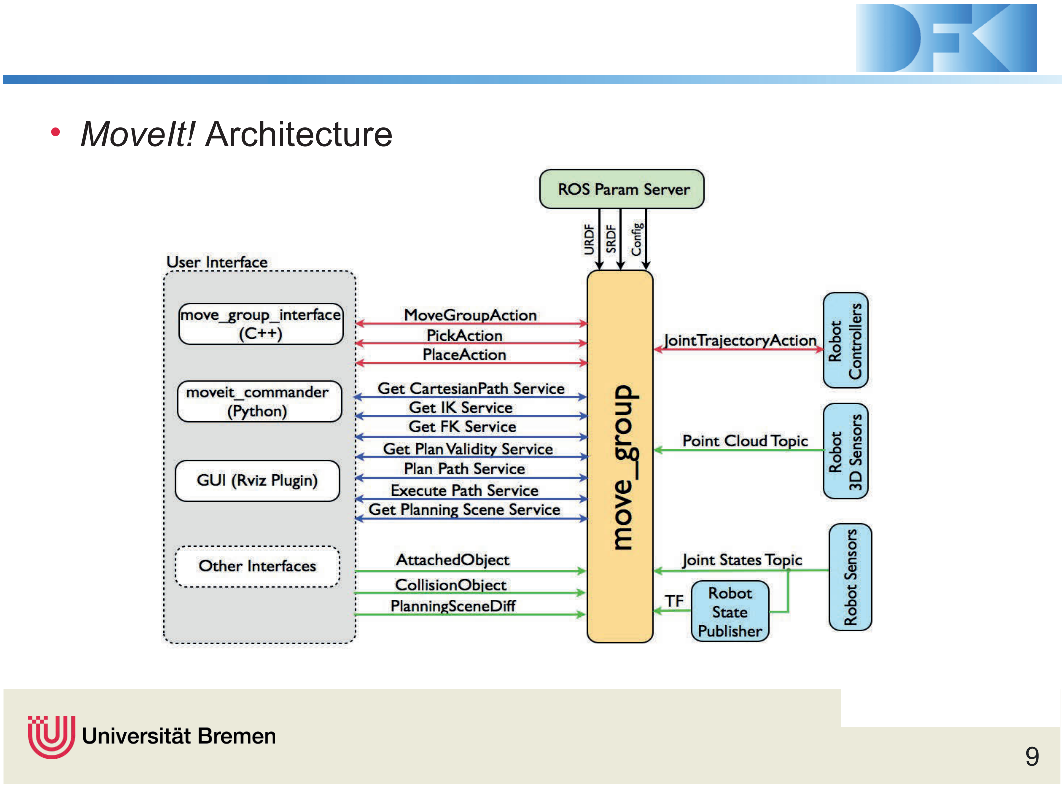
Collision Checking
FCL
PQP

Universität Bremen



- Features *MoveIt!* can offer?
  - Environment representation from sensor data
  - Trajectory execution and monitoring
  - Manipulation task
    - Pick and place task
  - Constraint representation
    - Joint constraints
    - Position constraints
    - Orientation constraints
    - Visibility constraints
  - Easy setup assistant (live demo)

Universität Bremen



- *MoveIt!* Architecture

ROS Param Server

URDF
SRDF
Config

move_group

User Interface

move_group_interface (C++)

moveit_commander (Python)

GUI (Rviz Plugin)

Other Interfaces

MoveGroupAction
PickAction
PlaceAction

Get CartesianPath Service
Get IK Service
Get FK Service
Get Plan Validity Service
Plan Path Service
Execute Path Service
Get Planning Scene Service

AttachedObject
CollisionObject
PlanningSceneDiff

JointTrajectoryAction

Robot Controllers

Point Cloud Topic

Robot 3D Sensors

Joint States Topic

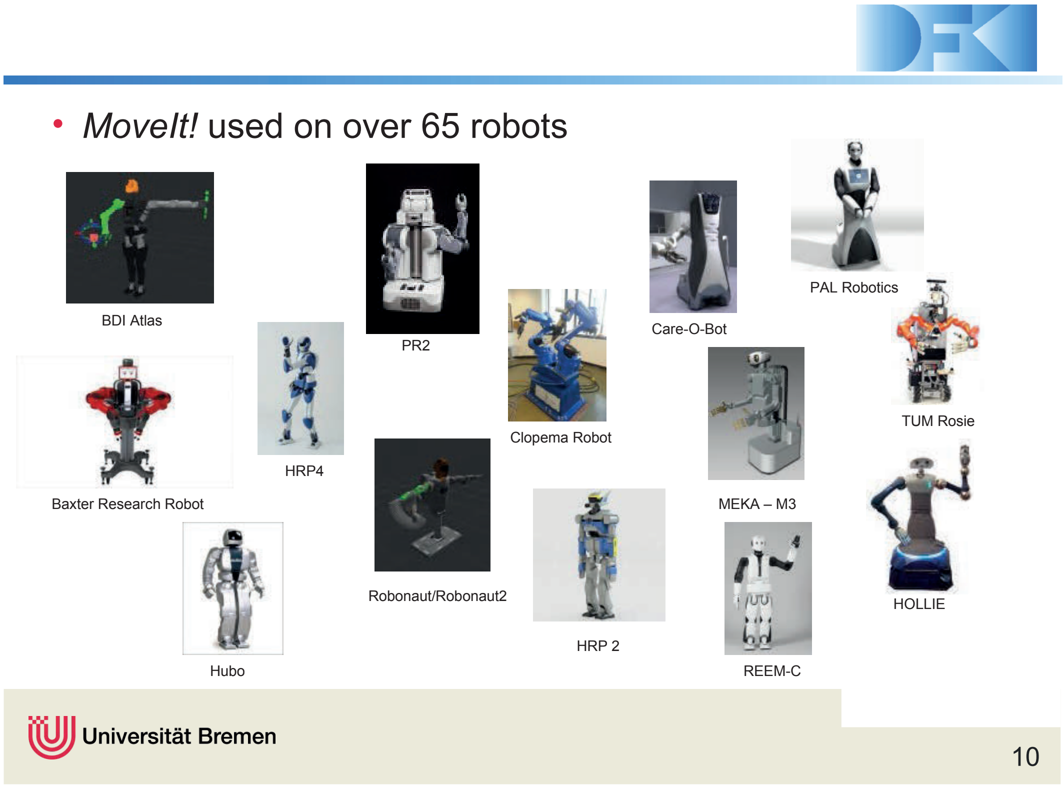
Robot Sensors

TF

Robot State Publisher

Universität Bremen

9



- *MoveIt!* used on over 65 robots

BDI Atlas

PR2

Care-O-Bot

PAL Robotics

HRP4

Clopema Robot

TUM Rosie

Baxter Research Robot

MEKA – M3

Robonaut/Robonaut2

HOLLIE

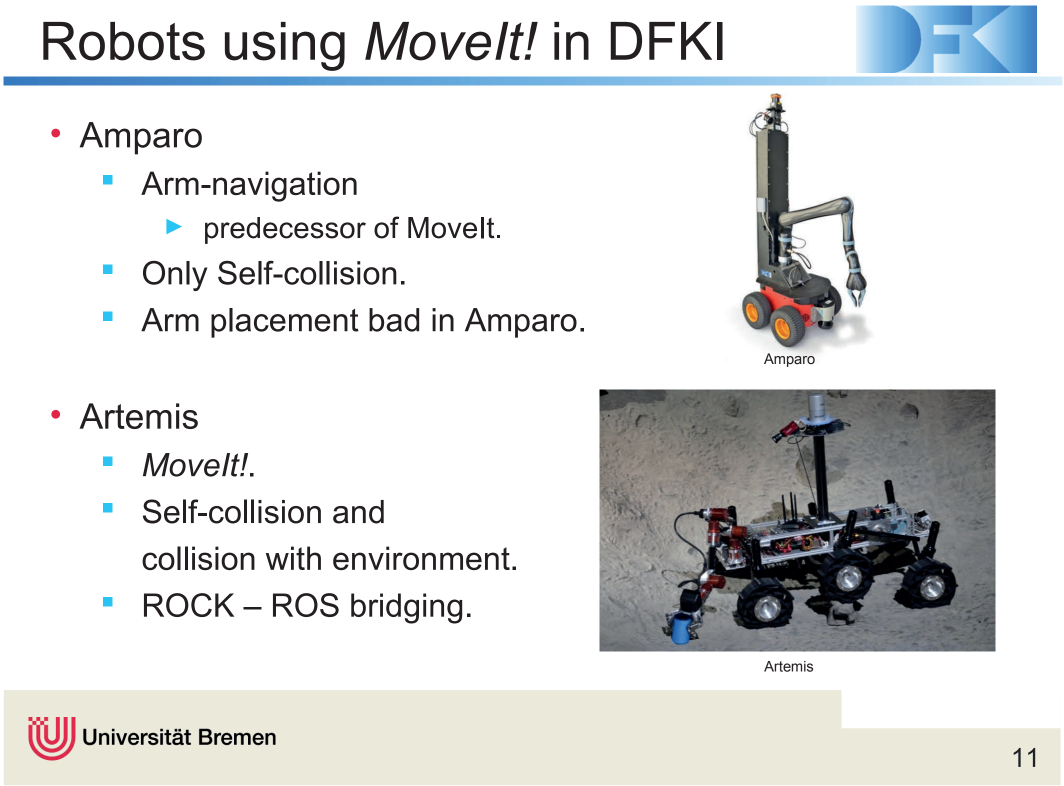
HRP 2

Hubo

REEM-C

Universität Bremen

10



Robots using *MoveIt!* in DFKI

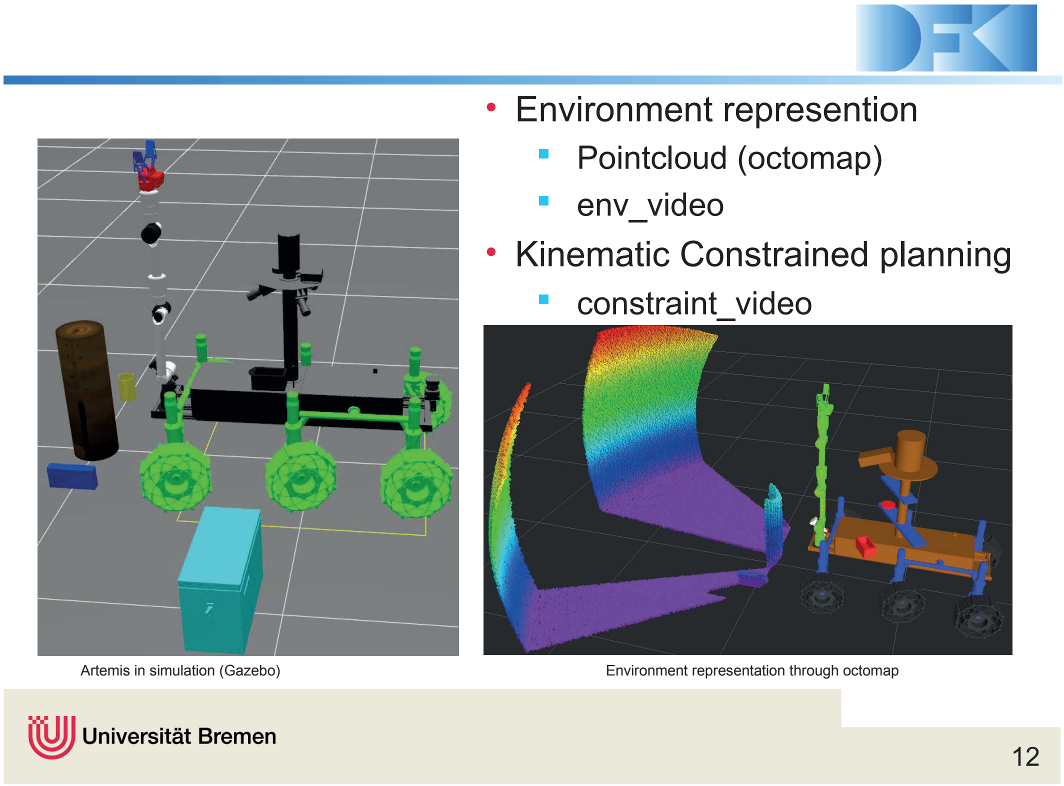
- Amparo
  - Arm-navigation
    - predecessor of MoveIt.
  - Only Self-collision.
  - Arm placement bad in Amparo.

Amparo

- Artemis
  - *MoveIt!*.
  - Self-collision and collision with environment.
  - ROCK – ROS bridging.

Artemis

Universität Bremen



- Environment represention
  - Pointcloud (octomap)
  - env_video
- Kinematic Constrained planning
  - constraint_video

Artemis in simulation (Gazebo)

Environment representation through octomap

Universität Bremen

LiveDemo

- Easy to setup

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

3.4 ‘Motion Planning Library for Robotic Application’ (MC-T-04)

Behnam Asadi(1)

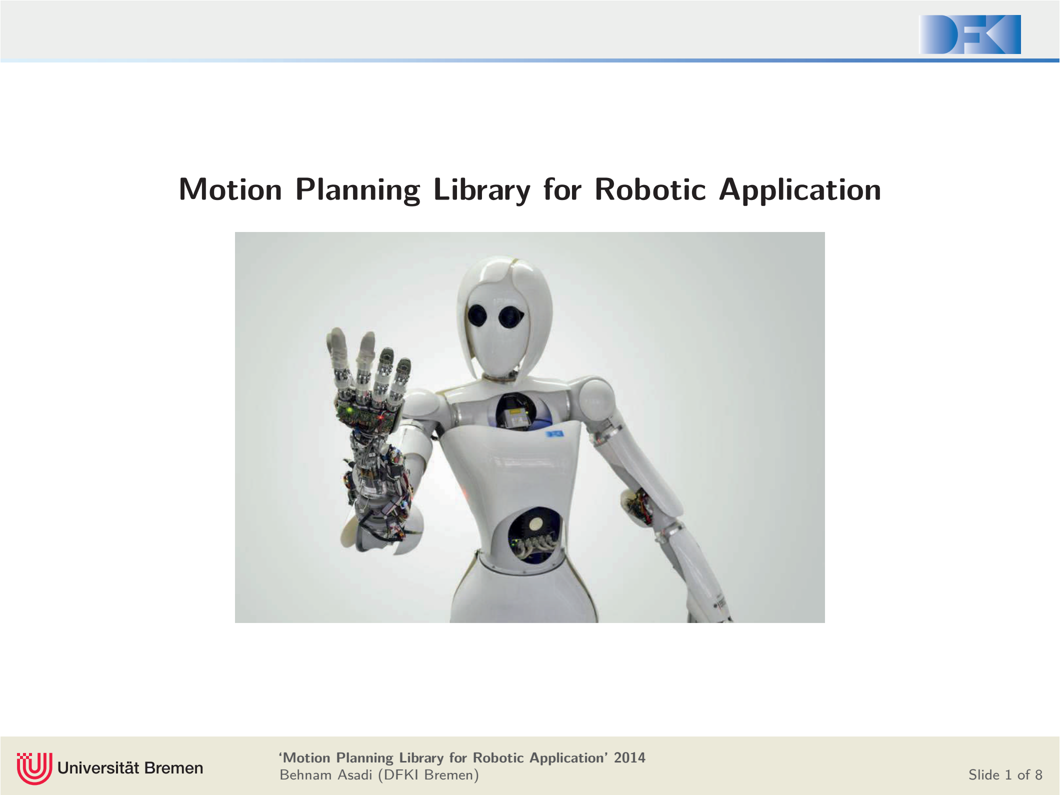
(1) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `behnam@informatik.uni-bremen.de`

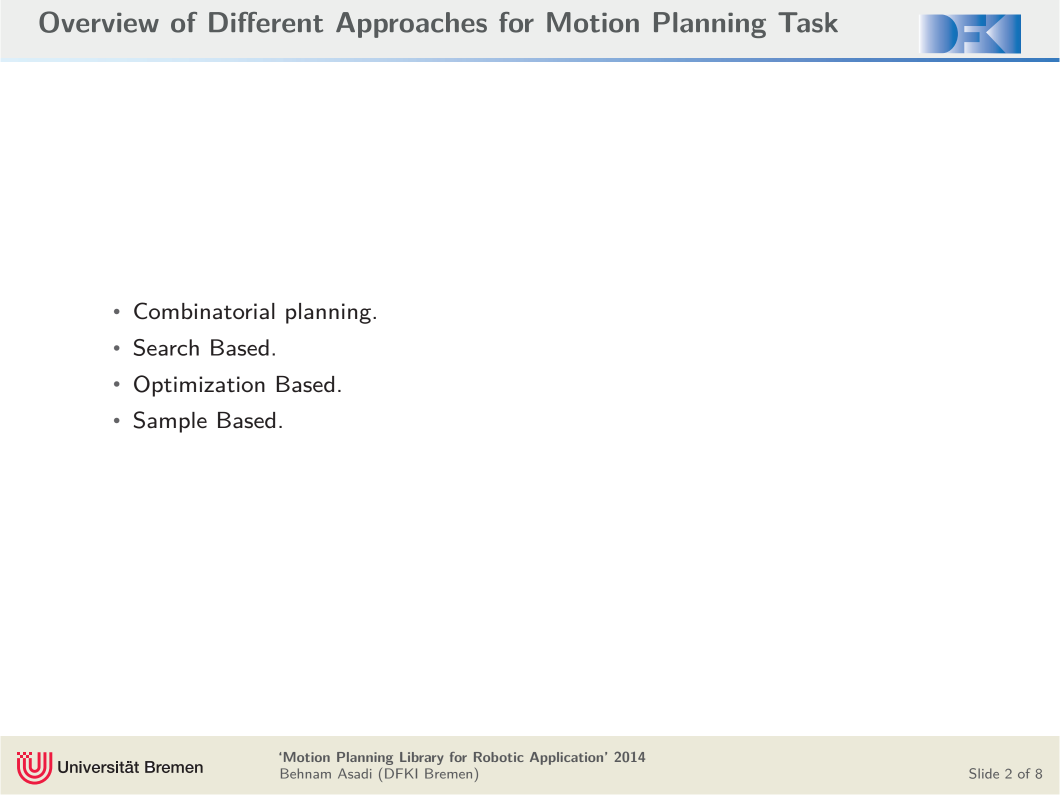
Abstract

A stand alone library for motion planning task, based on sample based motion planners, implemented in C++. Functionalities of the library:

- Computing analytical and numerical solutions for forward and inverse kinematics queries.
- Hierarchical Brute-force search for the robots with more than 6 DOF for minimum movement.
- Self collision and environment collision checker.
- Replanning for dynamic environment.
- Dual Arm manipulation.

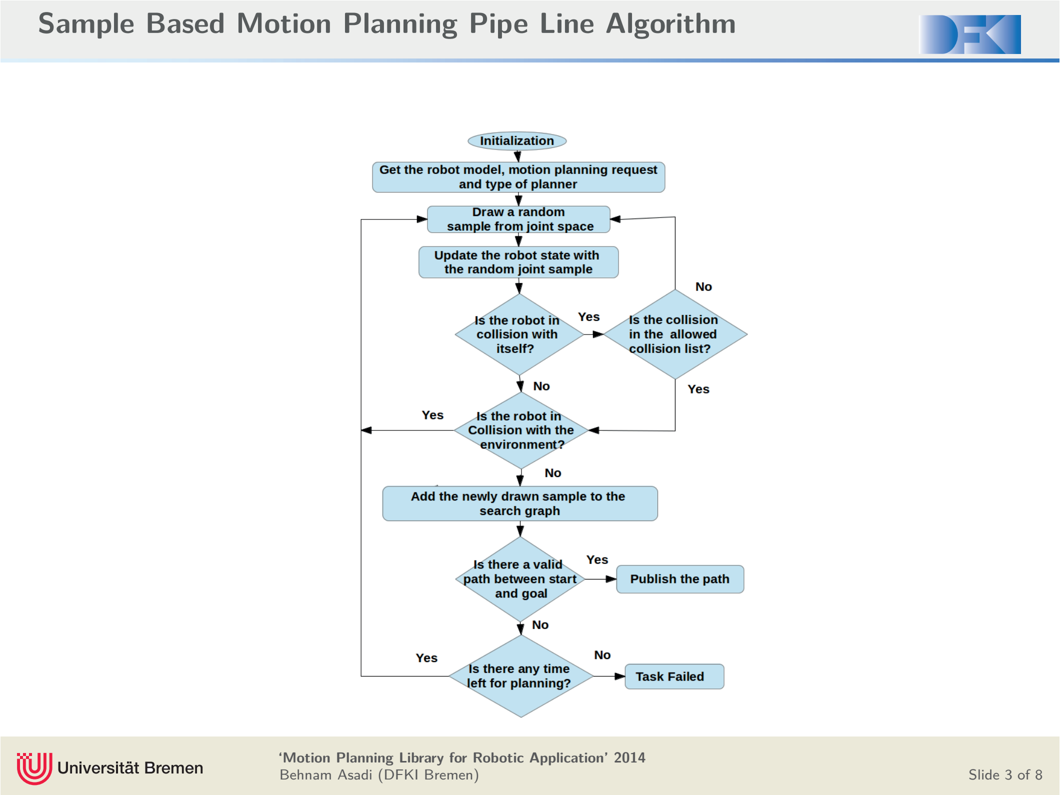


Motion Planning Library for Robotic Application



Overview of Different Approaches for Motion Planning Task

- Combinatorial planning.
- Search Based.
- Optimization Based.
- Sample Based.



Sample Based Motion Planning Pipe Line Algorithm

Initialization

Get the robot model, motion planning request and type of planner

Draw a random sample from joint space

Update the robot state with the random joint sample

Is the robot in collision with itself?

Yes

Is the collision in the allowed collision list?

No

Yes

No

Yes

Is the robot in Collision with the environment?

No

Add the newly drawn sample to the search graph

Is there a valid path between start and goal

Yes

Publish the path

No

Yes

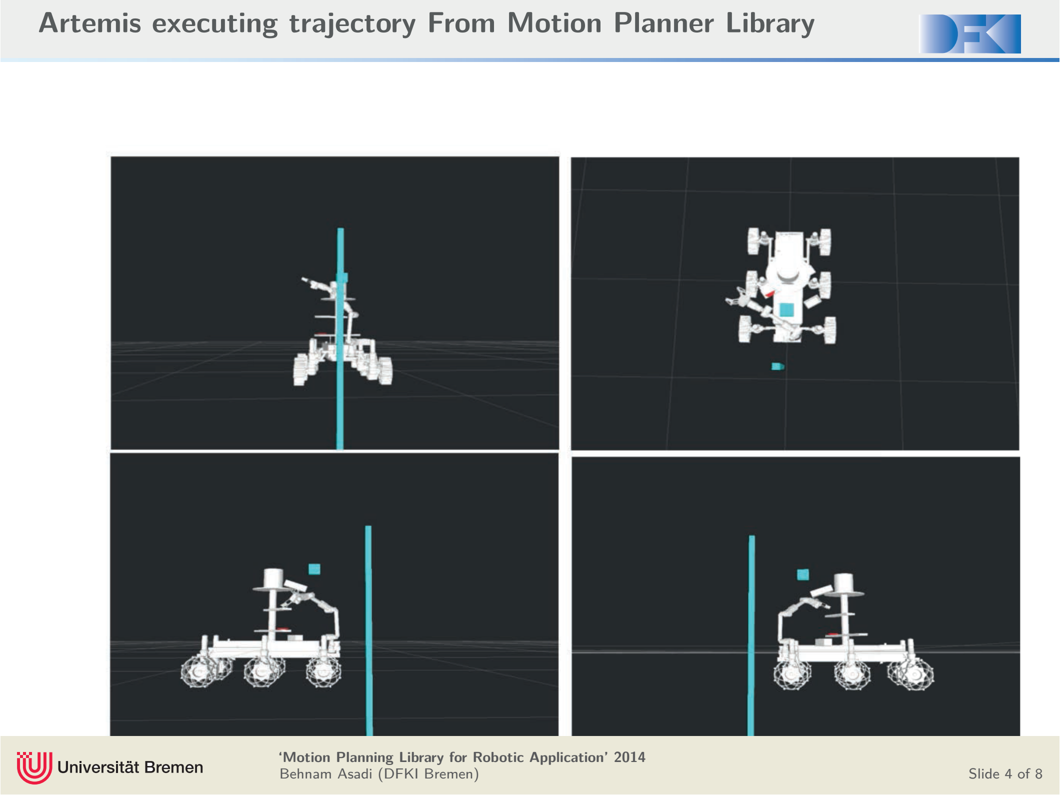
Is there any time left for planning?

No

Task Failed

Universität Bremen

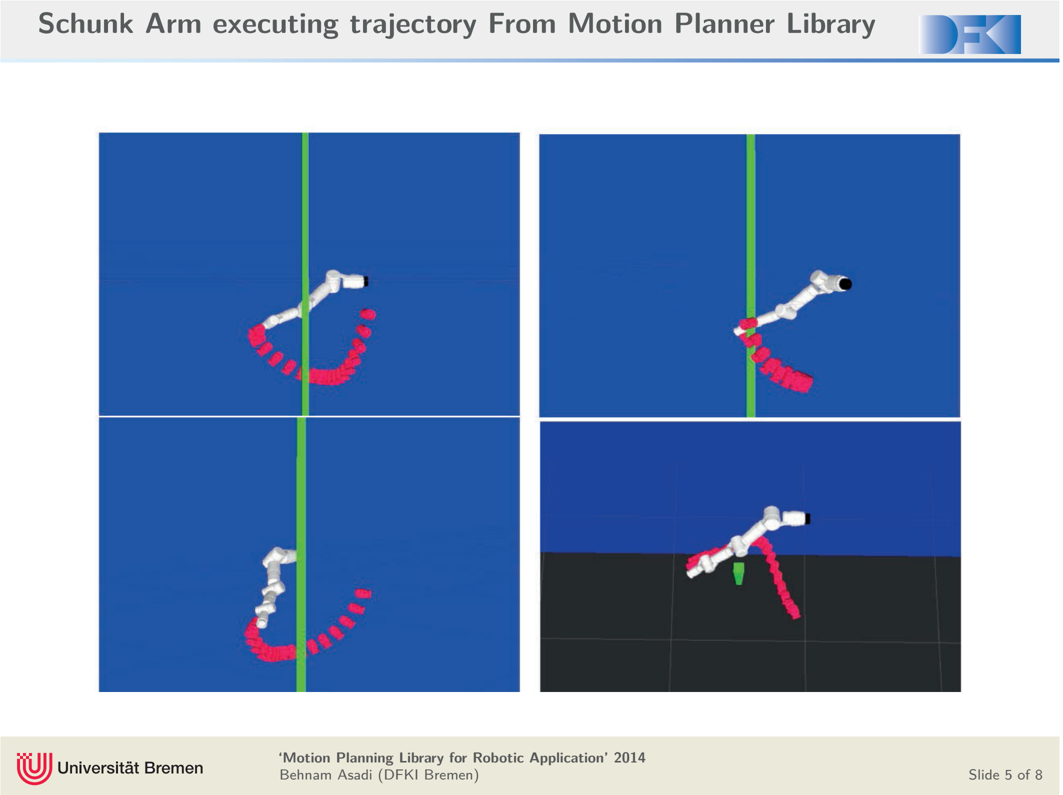
'Motion Planning Library for Robotic Application' 2014
Behnam Asadi (DFKI Bremen)



Artemis executing trajectory From Motion Planner Library

Universität Bremen

'Motion Planning Library for Robotic Application' 2014
Behnam Asadi (DFKI Bremen)

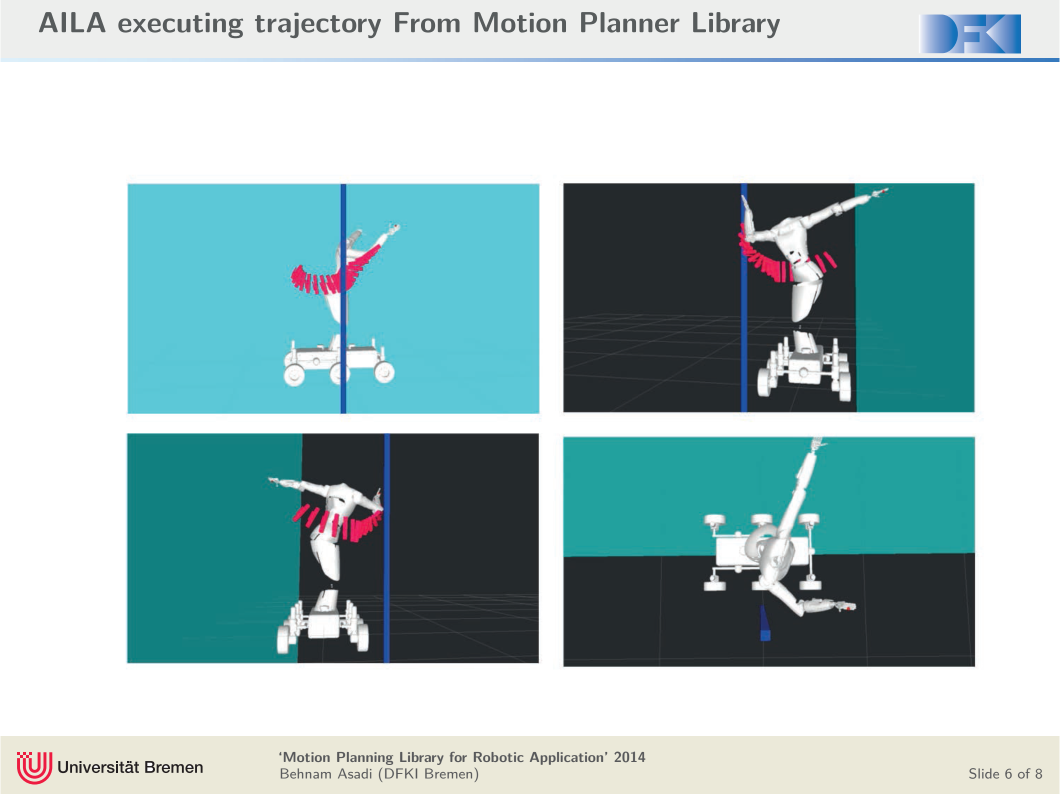


Schunk Arm executing trajectory From Motion Planner Library

Universität Bremen

'Motion Planning Library for Robotic Application' 2014
Behnam Asadi (DFKI Bremen)

Slide 5 of 8

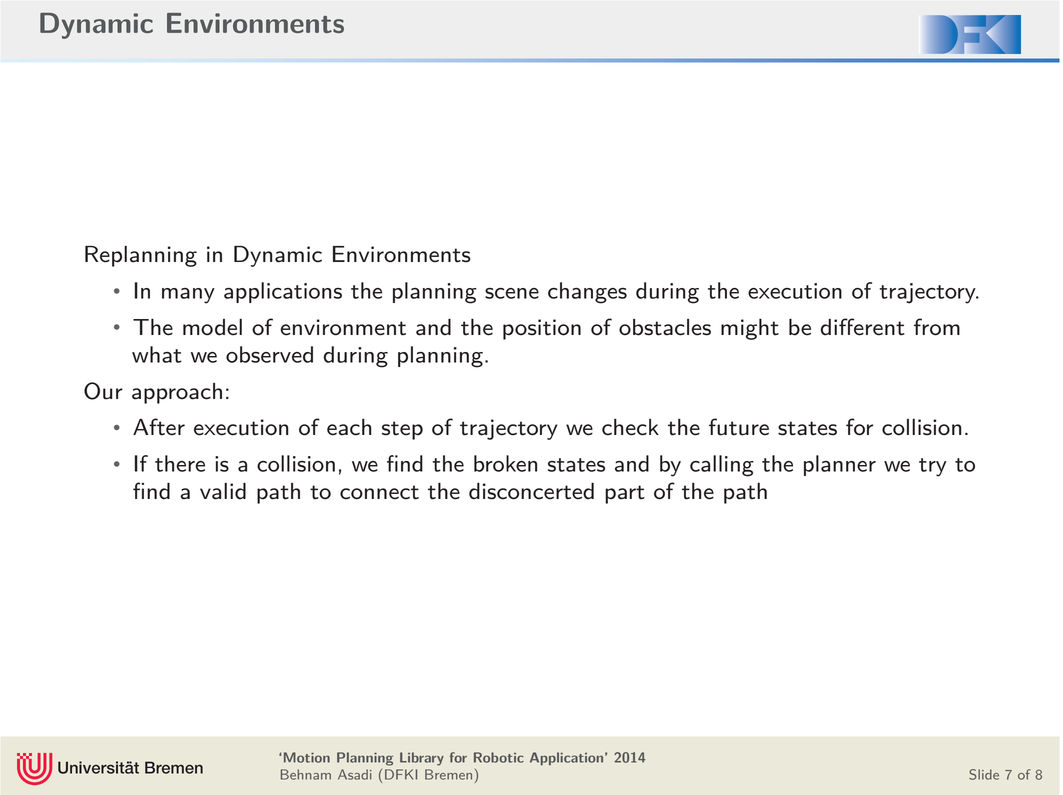


AILA executing trajectory From Motion Planner Library

Universität Bremen

'Motion Planning Library for Robotic Application' 2014
Behnam Asadi (DFKI Bremen)

Slide 6 of 8



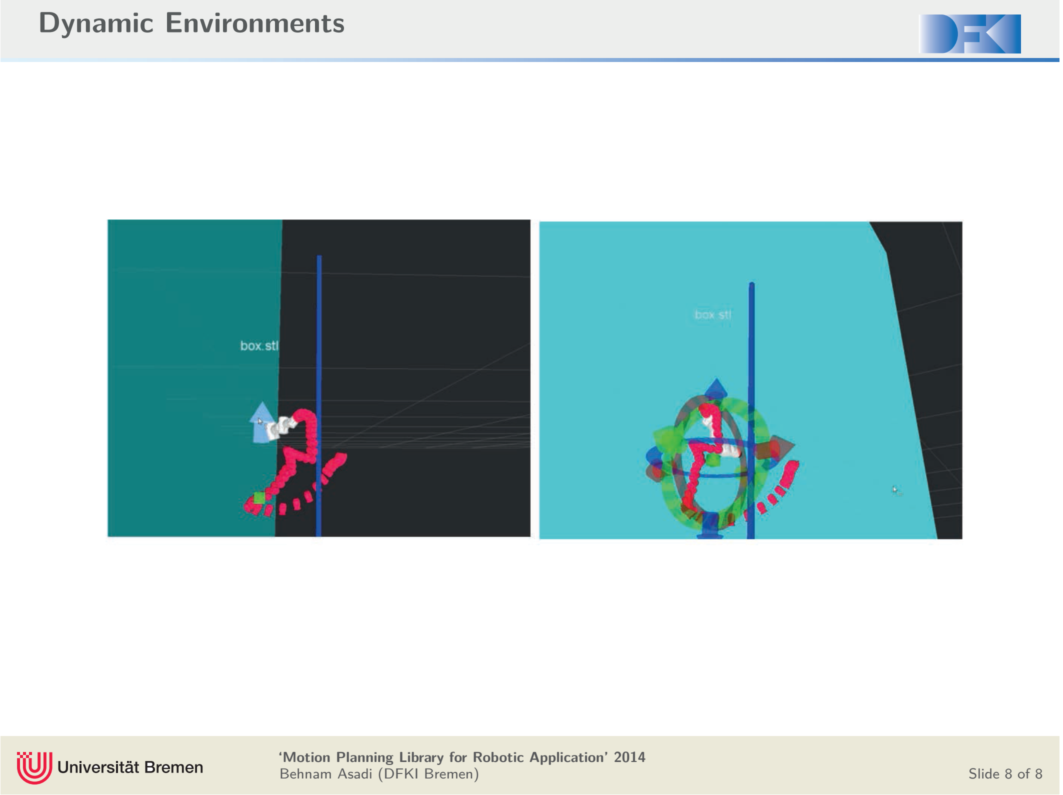
Dynamic Environments

Replanning in Dynamic Environments

- In many applications the planning scene changes during the execution of trajectory.
- The model of environment and the position of obstacles might be different from what we observed during planning.

Our approach:

- After execution of each step of trajectory we check the future states for collision.
- If there is a collision, we find the broken states and by calling the planner we try to find a valid path to connect the disconcerted part of the path



Dynamic Environments

3.5 ‘Online trajectory generation using the Reflexxes library’ (MC-T-05)

Benjamin Girault(1), Malte Wirkus(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `benjamin.girault@dfki.de`, `malte.wirkus@dfki.de`

Abstract

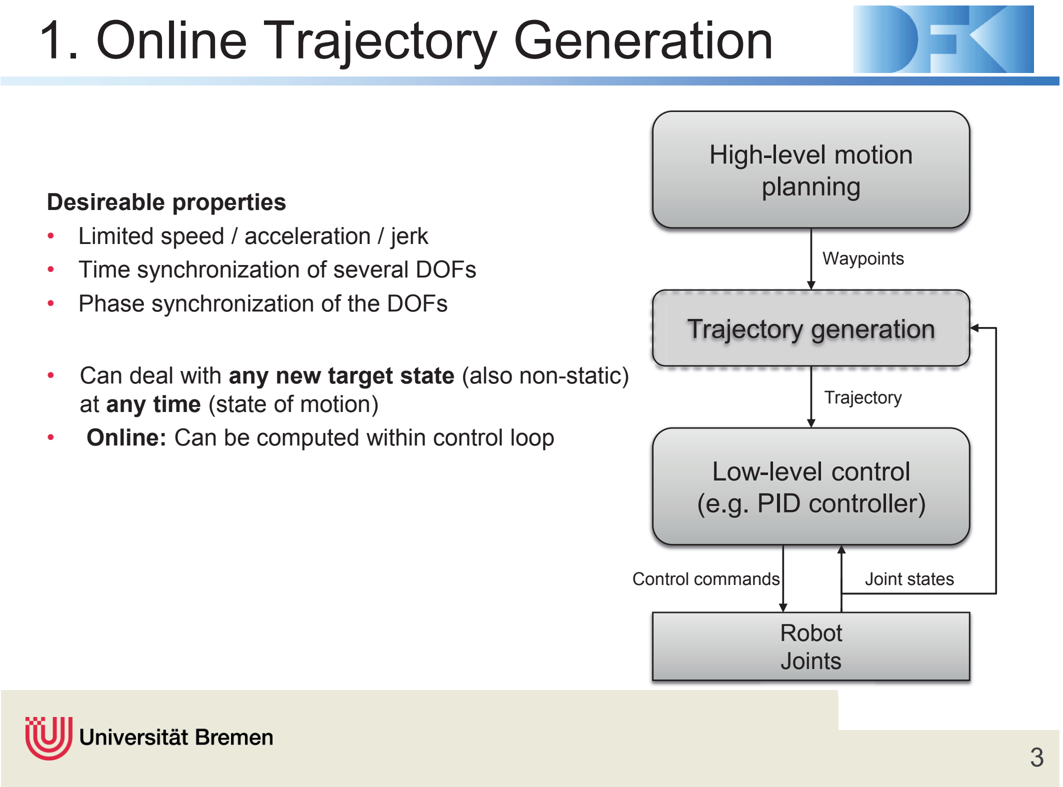
This talk presents the case of online trajectory generation and how the library Reflexxes (www.reflexxes.com) solved this problem. The first part explains why we need trajectory generation algorithms that can deal with arbitrary initial states, can be computed online and allows the joints of the robot to be time synchronized and reach a target position and speed. The second parts describes the library Reflexxes, its features and how it works. Finally, examples of implementation at the DFKI are presented.

Online Trajectory generation using the Reflexxes Library

Project Day
AG Manipulation & Control
19th June 2014
Benjamin Girault, Malte Wirkus, DFKI RIC

Content

1. Online Trajectory Generation
   - What is it?
   - What do we expect from it?
2. Reflexxes Library
   - Where does it come from?
   - What does it do?
   - How does it work?
3. Using the Library



1. Online Trajectory Generation

Desireable properties

- Limited speed / acceleration / jerk
- Time synchronization of several DOFs
- Phase synchronization of the DOFs

- Can deal with **any new target state** (also non-static) at **any time** (state of motion)
- **Online:** Can be computed within control loop

High-level motion planning

Waypoints

Trajectory generation

Trajectory

Low-level control (e.g. PID controller)

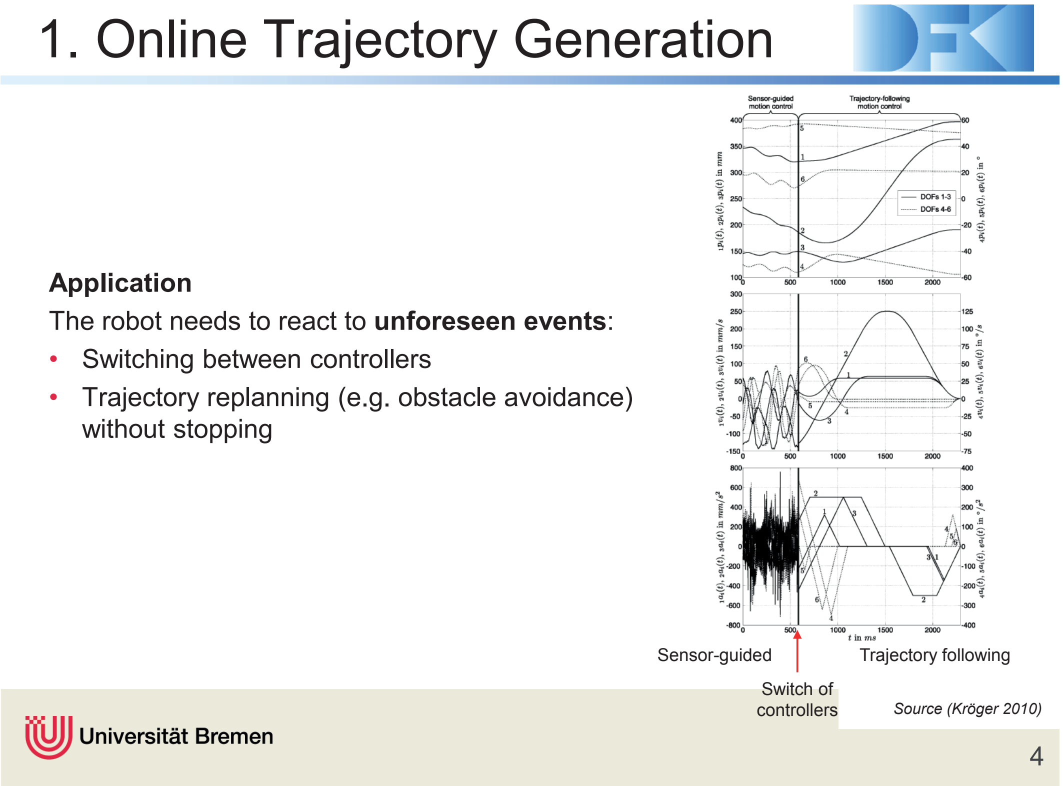
Control commands

Joint states

Robot Joints

Universität Bremen

3



1. Online Trajectory Generation

Application

The robot needs to react to **unforeseen events**:

- Switching between controllers
- Trajectory replanning (e.g. obstacle avoidance) without stopping

Sensor-guided

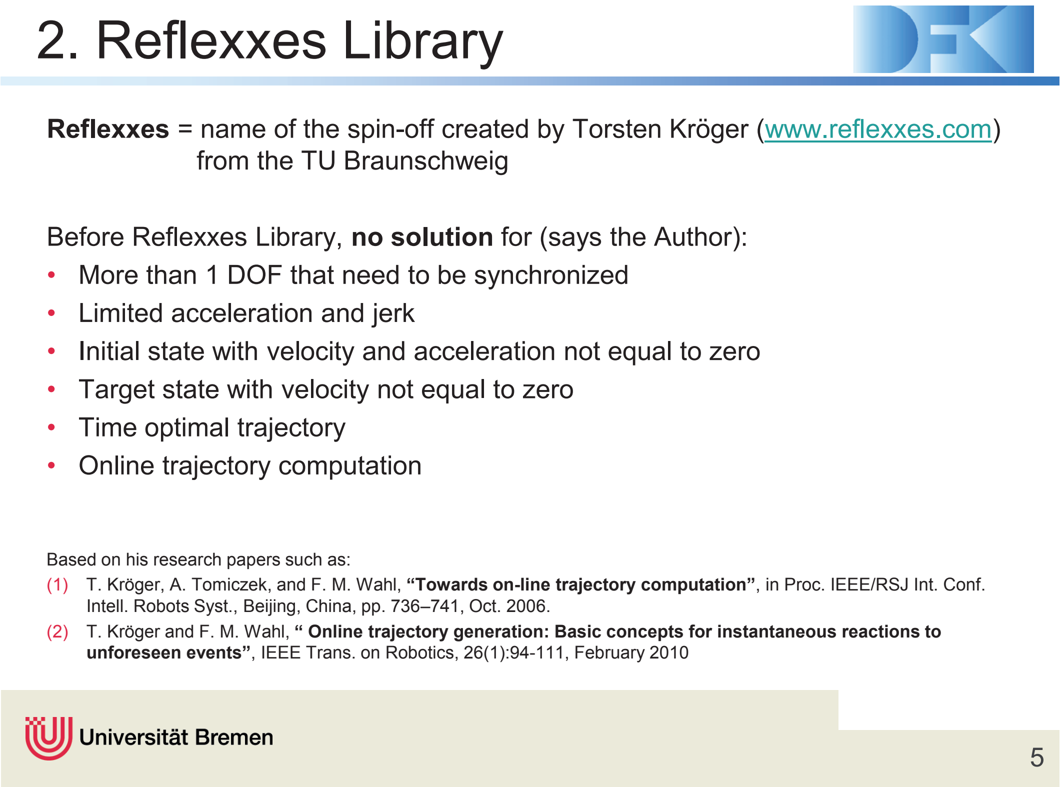
Trajectory following

Switch of controllers

Source (Kröger 2010)

Universität Bremen

4



2. Reflexxes Library

Reflexxes = name of the spin-off created by Torsten Kröger (www.reflexxes.com)
from the TU Braunschweig

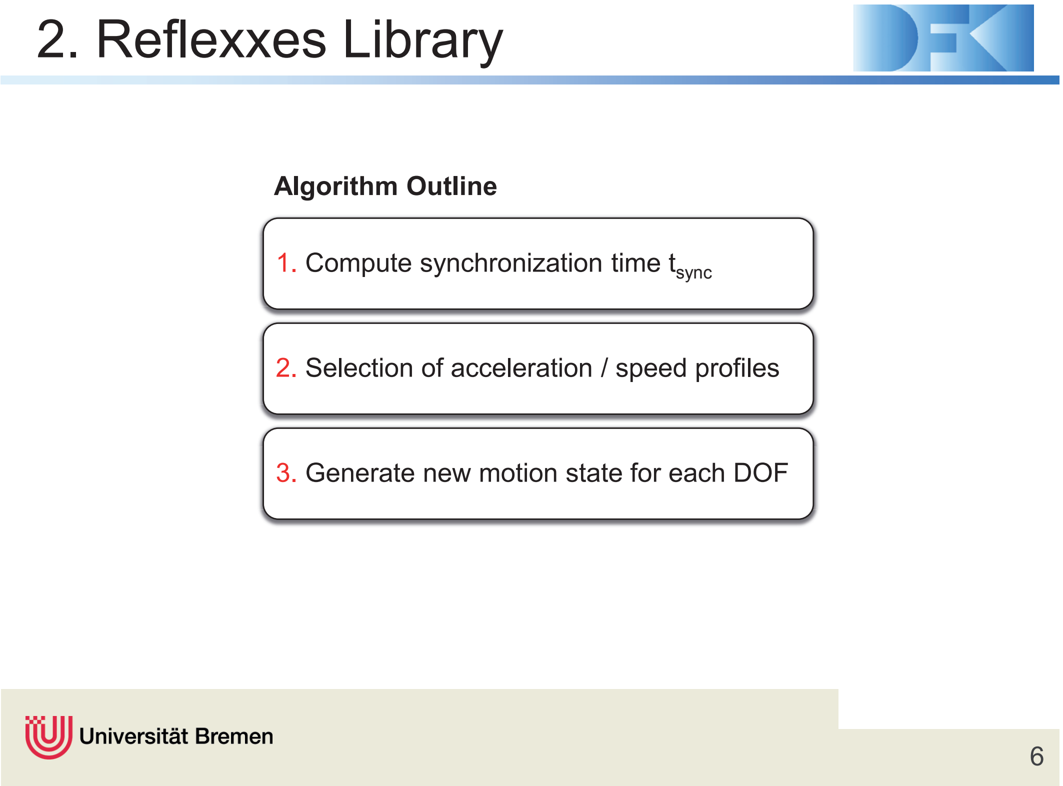
Before Reflexxes Library, **no solution** for (says the Author):

- More than 1 DOF that need to be synchronized
- Limited acceleration and jerk
- Initial state with velocity and acceleration not equal to zero
- Target state with velocity not equal to zero
- Time optimal trajectory
- Online trajectory computation

Based on his research papers such as:

(1) T. Kröger, A. Tomiczek, and F. M. Wahl, **"Towards on-line trajectory computation"**, in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Beijing, China, pp. 736–741, Oct. 2006.

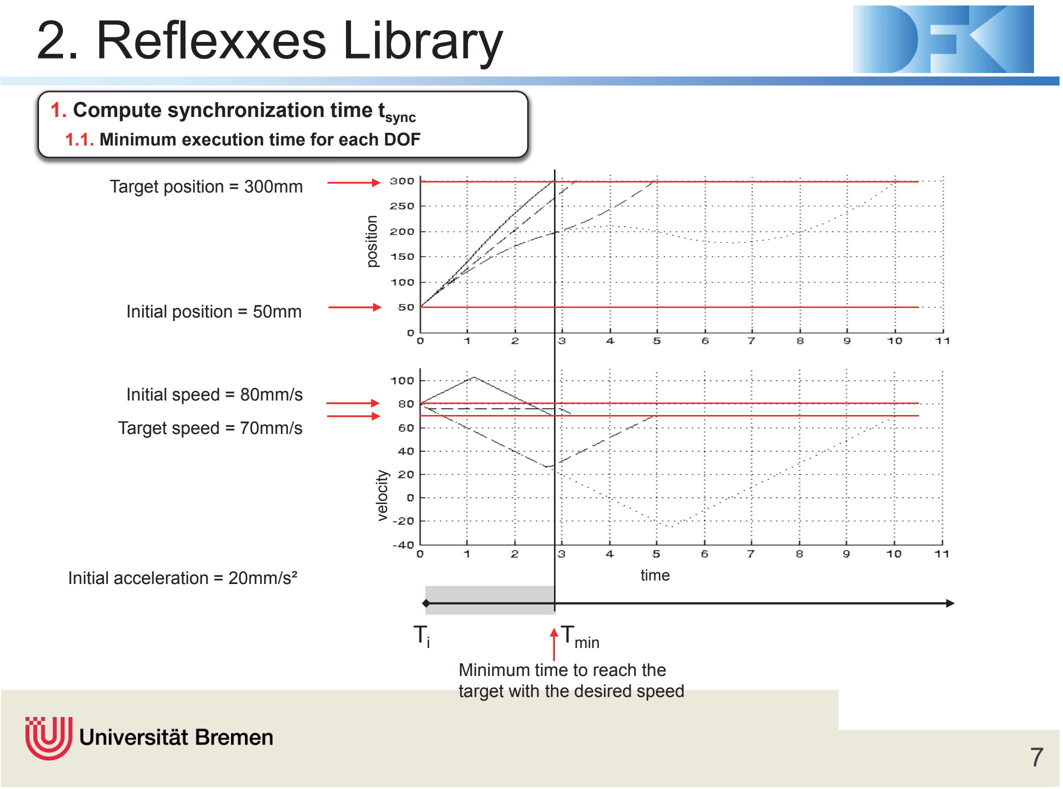
(2) T. Kröger and F. M. Wahl, **" Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events"**, IEEE Trans. on Robotics, 26(1):94-111, February 2010



2. Reflexxes Library

Algorithm Outline

1. Compute synchronization time t_{sync}
2. Selection of acceleration / speed profiles
3. Generate new motion state for each DOF



2. Reflexxes Library

1. Compute synchronization time t_{sync}

1.1. Minimum execution time for each DOF

Target position = 300mm

Initial position = 50mm

Initial speed = 80mm/s

Target speed = 70mm/s

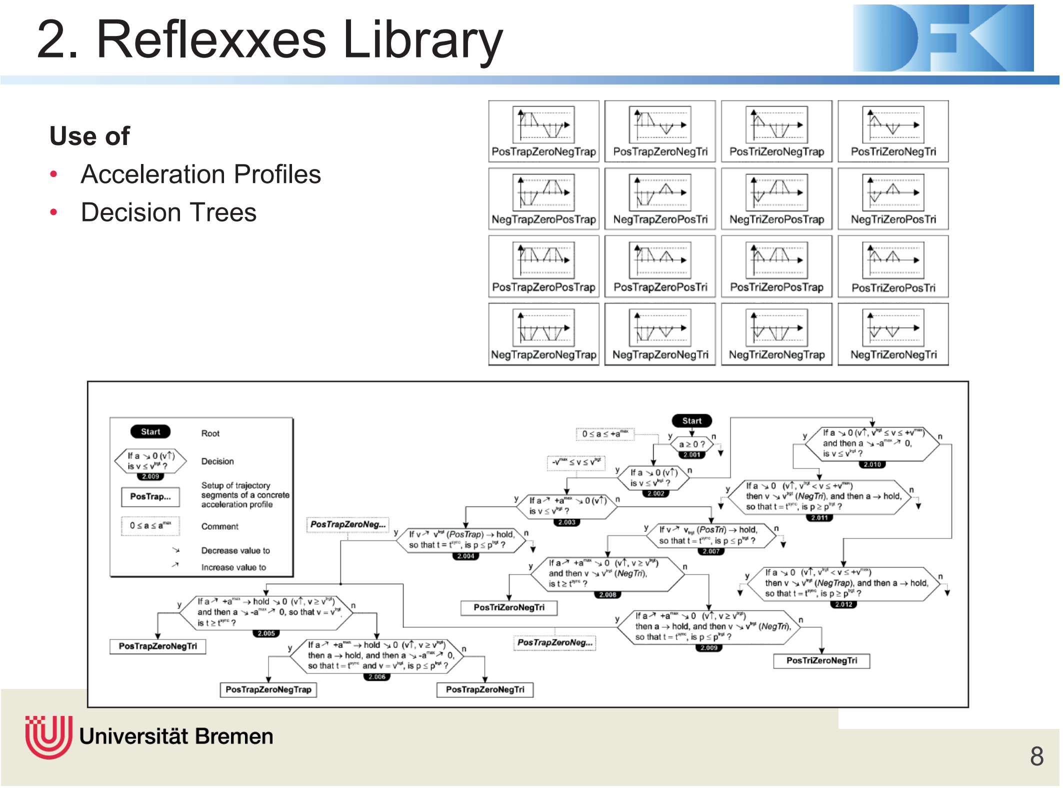
Initial acceleration = 20mm/s²

T_i

T_{min}

Minimum time to reach the target with the desired speed

Universität Bremen

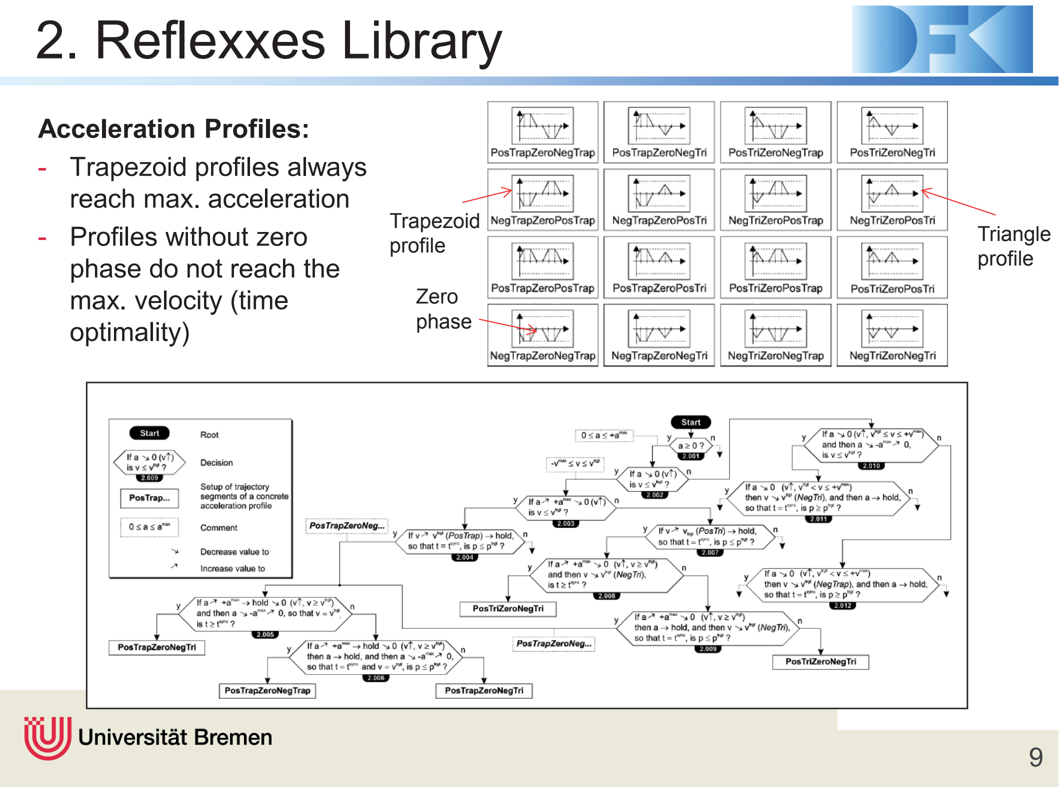


2. Reflexxes Library

Use of

- Acceleration Profiles
- Decision Trees

Universität Bremen

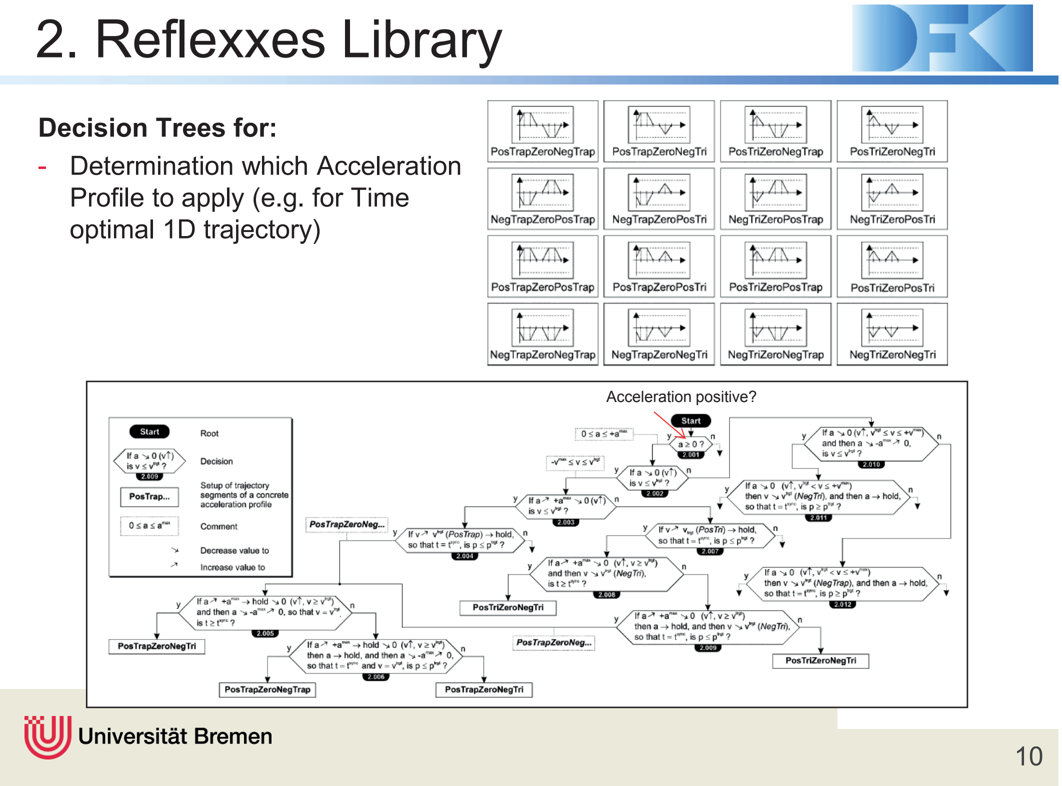


2. Reflexxes Library

Acceleration Profiles:

- Trapezoid profiles always reach max. acceleration
- Profiles without zero phase do not reach the max. velocity (time optimality)

Universität Bremen

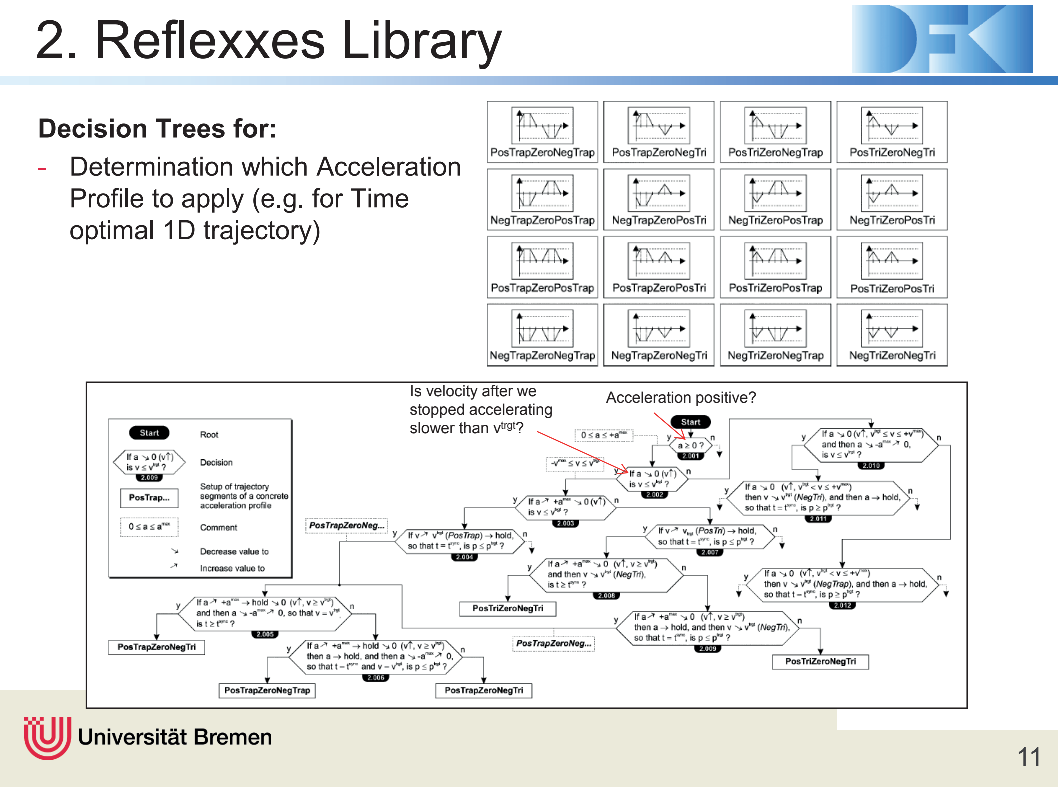


2. Reflexxes Library

Decision Trees for:

- Determination which Acceleration Profile to apply (e.g. for Time optimal 1D trajectory)

Universität Bremen



2. Reflexxes Library

Decision Trees for:

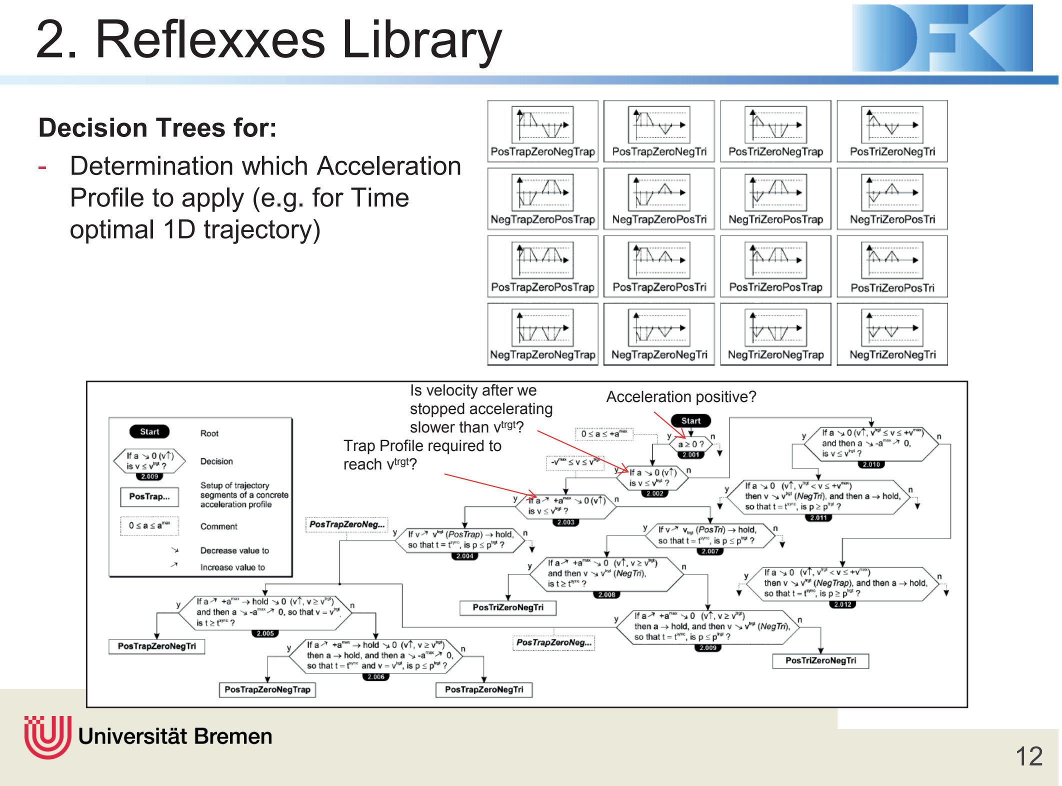
- Determination which Acceleration Profile to apply (e.g. for Time optimal 1D trajectory)

PosTrapZeroNegTrap	PosTrapZeroNegTri	PosTriZeroNegTrap	PosTriZeroNegTri
NegTrapZeroPosTrap	NegTrapZeroPosTri	NegTriZeroPosTrap	NegTriZeroPosTri
PosTrapZeroPosTrap	PosTrapZeroPosTri	PosTriZeroPosTrap	PosTriZeroPosTri
NegTrapZeroNegTrap	NegTrapZeroNegTri	NegTriZeroNegTrap	NegTriZeroNegTri

Is velocity after we stopped accelerating slower than v^{trgt}?

Acceleration positive?

Universität Bremen



2. Reflexxes Library

Decision Trees for:

- Determination which Acceleration Profile to apply (e.g. for Time optimal 1D trajectory)

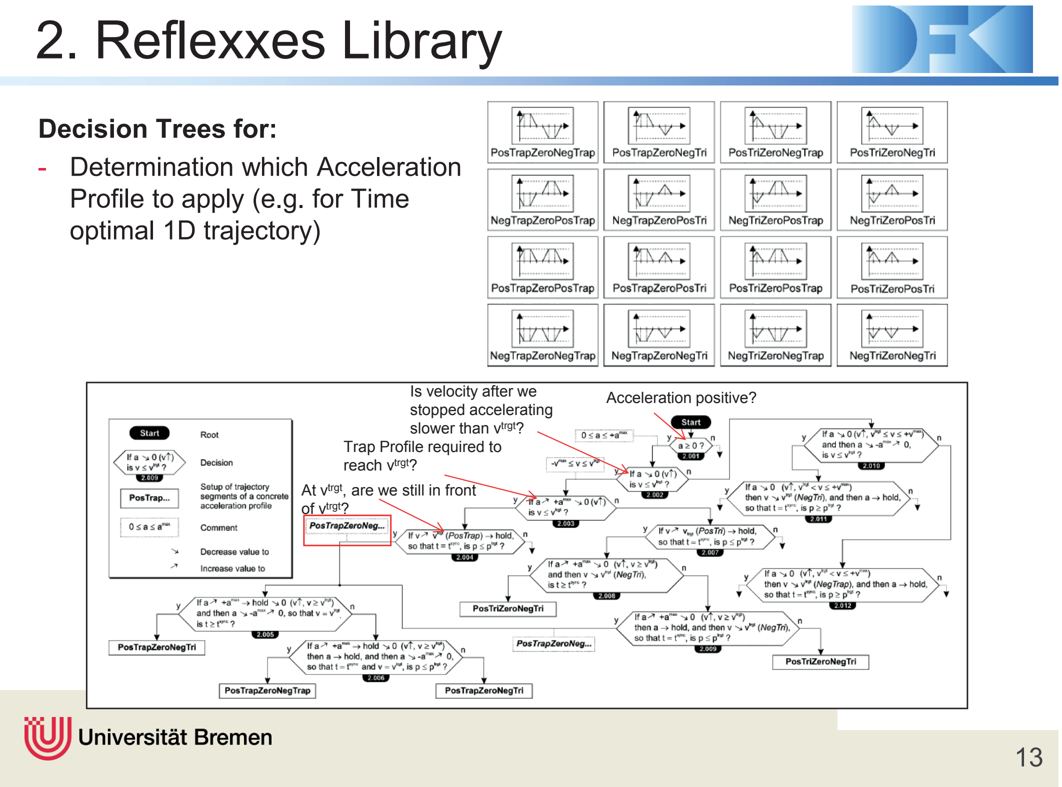
PosTrapZeroNegTrap	PosTrapZeroNegTri	PosTriZeroNegTrap	PosTriZeroNegTri
NegTrapZeroPosTrap	NegTrapZeroPosTri	NegTriZeroPosTrap	NegTriZeroPosTri
PosTrapZeroPosTrap	PosTrapZeroPosTri	PosTriZeroPosTrap	PosTriZeroPosTri
NegTrapZeroNegTrap	NegTrapZeroNegTri	NegTriZeroNegTrap	NegTriZeroNegTri

Is velocity after we stopped accelerating slower than v^{trgt}?

Acceleration positive?

Trap Profile required to reach v^{trgt}?

Universität Bremen



2. Reflexxes Library

Decision Trees for:

- Determination which Acceleration Profile to apply (e.g. for Time optimal 1D trajectory)

PosTrapZeroNegTrap
PosTrapZeroNegTri
PosTriZeroNegTrap
PosTriZeroNegTri
NegTrapZeroPosTrap
NegTrapZeroPosTri
NegTriZeroPosTrap
NegTriZeroPosTri
PosTrapZeroPosTrap
PosTrapZeroPosTri
PosTriZeroPosTrap
PosTriZeroPosTri
NegTrapZeroNegTrap
NegTrapZeroNegTri
NegTriZeroNegTrap
NegTriZeroNegTri

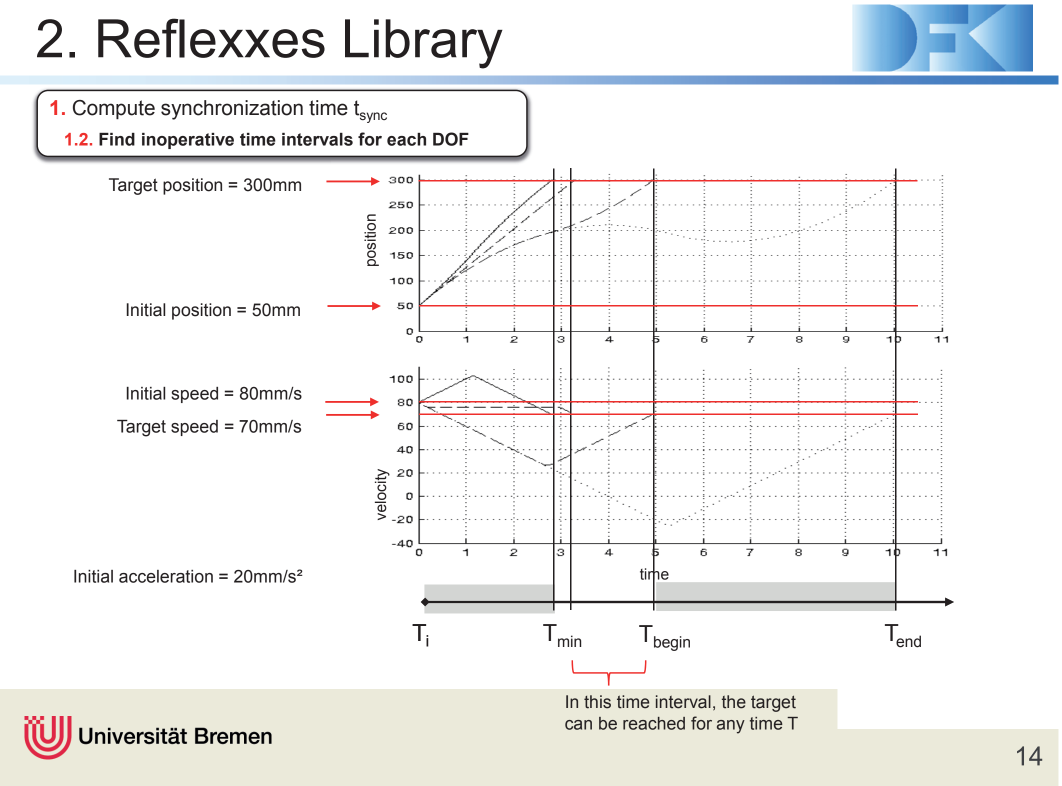
Is velocity after we stopped accelerating slower than v^{trgt}?

Acceleration positive?

Trap Profile required to reach v^{trgt}?

At v^{trgt}, are we still in front of v^{trgt}?

Universität Bremen



2. Reflexxes Library

1. Compute synchronization time t_{sync}

1.2. Find inoperative time intervals for each DOF

Target position = 300mm

Initial position = 50mm

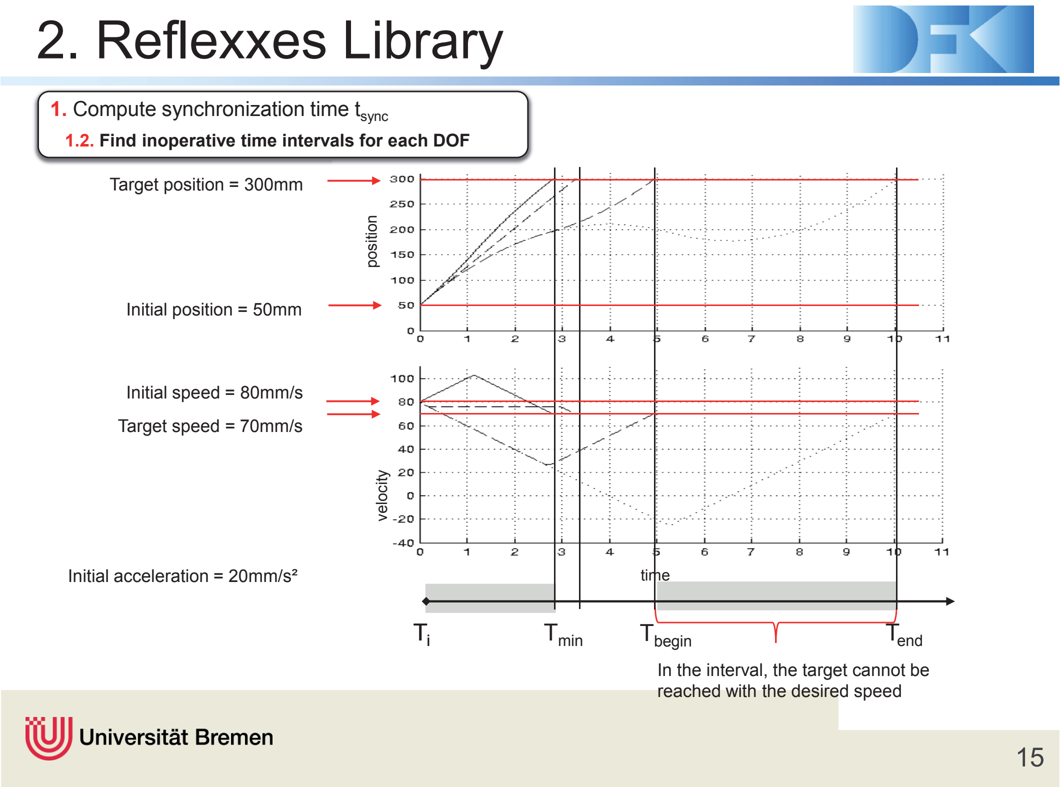
Initial speed = 80mm/s

Target speed = 70mm/s

Initial acceleration = 20mm/s²

In this time interval, the target can be reached for any time T

Universität Bremen



2. Reflexxes Library

1. Compute synchronization time t_{sync}

1.2. Find inoperative time intervals for each DOF

Target position = 300mm

Initial position = 50mm

Initial speed = 80mm/s

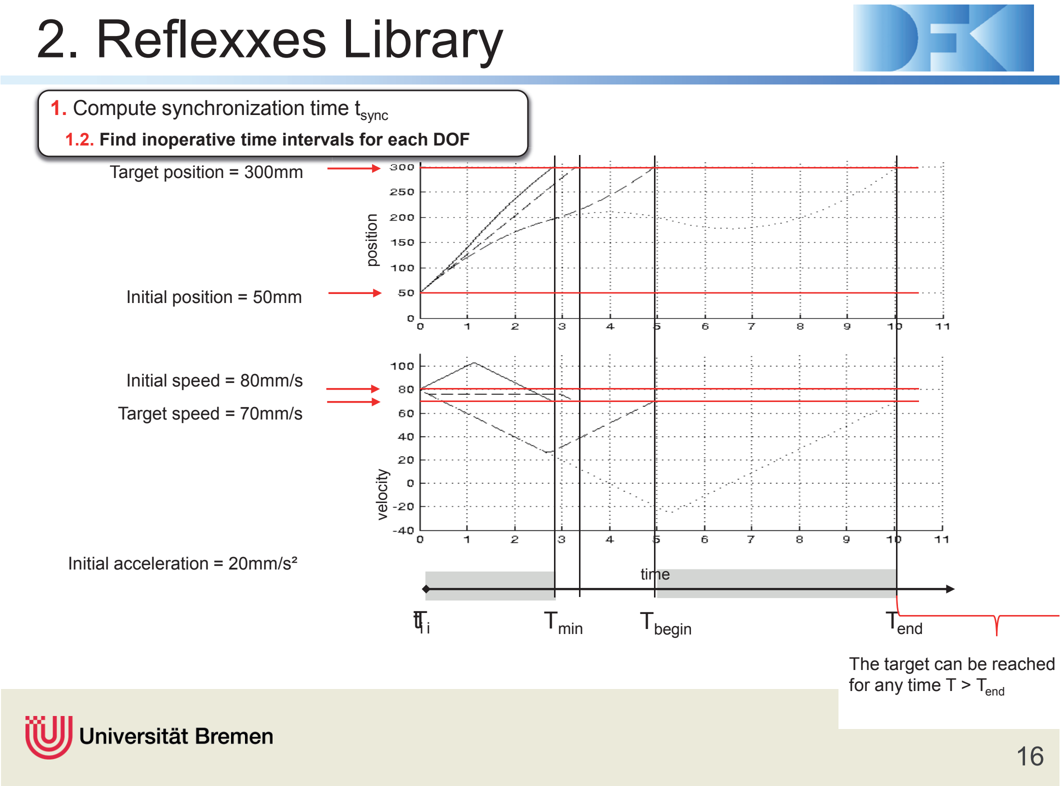
Target speed = 70mm/s

Initial acceleration = 20mm/s²

T_i T_{min} T_{begin} T_{end}

In the interval, the target cannot be reached with the desired speed

Universität Bremen



2. Reflexxes Library

1. Compute synchronization time t_{sync}

1.2. Find inoperative time intervals for each DOF

Target position = 300mm

Initial position = 50mm

Initial speed = 80mm/s

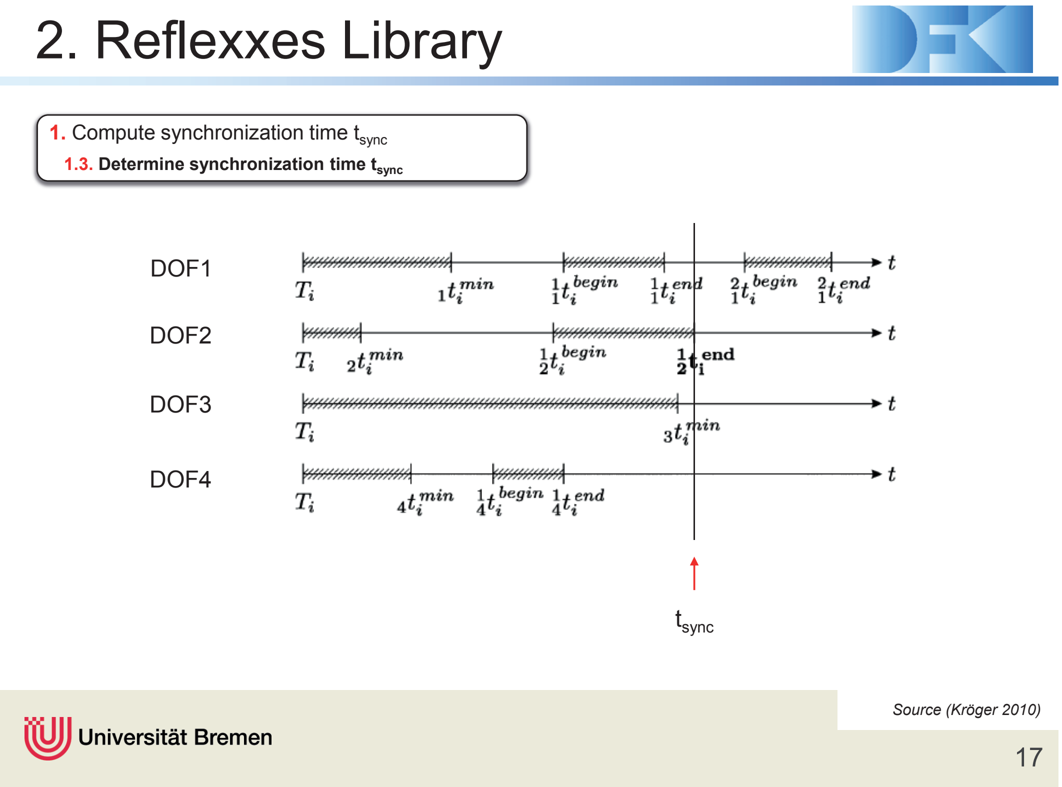
Target speed = 70mm/s

Initial acceleration = 20mm/s²

T_i T_{min} T_{begin} T_{end}

The target can be reached for any time T > T_{end}

Universität Bremen



2. Reflexxes Library

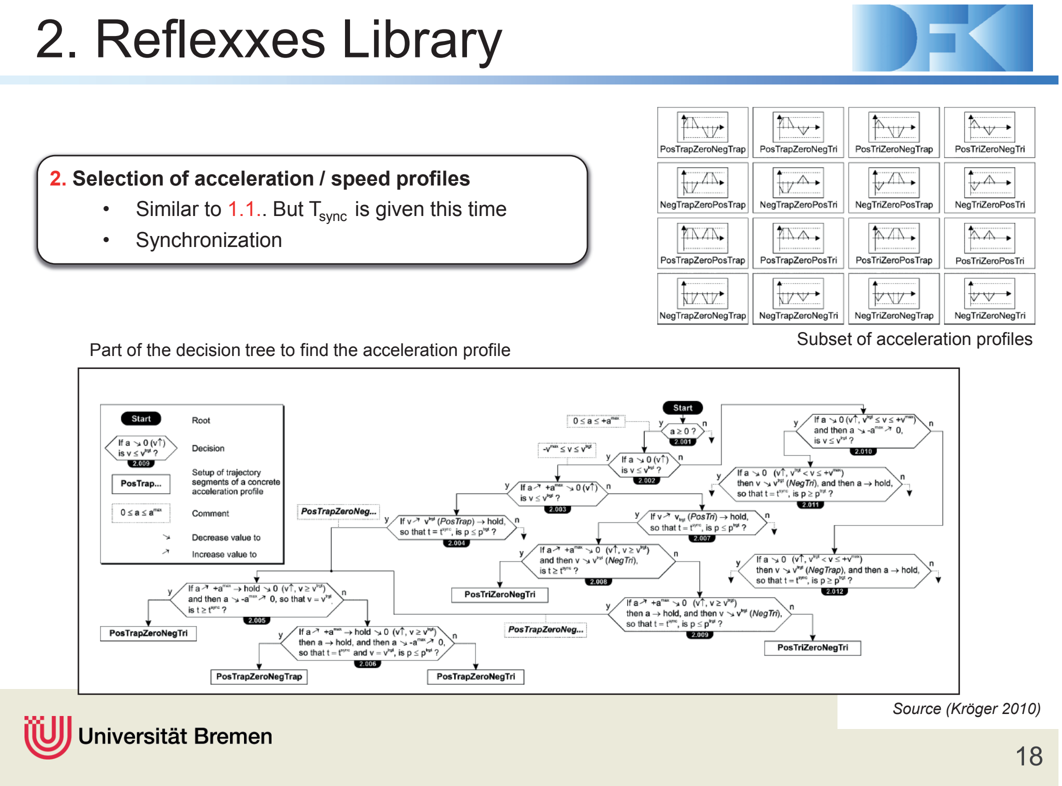
1. Compute synchronization time t_{sync}

1.3. Determine synchronization time t_{sync}

Universität Bremen

Source (Kröger 2010)

17



2. Reflexxes Library

2. Selection of acceleration / speed profiles

- Similar to 1.1.. But T_{sync} is given this time
- Synchronization

Subset of acceleration profiles

Part of the decision tree to find the acceleration profile

Universität Bremen

Source (Kröger 2010)

18

2. Reflexxes Library

Algorithm Outline

1. Compute synchronization time t_{sync}
   - 1.1. Minimum execution time for each DOF
   - 1.2. Find inoperative time intervals for each DOF
   - 1.3. Determine synchronization time t_{sync}

2. Selection of acceleration / speed profiles

3. **Generate new motion state for each DOF**
   - Trivial: apply acceleration profile to current motion state

3. Using the Library

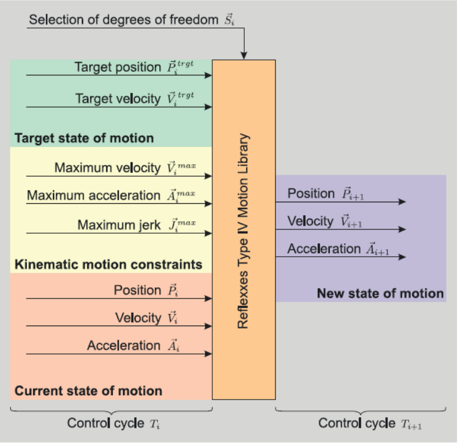
Available Libraries:

- Reflexxes Motion Libraries **Type II**
- Reflexxes Motion Libraries **Type IV**

→ Handle different sets of constraints

Target motion state

Constraints	Target speed = 0 AND acceleration = 0 AND jerk = 0	Target acceleration = 0 AND jerk = 0	Target jerk = 0	-
Limited acceleration	Type I	Type II	-	-
Limited acceleration AND jerk	Type III	Type IV	Type V	-
Limited accceleration AND jerk AND jerk-derivative	Type VI	Type VII	Type VIII	Type IX



Source (Kröger 2010)

3. Using the Library

ROCK:

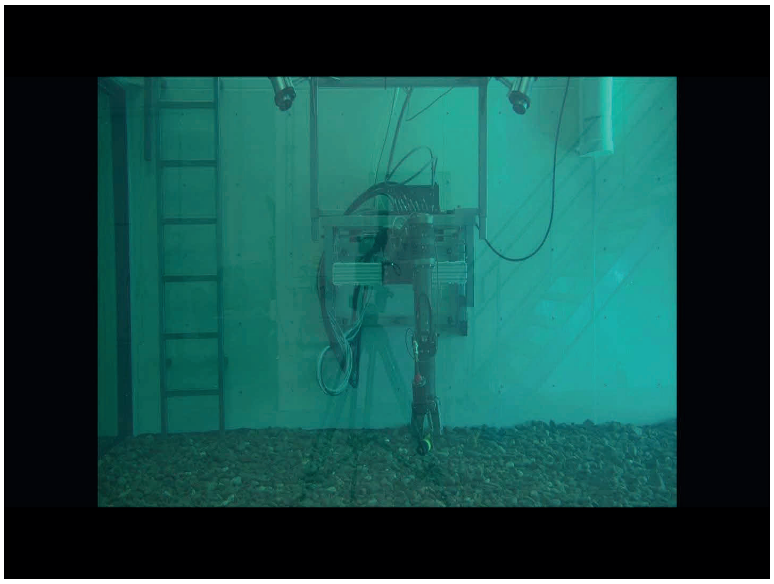
- Type II Library → on gitorious, „rock-control/reflexxes“
- Type IV Library → on spacegit, „besman-rock/reflexxes_type_iv“
- oroGen component → on gitorious, „rock-control/orogen/trajectory_generation“

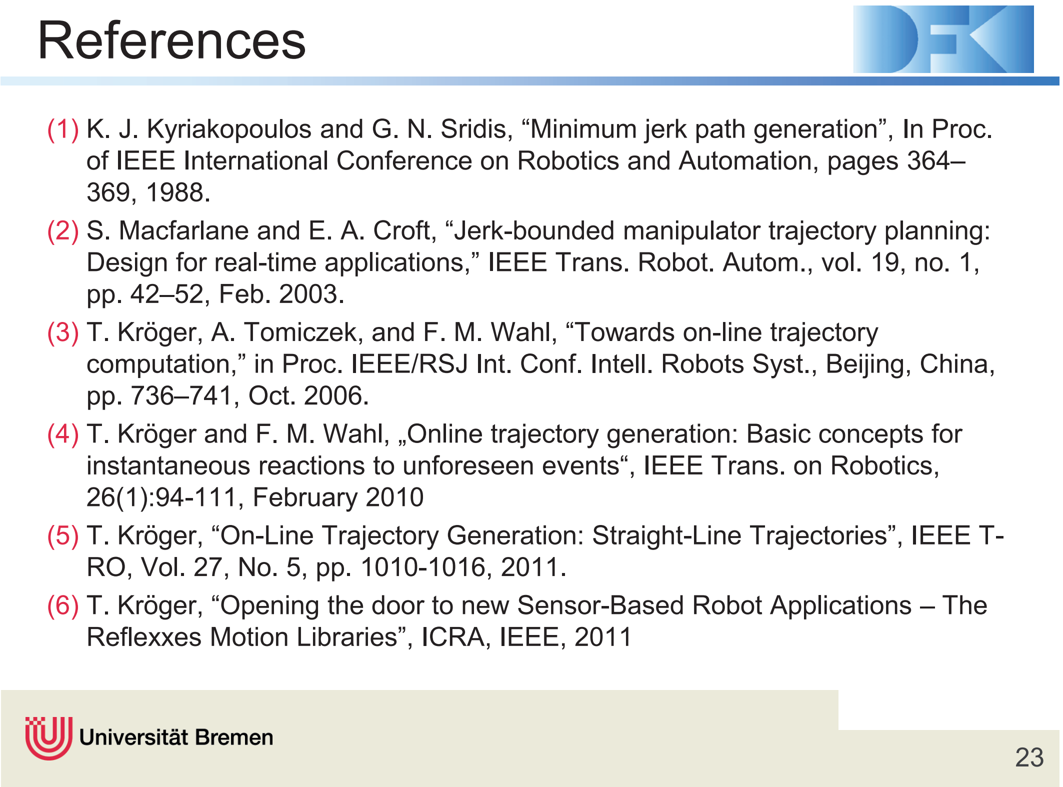
Matlab / Simulink:

Library Type IV embedded in a S-Function (project EO2)

Projects that use Reflexxes:

- Besman (control of Aila, collision avoidance)
- Cmanipulator demo (control of the Orion arm)
- Artemis arm
- Sherpa legs





References

(1) K. J. Kyriakopoulos and G. N. Sridis, “Minimum jerk path generation”, In Proc. of IEEE International Conference on Robotics and Automation, pages 364–369, 1988.

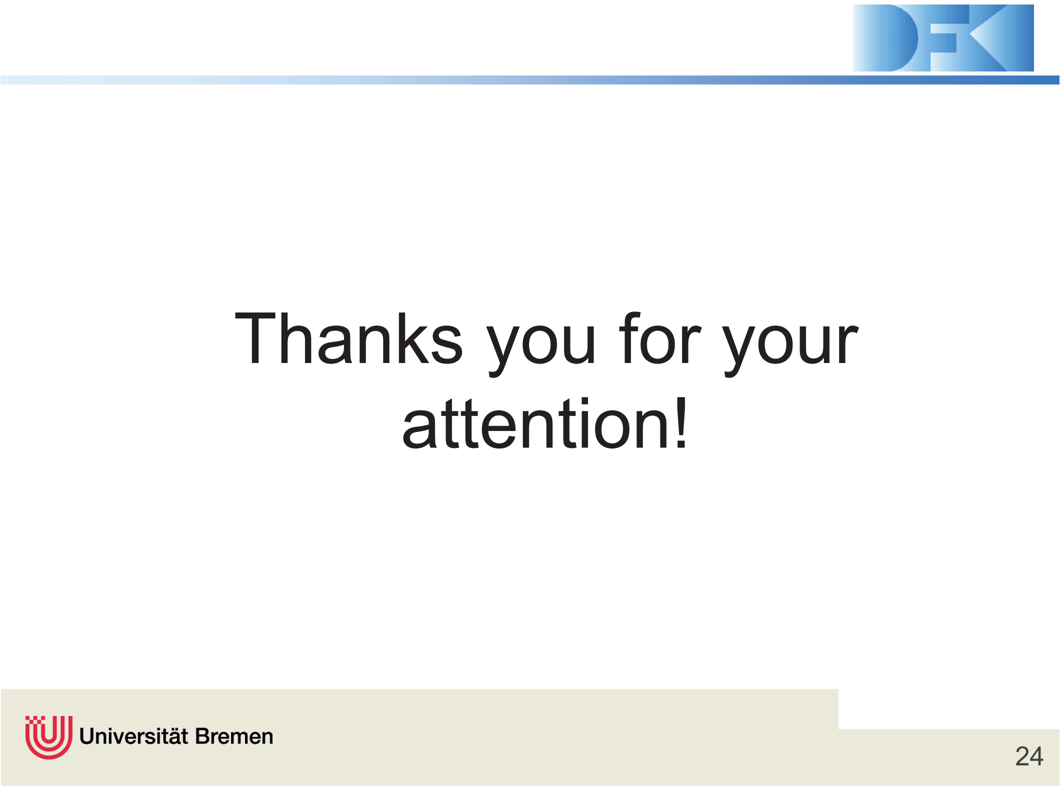
(2) S. Macfarlane and E. A. Croft, “Jerk-bounded manipulator trajectory planning: Design for real-time applications,” IEEE Trans. Robot. Autom., vol. 19, no. 1, pp. 42–52, Feb. 2003.

(3) T. Kröger, A. Tomiczek, and F. M. Wahl, “Towards on-line trajectory computation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Beijing, China, pp. 736–741, Oct. 2006.

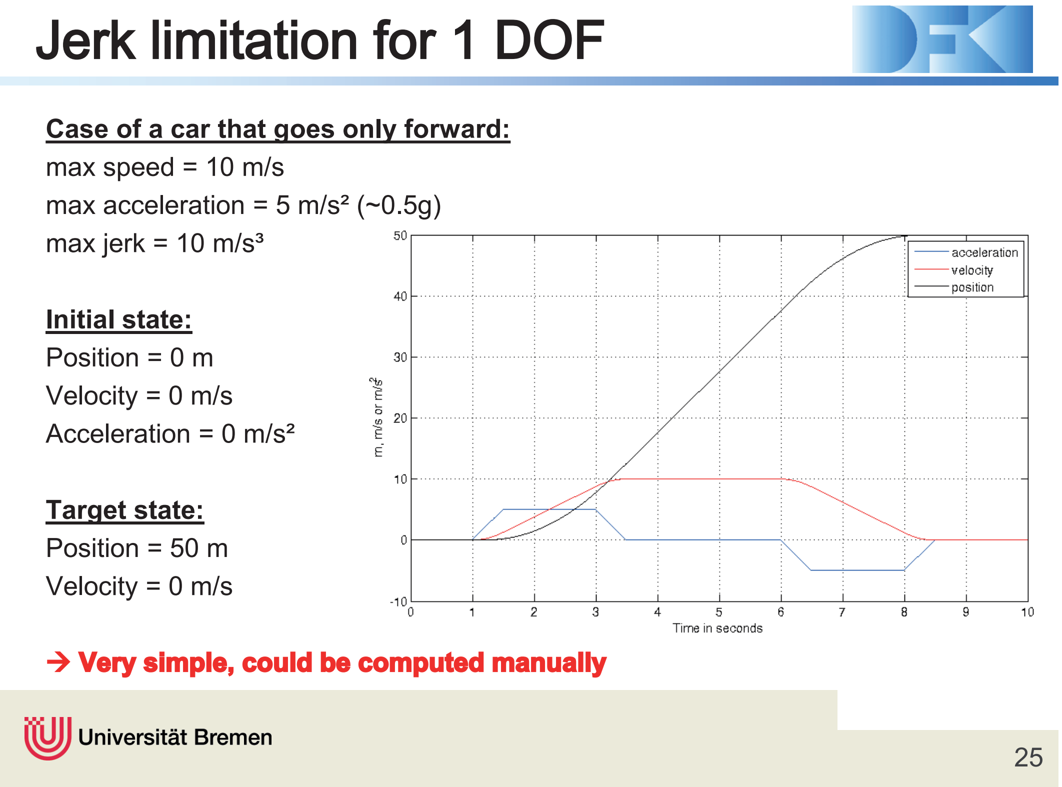
(4) T. Kröger and F. M. Wahl, „Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events“, IEEE Trans. on Robotics, 26(1):94-111, February 2010

(5) T. Kröger, “On-Line Trajectory Generation: Straight-Line Trajectories”, IEEE T-RO, Vol. 27, No. 5, pp. 1010-1016, 2011.

(6) T. Kröger, “Opening the door to new Sensor-Based Robot Applications – The Reflexxes Motion Libraries”, ICRA, IEEE, 2011



Thanks you for your attention!



Jerk limitation for 1 DOF

Case of a car that goes only forward:

max speed = 10 m/s

max acceleration = 5 m/s² (~0.5g)

max jerk = 10 m/s³

Initial state:

Position = 0 m

Velocity = 0 m/s

Acceleration = 0 m/s²

Target state:

Position = 50 m

Velocity = 0 m/s

→ Very simple, could be computed manually

2. Reflexxes Algorithm

There are two C++ libraries:

- Reflexxes Motion Libraries **Type II**
- Reflexxes Motion Libraries **Type IV**

→ Corresponds to different constraints on the Online Trajectory Generator (OTG)

3.6 'Cascaded Robot Joint Control' (MC-T-06)

Vinzenz Bargsten(1)

(1) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `bargsten@uni-bremen.de`

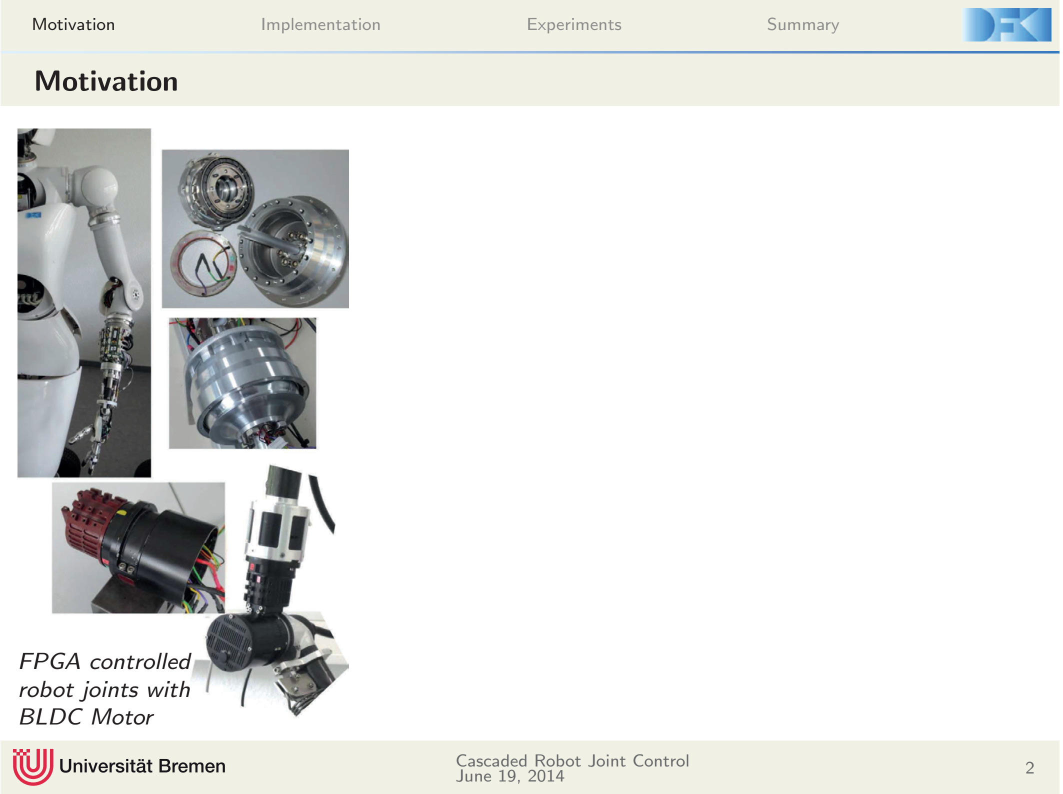
Abstract

High-Level robot control algorithms rely on the low-level control of robotic joints. This talk presents a FPGA-based, cascaded control approach for robotic joints. The approach allows to deal with different control objectives and joint limits.

Cascaded Robot Joint Control

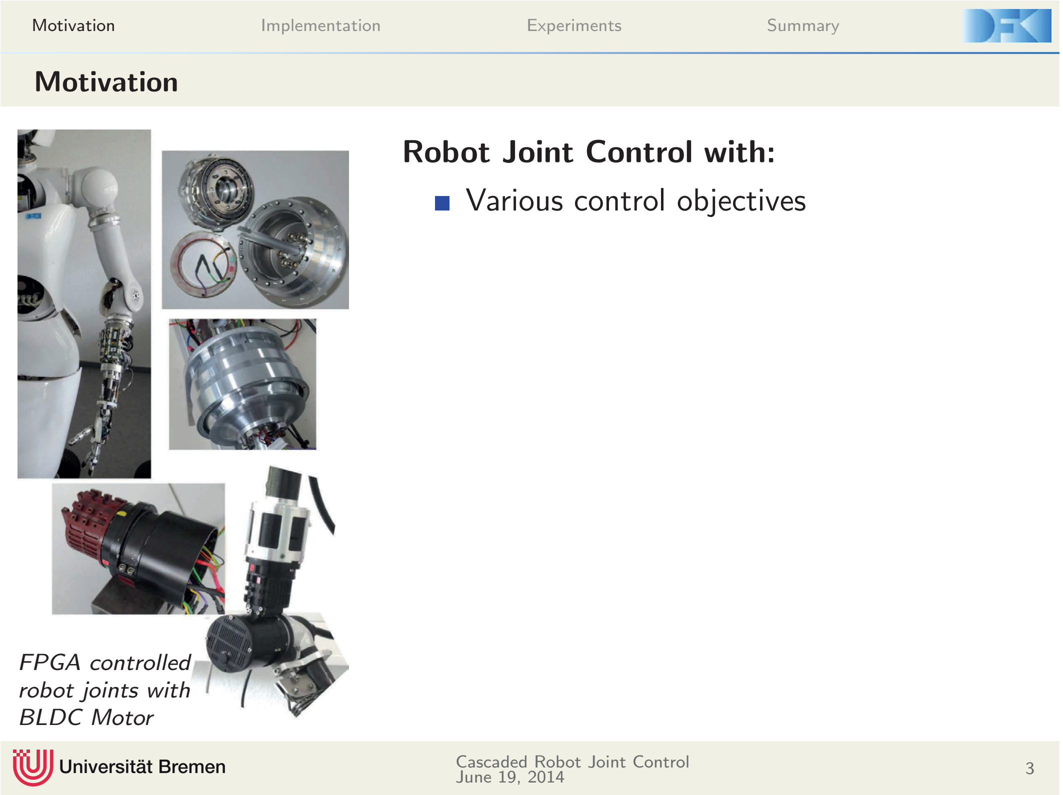
Vinzenz Bargsten

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de



Motivation

FPGA controlled robot joints with BLDC Motor

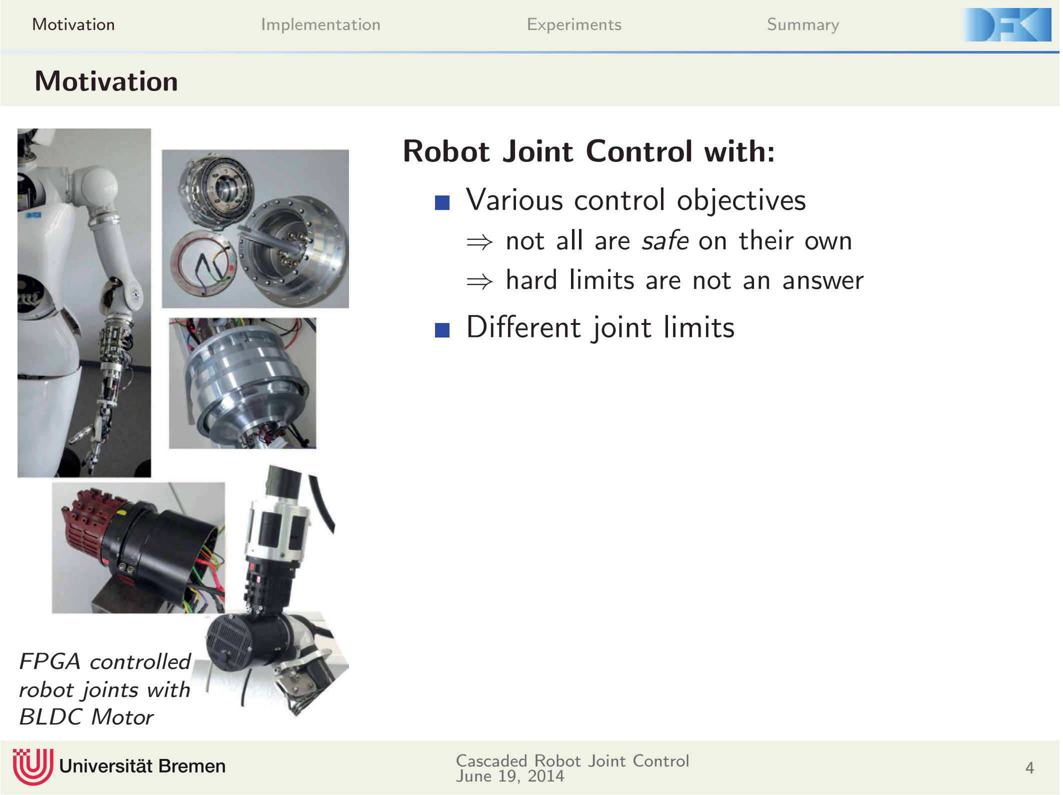


Motivation

Robot Joint Control with:

- Various control objectives

FPGA controlled robot joints with BLDC Motor

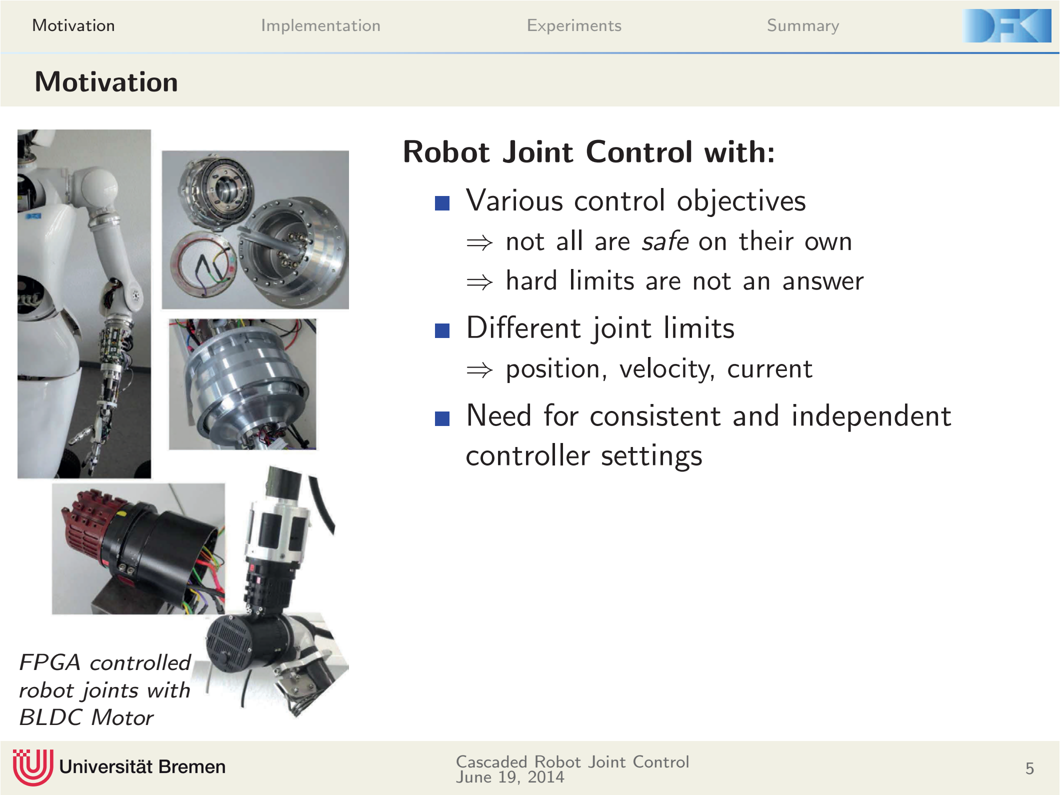


Motivation

Robot Joint Control with:

- Various control objectives
  - ⇒ not all are *safe* on their own
  - ⇒ hard limits are not an answer
- Different joint limits

FPGA controlled robot joints with BLDC Motor



Motivation Implementation Experiments Summary

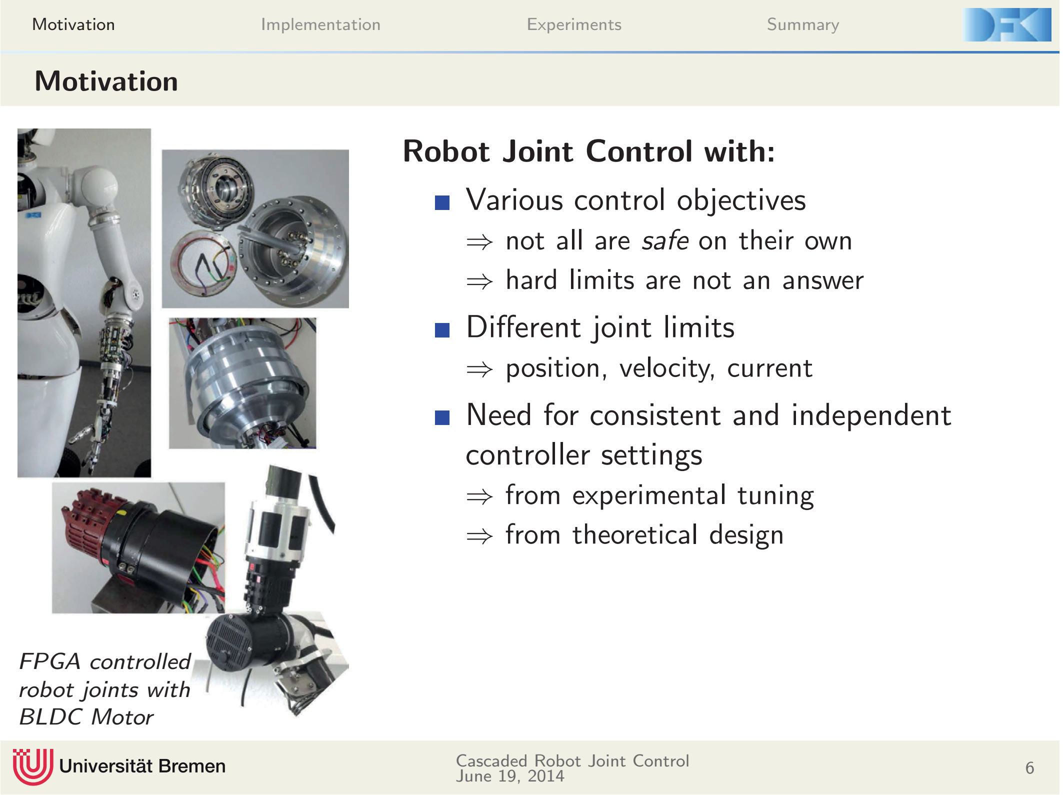
Motivation

FPGA controlled robot joints with BLDC Motor

Robot Joint Control with:

- Various control objectives
  - ⇒ not all are *safe* on their own
  - ⇒ hard limits are not an answer
- Different joint limits
  - ⇒ position, velocity, current
- Need for consistent and independent controller settings

Universität Bremen Cascaded Robot Joint Control June 19, 2014 5



Motivation Implementation Experiments Summary

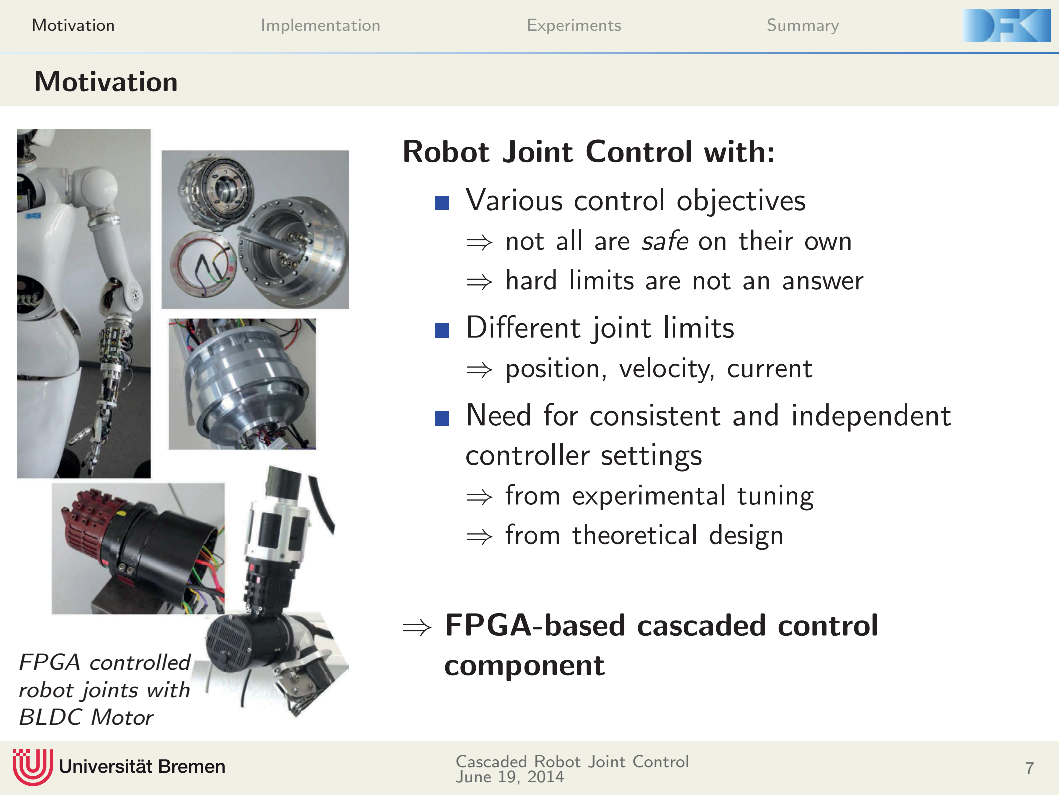
Motivation

FPGA controlled robot joints with BLDC Motor

Robot Joint Control with:

- Various control objectives
  - ⇒ not all are *safe* on their own
  - ⇒ hard limits are not an answer
- Different joint limits
  - ⇒ position, velocity, current
- Need for consistent and independent controller settings
  - ⇒ from experimental tuning
  - ⇒ from theoretical design

Universität Bremen Cascaded Robot Joint Control June 19, 2014 6



Motivation Implementation Experiments Summary

Motivation

Robot Joint Control with:

- Various control objectives
  - ⇒ not all are *safe* on their own
  - ⇒ hard limits are not an answer
- Different joint limits
  - ⇒ position, velocity, current
- Need for consistent and independent controller settings
  - ⇒ from experimental tuning
  - ⇒ from theoretical design

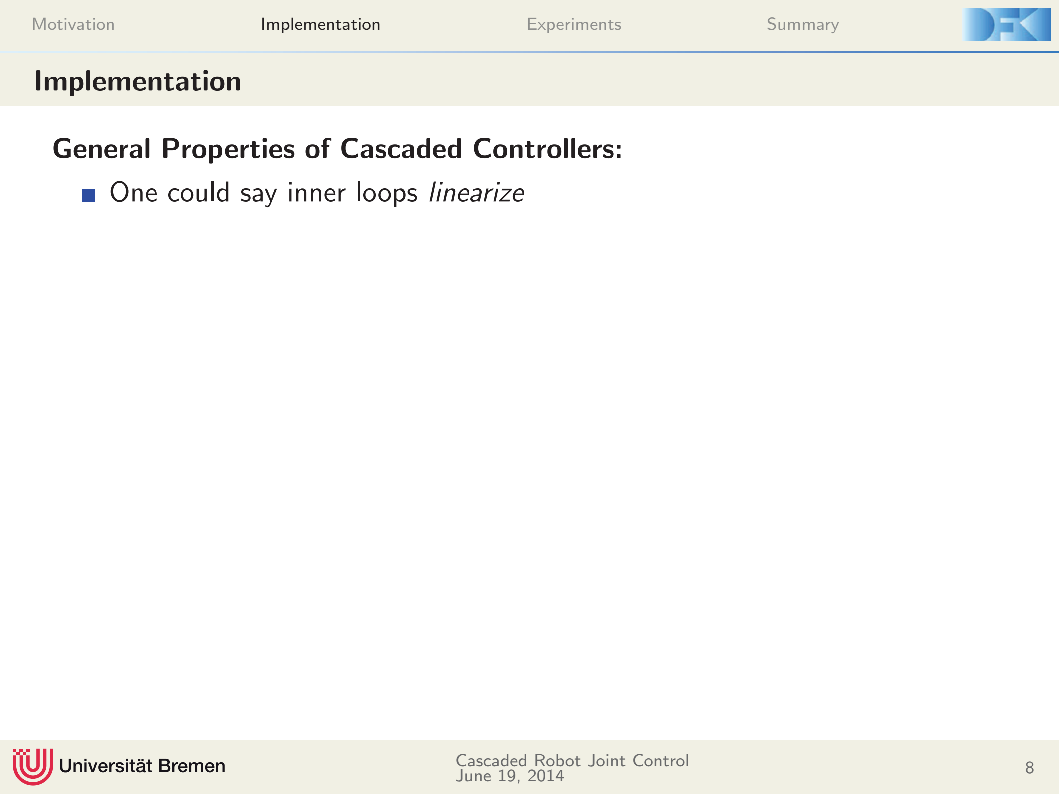
⇒ **FPGA-based cascaded control component**

FPGA controlled robot joints with BLDC Motor

Universität Bremen

Cascaded Robot Joint Control
June 19, 2014

7



Motivation Implementation Experiments Summary

Implementation

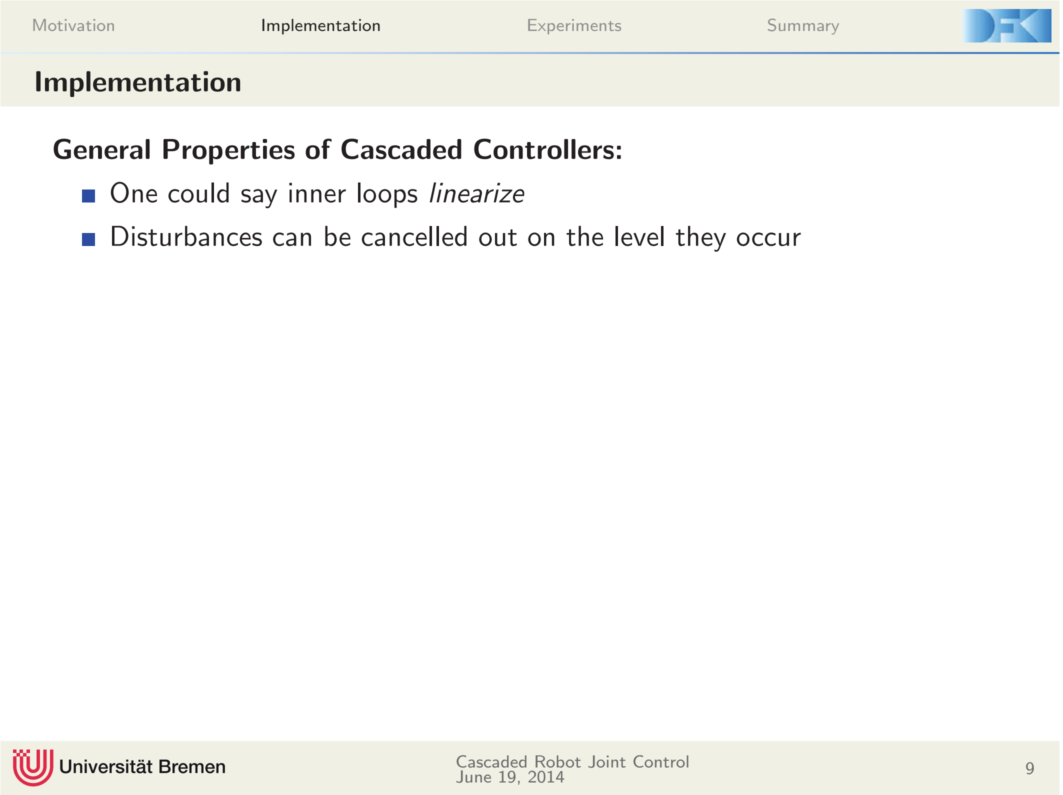
General Properties of Cascaded Controllers:

- One could say inner loops *linearize*

Universität Bremen

Cascaded Robot Joint Control
June 19, 2014

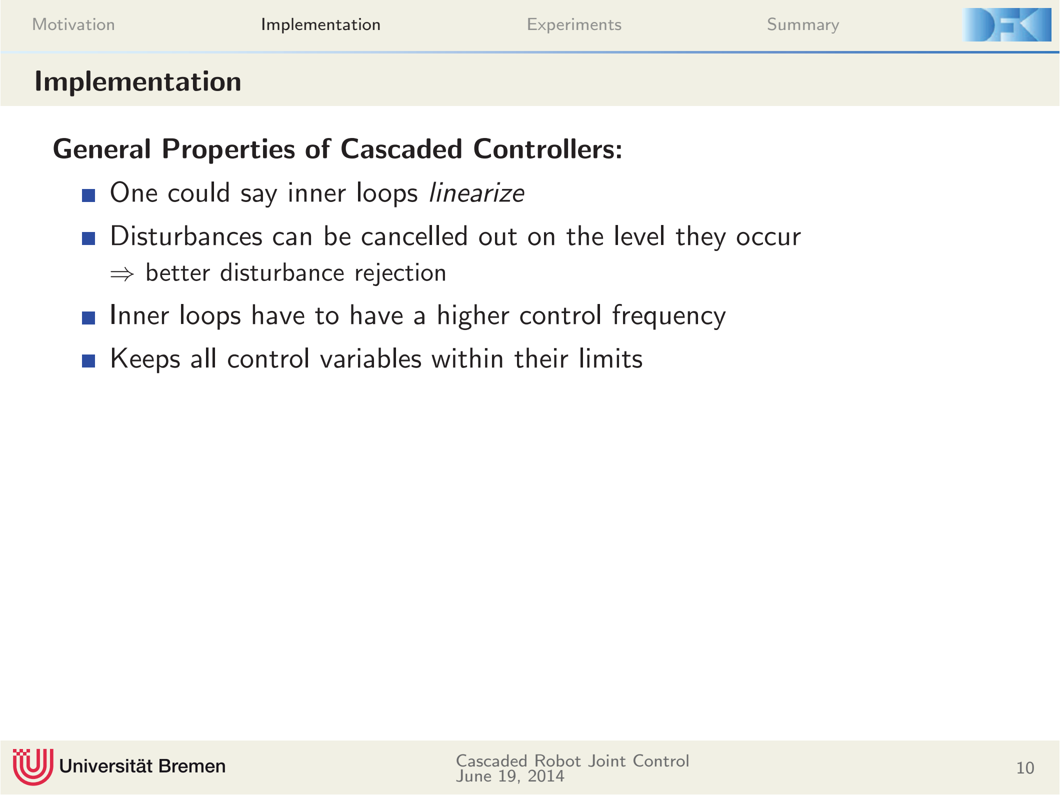
8



Implementation

General Properties of Cascaded Controllers:

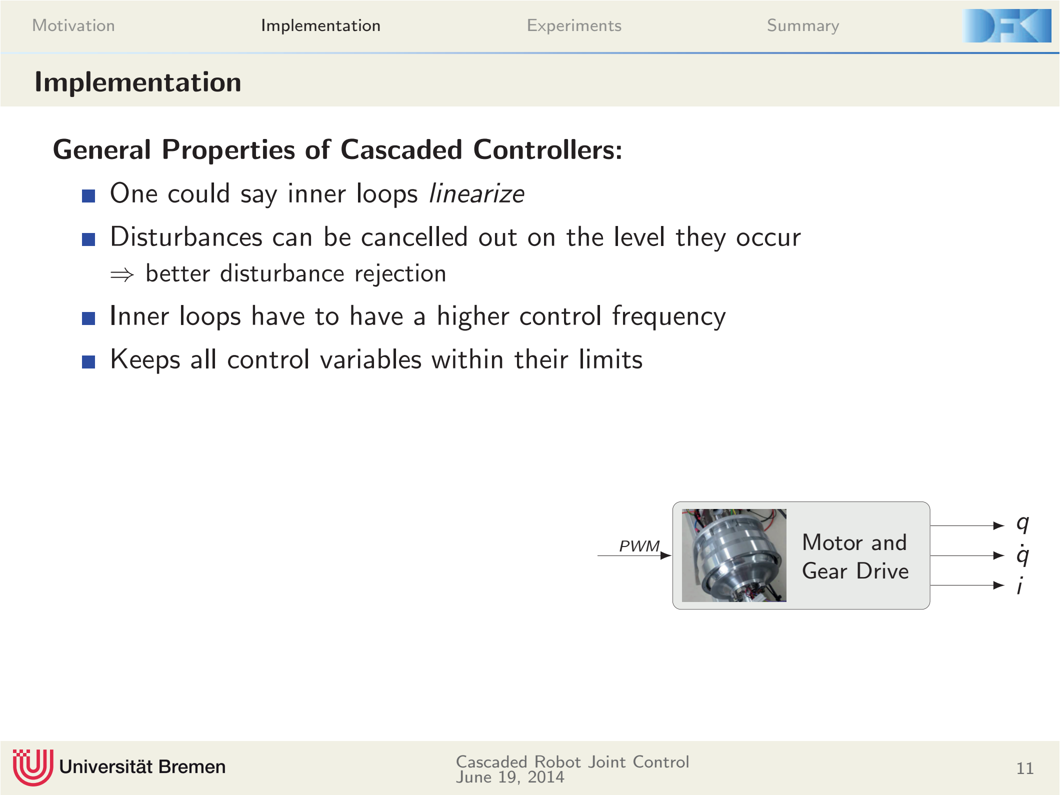
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur



Implementation

General Properties of Cascaded Controllers:

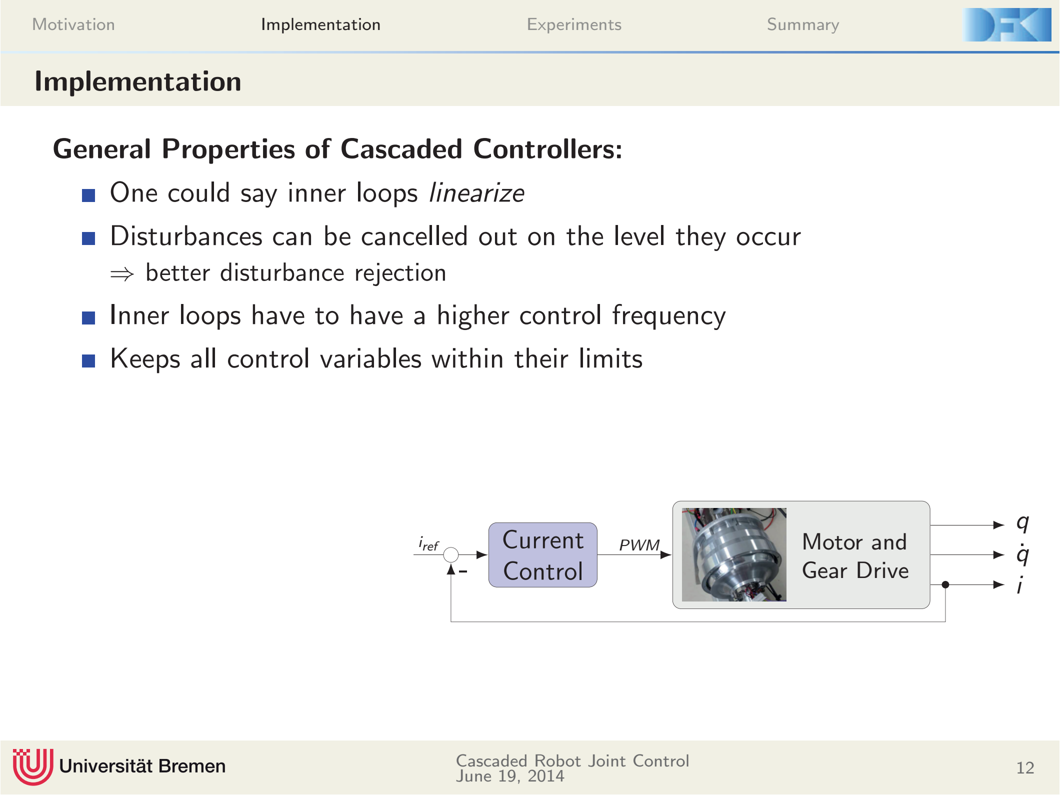
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur
  ⇒ better disturbance rejection
- Inner loops have to have a higher control frequency
- Keeps all control variables within their limits



Implementation

General Properties of Cascaded Controllers:

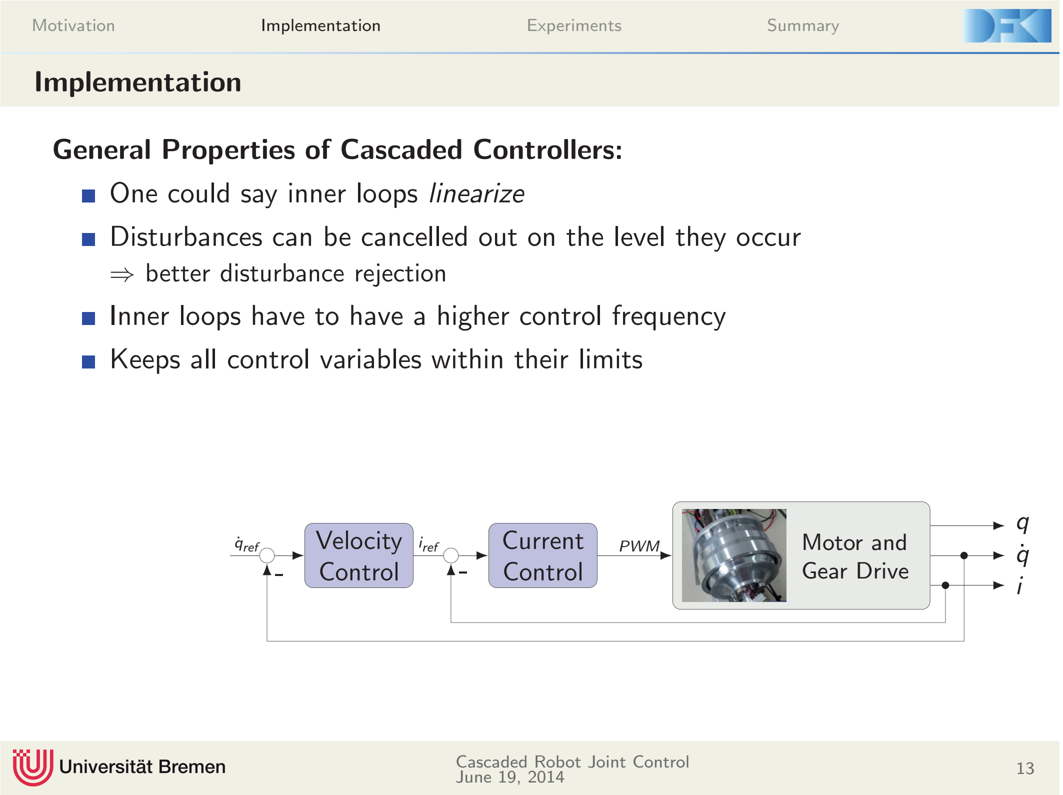
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur
  ⇒ better disturbance rejection
- Inner loops have to have a higher control frequency
- Keeps all control variables within their limits



Implementation

General Properties of Cascaded Controllers:

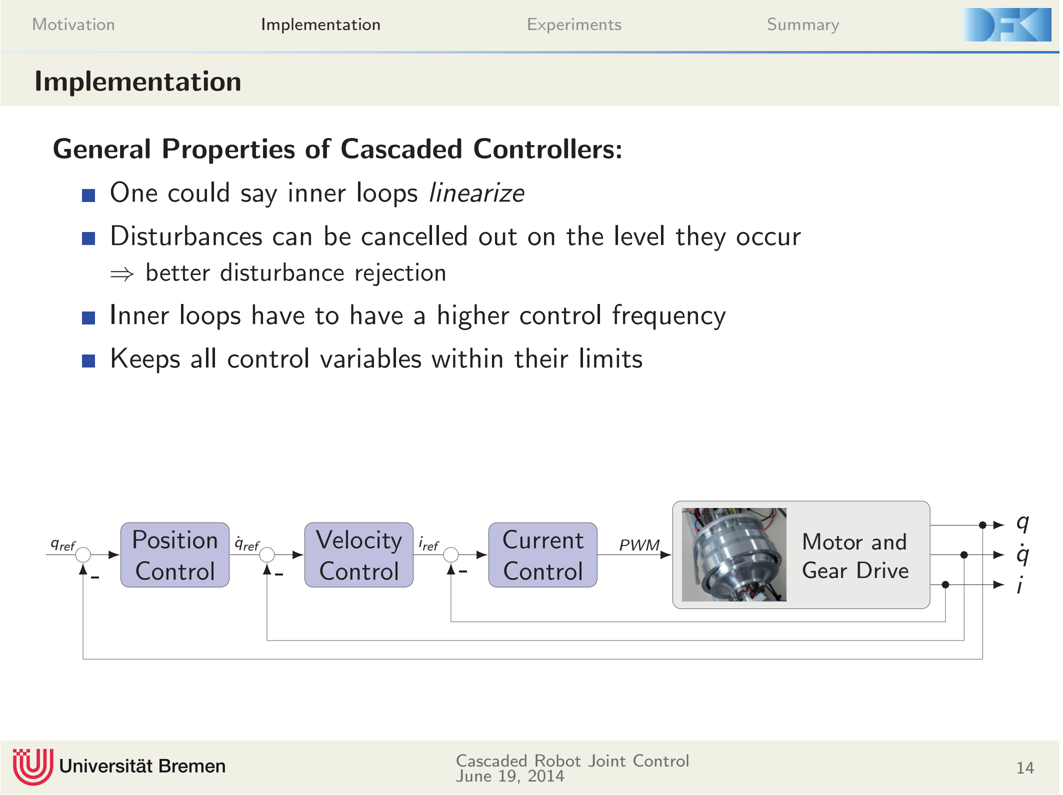
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur
  ⇒ better disturbance rejection
- Inner loops have to have a higher control frequency
- Keeps all control variables within their limits



Implementation

General Properties of Cascaded Controllers:

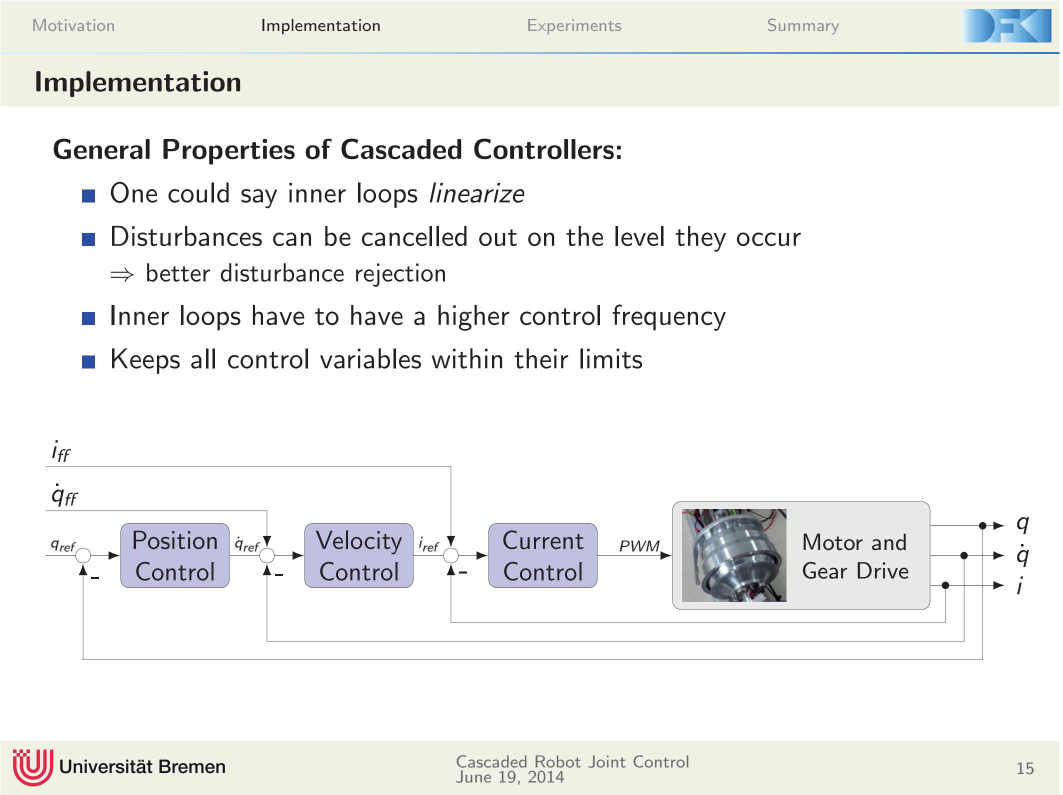
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur
  ⇒ better disturbance rejection
- Inner loops have to have a higher control frequency
- Keeps all control variables within their limits



Implementation

General Properties of Cascaded Controllers:

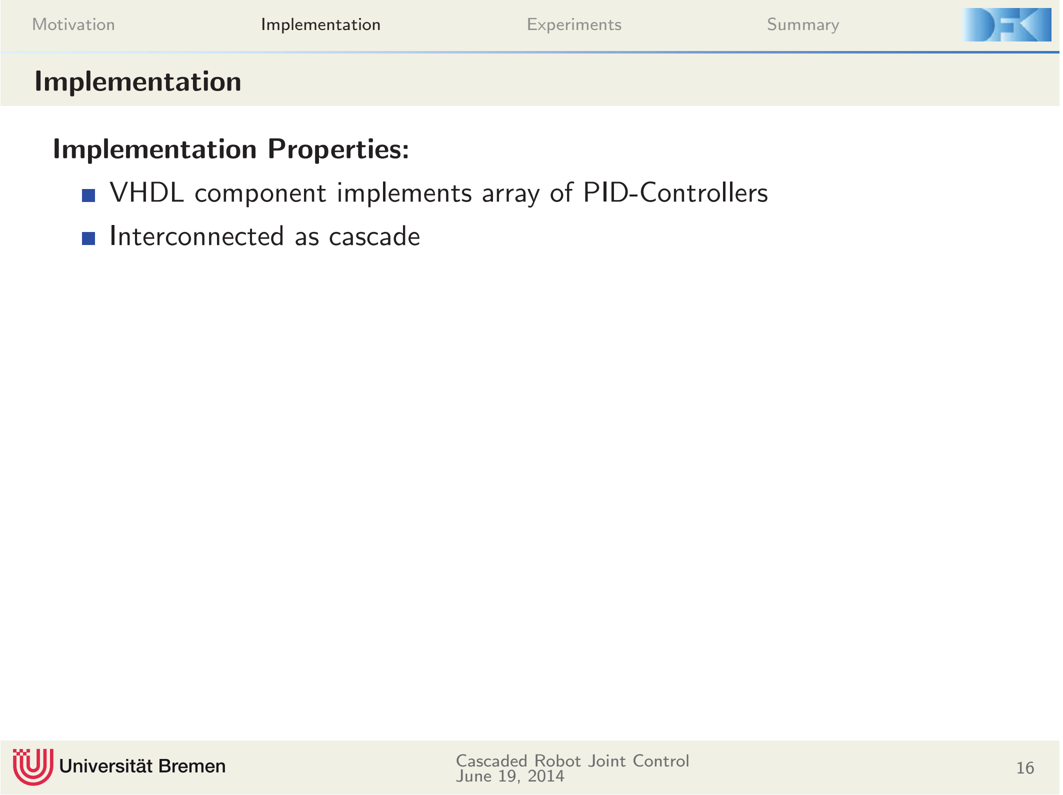
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur
  ⇒ better disturbance rejection
- Inner loops have to have a higher control frequency
- Keeps all control variables within their limits



Implementation

General Properties of Cascaded Controllers:

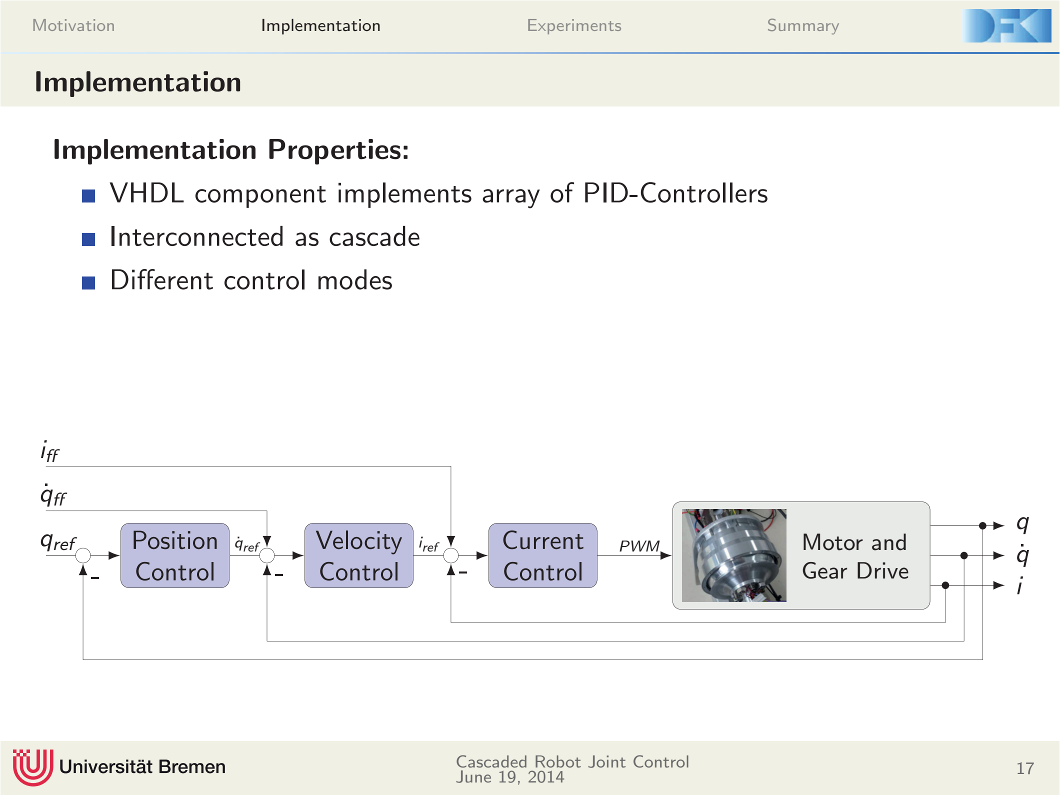
- One could say inner loops *linearize*
- Disturbances can be cancelled out on the level they occur
  ⇒ better disturbance rejection
- Inner loops have to have a higher control frequency
- Keeps all control variables within their limits



Implementation

Implementation Properties:

- VHDL component implements array of PID-Controllers
- Interconnected as cascade



Motivation Implementation Experiments Summary

Implementation

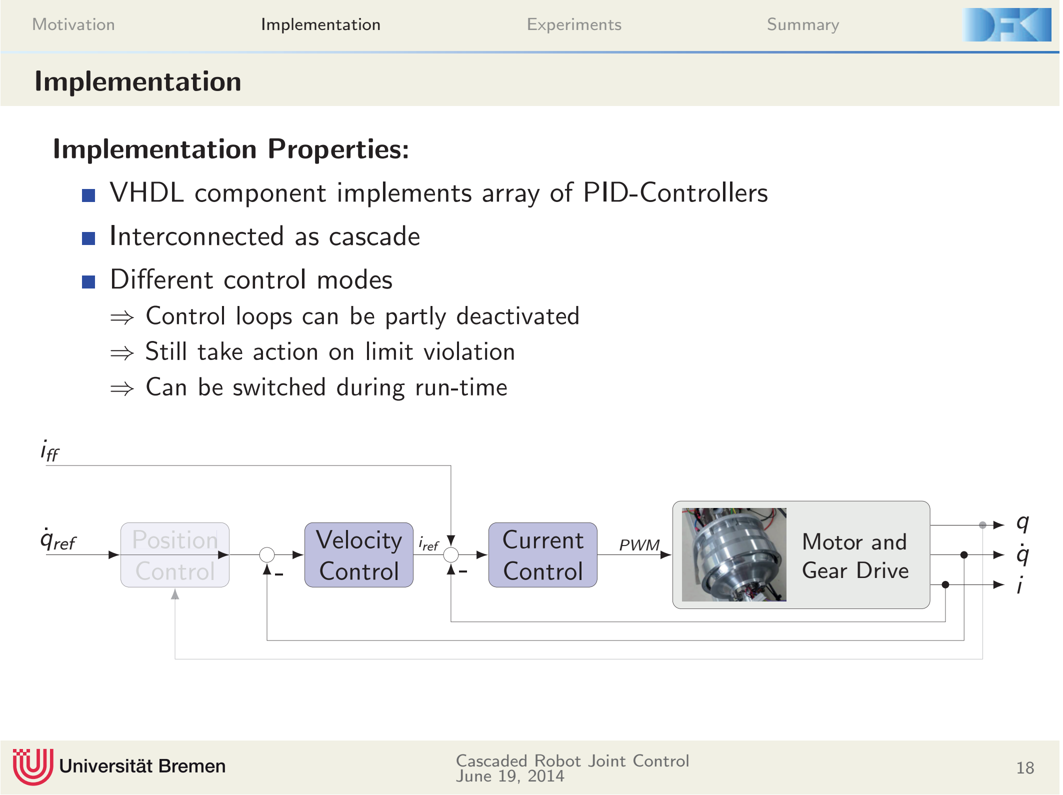
Implementation Properties:

- VHDL component implements array of PID-Controllers
- Interconnected as cascade
- Different control modes

Universität Bremen

Cascaded Robot Joint Control
June 19, 2014

17



Motivation Implementation Experiments Summary

Implementation

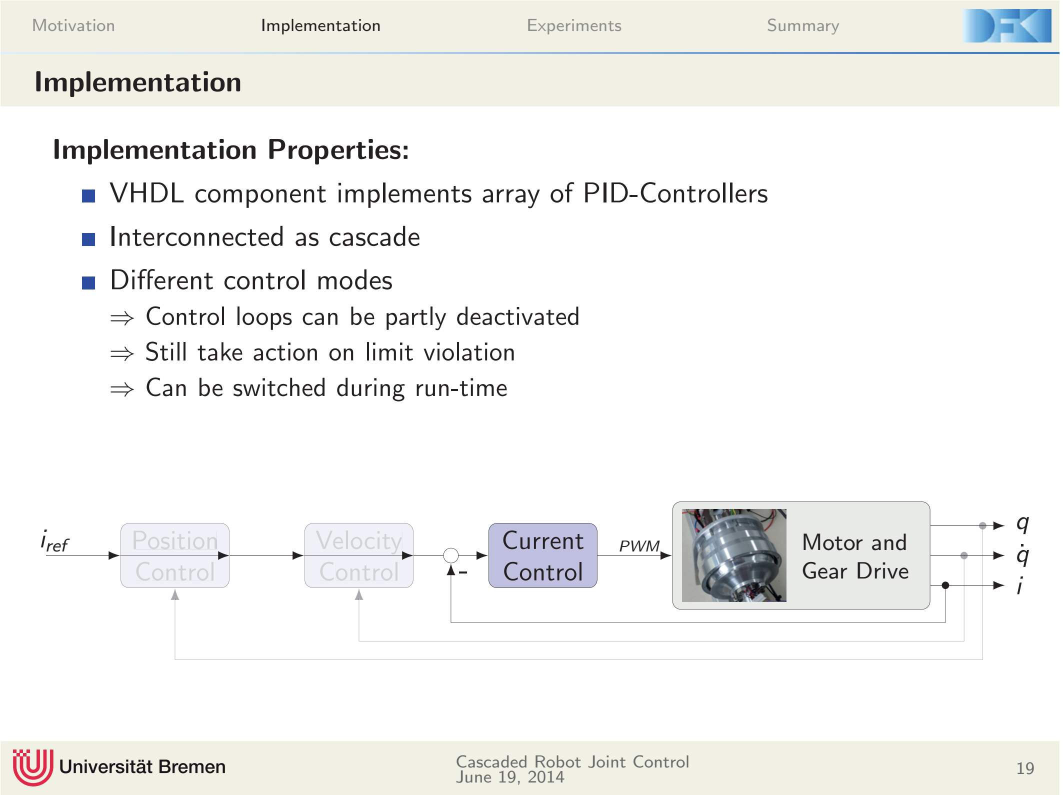
Implementation Properties:

- VHDL component implements array of PID-Controllers
- Interconnected as cascade
- Different control modes
  - ⇒ Control loops can be partly deactivated
  - ⇒ Still take action on limit violation
  - ⇒ Can be switched during run-time

Universität Bremen

Cascaded Robot Joint Control
June 19, 2014

18



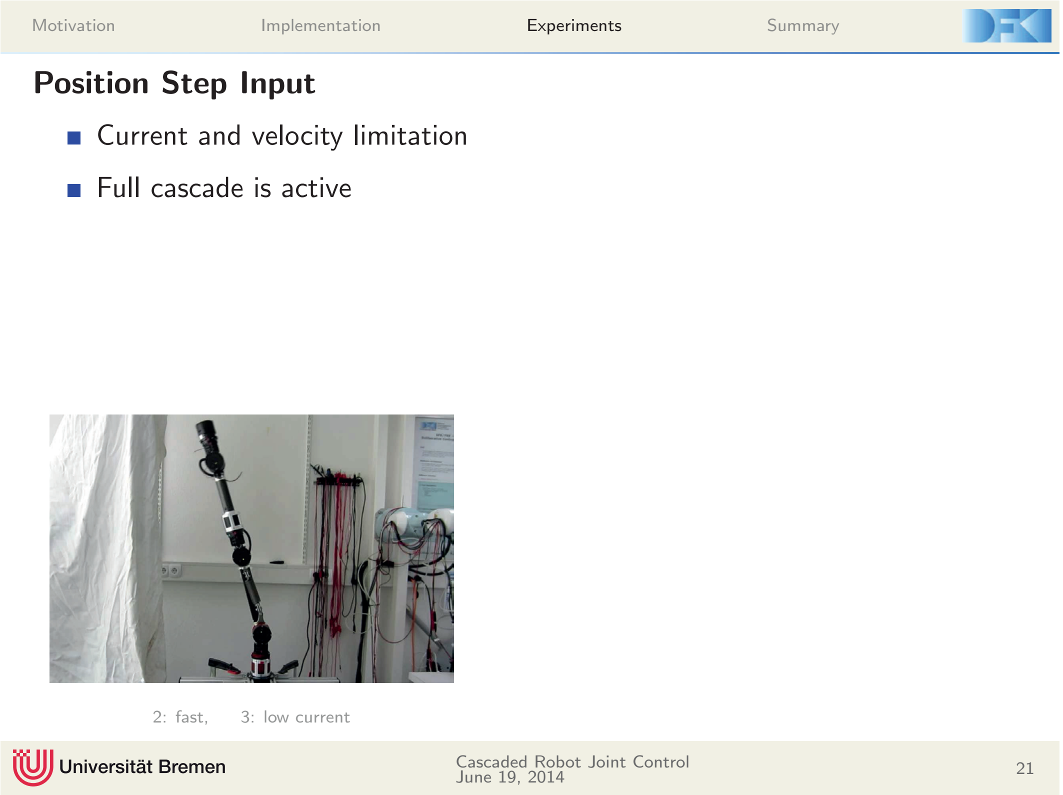
Implementation

Implementation Properties:

- VHDL component implements array of PID-Controllers
- Interconnected as cascade
- Different control modes
  - ⇒ Control loops can be partly deactivated
  - ⇒ Still take action on limit violation
  - ⇒ Can be switched during run-time

Position Step Input

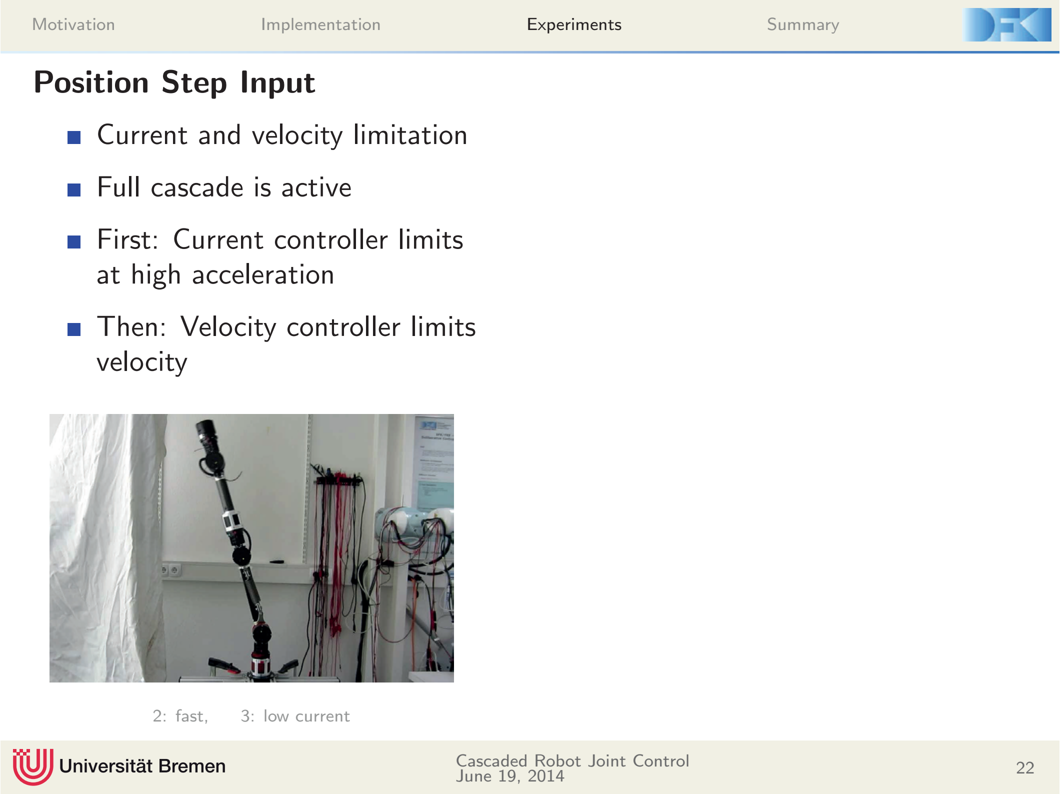
- Current and velocity limitation
- Full cascade is active



Position Step Input

- Current and velocity limitation
- Full cascade is active

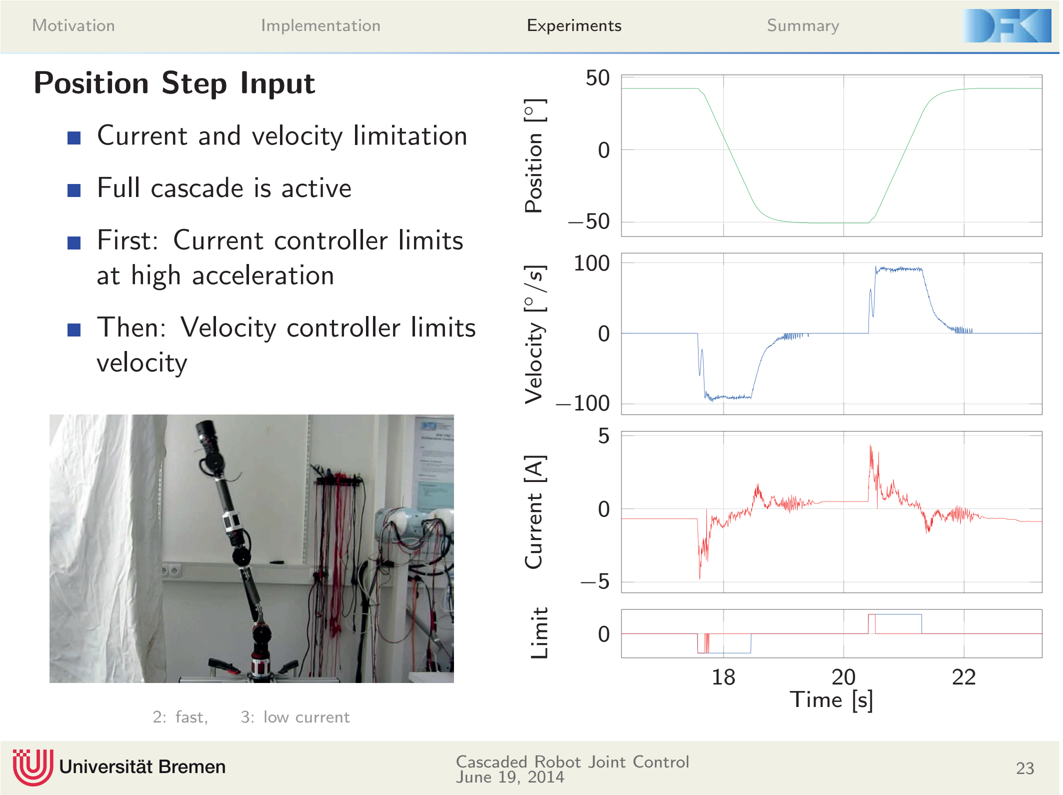
2: fast, 3: low current



Position Step Input

- Current and velocity limitation
- Full cascade is active
- First: Current controller limits at high acceleration
- Then: Velocity controller limits velocity

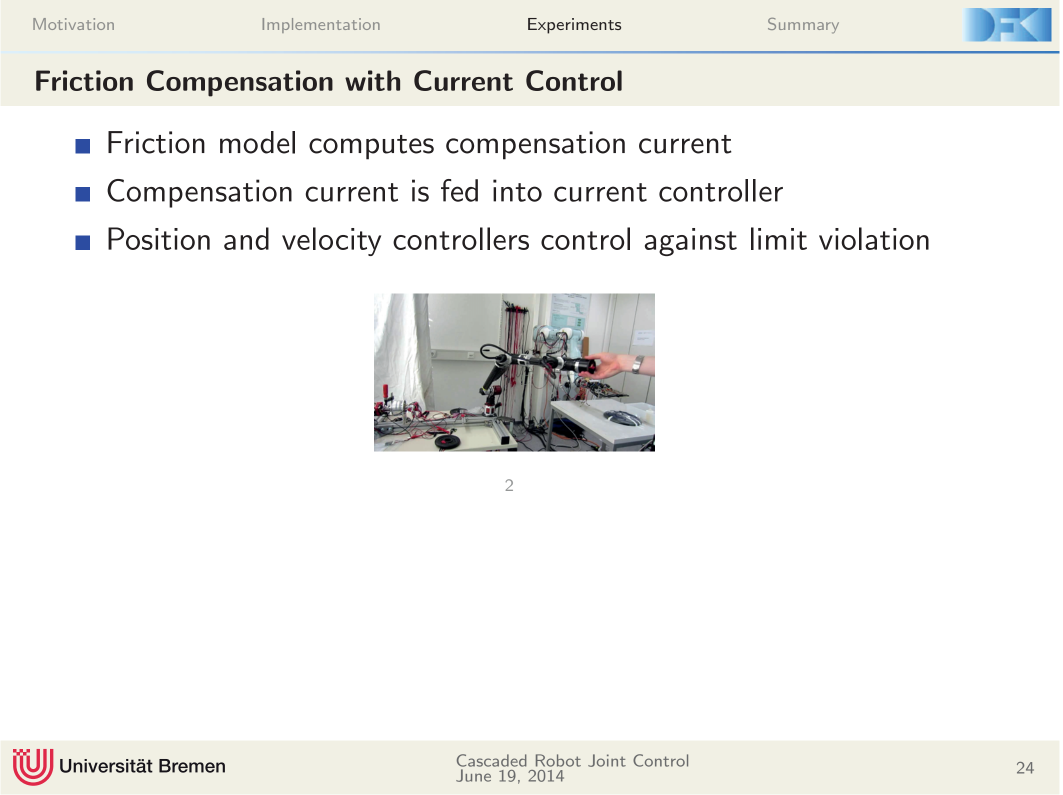
2: fast, 3: low current



Position Step Input

- Current and velocity limitation
- Full cascade is active
- First: Current controller limits at high acceleration
- Then: Velocity controller limits velocity

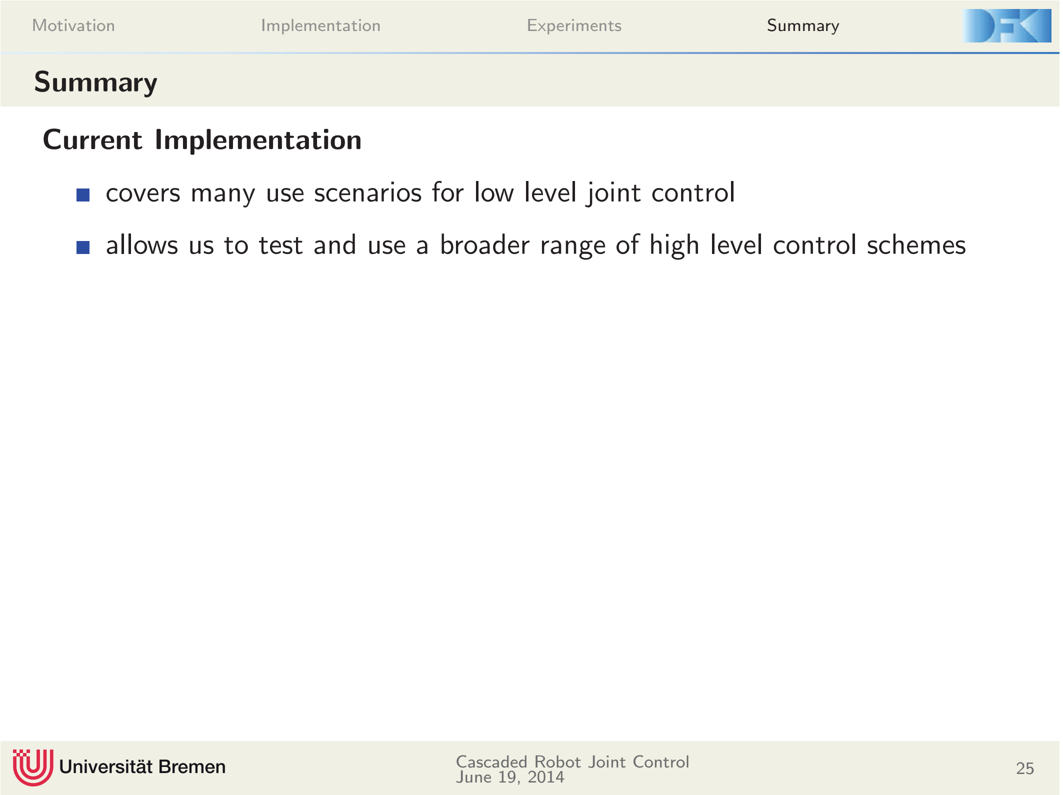
2: fast, 3: low current



Friction Compensation with Current Control

- Friction model computes compensation current
- Compensation current is fed into current controller
- Position and velocity controllers control against limit violation

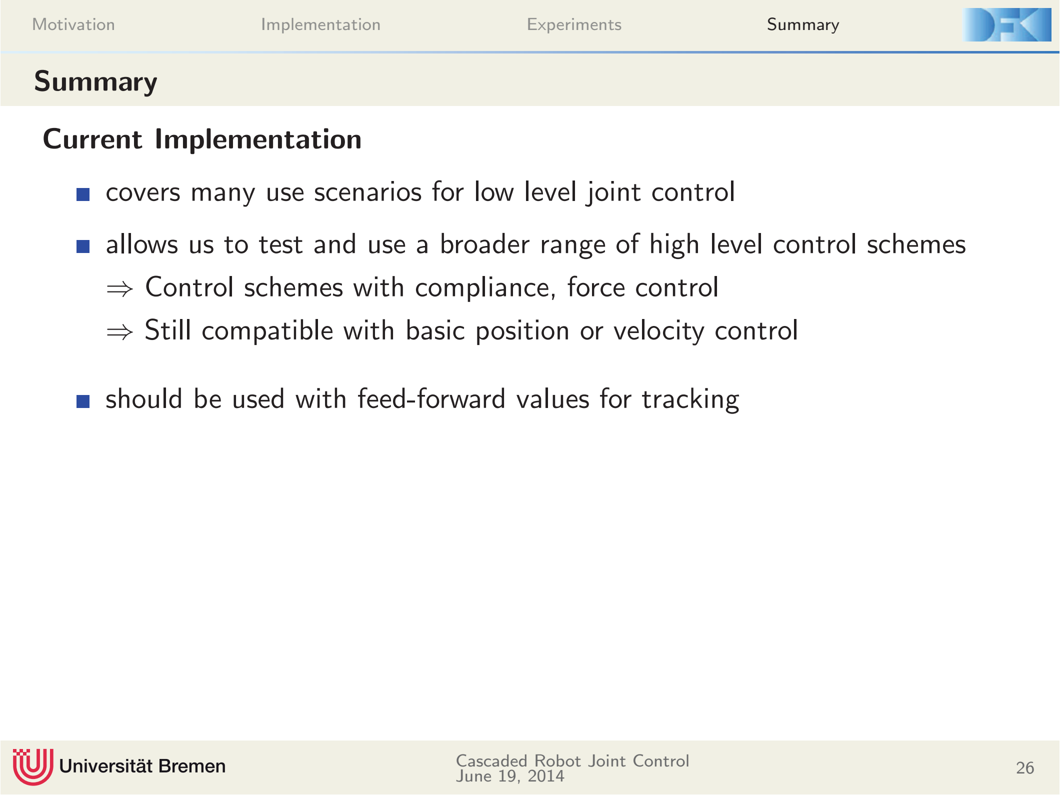
2



Summary

Current Implementation

- covers many use scenarios for low level joint control
- allows us to test and use a broader range of high level control schemes



Summary

Current Implementation

- covers many use scenarios for low level joint control
- allows us to test and use a broader range of high level control schemes
  $\Rightarrow$ Control schemes with compliance, force control
  $\Rightarrow$ Still compatible with basic position or velocity control
- should be used with feed-forward values for tracking

Summary

Current Implementation

- covers many use scenarios for low level joint control
- allows us to test and use a broader range of high level control schemes
  ⇒ Control schemes with compliance, force control
  ⇒ Still compatible with basic position or velocity control
- should be used with feed-forward values for tracking
- supports arbitrary interconnection as well
- can be used for different purposes (I know from hand control)

Summary

Current Implementation

- covers many use scenarios for low level joint control
- allows us to test and use a broader range of high level control schemes
  ⇒ Control schemes with compliance, force control
  ⇒ Still compatible with basic position or velocity control
- should be used with feed-forward values for tracking
- supports arbitrary interconnection as well
- can be used for different purposes (I know from hand control)

Future Aspects:

- Auto Tuning

3.7 ‘Autonomous Steering Controller for Path Following’ (MC-P-01)

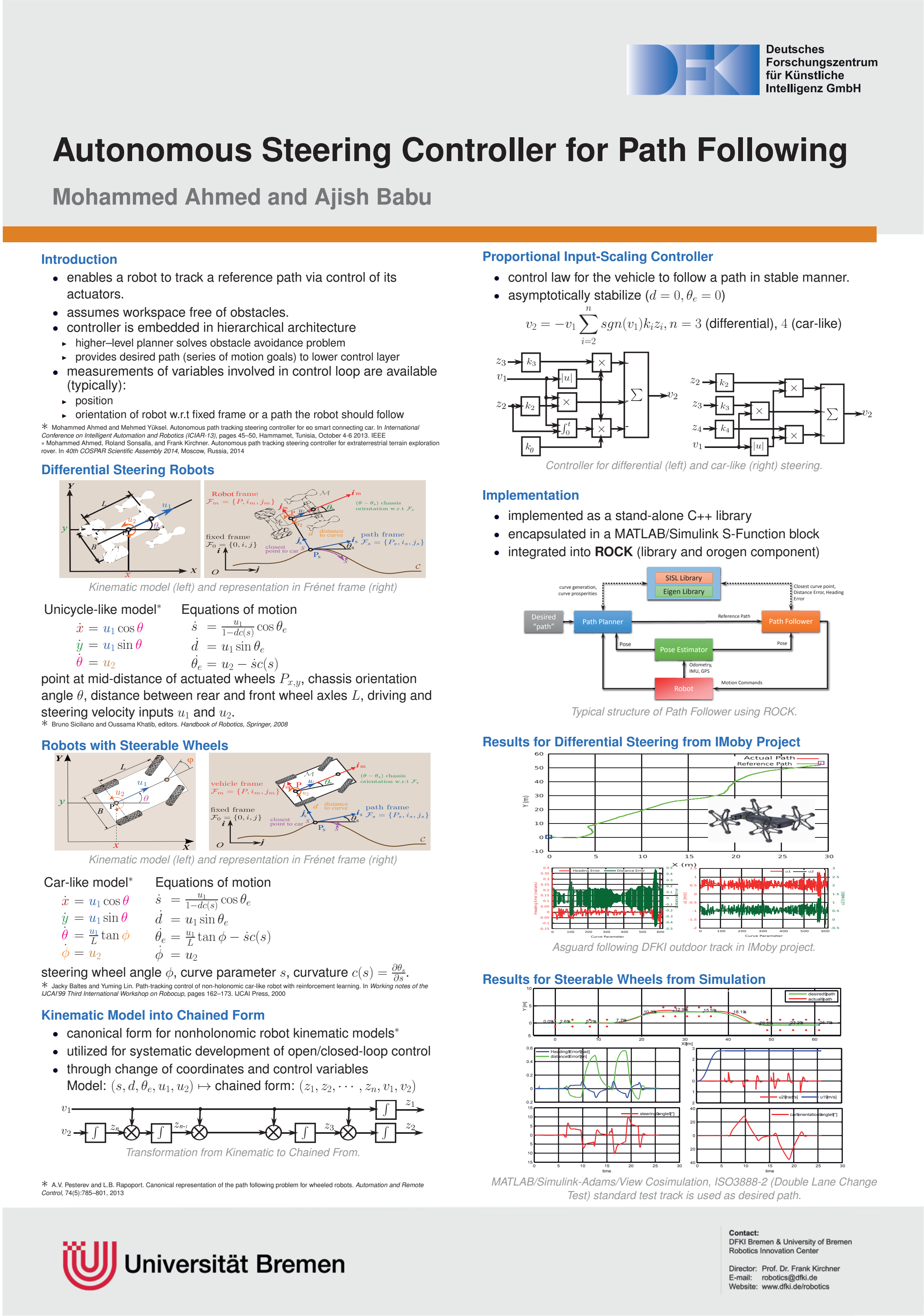
Mohammed Ahmed(1), *Ajish Babu*(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `mohammed.ahmed@dfki.de`, `ajish.babu@dfki.de`

Abstract

This poster presents an autonomous path following controller for mobile robots. Controller designs using the kinematic model of the robot is discussed for both differential and car-like steering. The kinematic model of the robot is transformed to chained form, from which the controller is developed. It is intergrated in ROCK frame work and are being used by many projects at DFKI. Results of the controller from the IMoby and EO2 projects are also presented.



Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Autonomous Steering Controller for Path Following

Mohammed Ahmed and Ajish Babu

Introduction

- enables a robot to track a reference path via control of its actuators.
- assumes workspace free of obstacles.
- controller is embedded in hierarchical architecture
  - higher–level planner solves obstacle avoidance problem
  - provides desired path (series of motion goals) to lower control layer
- measurements of variables involved in control loop are available (typically):
  - position
  - orientation of robot w.r.t fixed frame or a path the robot should follow

∗ Mohammed Ahmed and Mehmed Yüksel. Autonomous path tracking steering controller for eo smart connecting car. In *International Conference on Intelligent Automation and Robotics (ICIAR-13)*, pages 45–50, Hammamet, Tunisia, October 4-6 2013. IEEE

∗ Mohammed Ahmed, Roland Sonsalla, and Frank Kirchner. Autonomous path tracking steering controller for extraterrestrial terrain exploration rover. In *40th COSPAR Scientific Assembly 2014*, Moscow, Russia, 2014

Differential Steering Robots

Kinematic model (left) and representation in Frénet frame (right)

Unicycle-like model∗

$$\dot{x} = u_1 \cos\theta \qquad \dot{y} = u_1 \sin\theta \qquad \dot{\theta} = u_2$$

Equations of motion

$$\dot{s} = \tfrac{u_1}{1-dc(s)} \cos\theta_e \qquad \dot{d} = u_1 \sin\theta_e \qquad \dot{\theta}_e = u_2 - \dot{s}c(s)$$

point at mid-distance of actuated wheels $P_{x,y}$, chassis orientation angle θ, distance between rear and front wheel axles L, driving and steering velocity inputs u_1 and u_2.

∗ Bruno Siciliano and Oussama Khatib, editors. *Handbook of Robotics, Springer, 2008*

Robots with Steerable Wheels

Kinematic model (left) and representation in Frénet frame (right)

Car-like model∗

$$\dot{x} = u_1 \cos\theta \qquad \dot{y} = u_1 \sin\theta \qquad \dot{\theta} = \tfrac{u_1}{L}\tan\phi \qquad \dot{\phi} = u_2$$

Equations of motion

$$\dot{s} = \tfrac{u_1}{1-dc(s)} \cos\theta_e \qquad \dot{d} = u_1 \sin\theta_e \qquad \dot{\theta}_e = \tfrac{u_1}{L}\tan\phi - \dot{s}c(s) \qquad \dot{\phi} = u_2$$

steering wheel angle ϕ, curve parameter s, curvature $c(s) = \frac{\partial\theta_s}{\partial s}$.

∗ Jacky Baltes and Yuming Lin. Path-tracking control of non-holonomic car-like robot with reinforcement learning. In *Working notes of the IJCAI'99 Third International Workshop on Robocup*, pages 162–173. IJCAI Press, 2000

Kinematic Model into Chained Form

- canonical form for nonholonomic robot kinematic models∗
- utilized for systematic development of open/closed-loop control
- through change of coordinates and control variables
  Model: $(s, d, \theta_e, u_1, u_2) \mapsto$ chained form: $(z_1, z_2, \cdots, z_n, v_1, v_2)$

Transformation from Kinematic to Chained From.

∗ A.V. Pesterev and L.B. Rapoport. Canonical representation of the path following problem for wheeled robots. *Automation and Remote Control*, 74(5):785–801, 2013

Proportional Input-Scaling Controller

- control law for the vehicle to follow a path in stable manner.
- asymptotically stabilize ($d = 0, \theta_e = 0$)

$$v_2 = -v_1 \sum_{i=2}^{n} sgn(v_1)k_i z_i, n = 3 \text{ (differential)}, 4 \text{ (car-like)}$$

Controller for differential (left) and car-like (right) steering.

Implementation

- implemented as a stand-alone C++ library
- encapsulated in a MATLAB/Simulink S-Function block
- integrated into **ROCK** (library and orogen component)

Typical structure of Path Follower using ROCK.

Results for Differential Steering from IMoby Project

Asguard following DFKI outdoor track in IMoby project.

Results for Steerable Wheels from Simulation

MATLAB/Simulink-Adams/View Cosimulation, ISO3888-2 (Double Lane Change Test) standard test track is used as desired path.

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

3.8 ‘Motion Planning for Manipulators’ (MC-P-02)

Behnam Asadi(1), *Sankaranarayanan Natarajan*(2)

(1) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany
(2) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `behnam@informatik.uni-bremen.de`, `sankaranarayanan.natarajan@dfki.de`

Abstract

On this poster, two libraries for motion planning for manipulator is introduced(*MoveIt!*, *MotionPlanner Library*). *MoveIt!* is state of the art motion planner framework for mobile manipulation applications. A framework independent motionplanning library is developed using sample-based motionplanning algorithm. This library also supports dual arm manipulation and replanning for dynamic environment.

**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

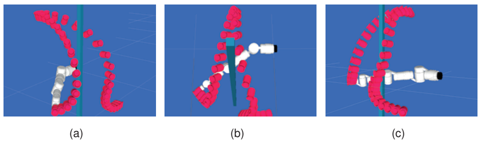
Motion Planning for Manipulators

Behnam Asadi, Sankaranarayanan Natarajan

Sample Based Motion Planning

Motion planning in robotics realm refers to the motions of a robot in a 2D or 3D world occupied with obstacles. Sample based planner are common and widely applied in many robotics tasks and they have the following advantageous:

- Probabilistically completeness.
- Capability of Dealing with high-dimensional *C-space*.
- No need for creating *C-space*

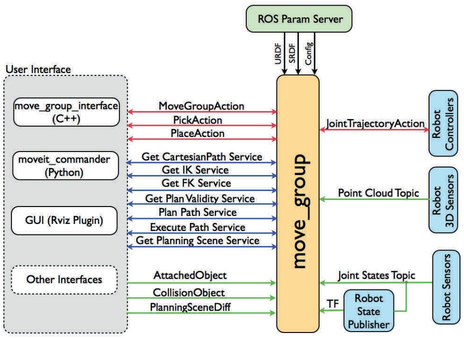

(a) (b) (c)

Schunk arm performing a collision free trajectory.

MoveIt! - A ROS based Motion Planner

MoveIt! is state of the art sampling based motion planner using OMPL [ŞMK12]. It provides an easy-to-use platform for developing manipulation applications.

- Easy to setup with new robots - using URDF
- Integrated Collision Checker(FCL) [PCM12] and Kinematics Solver(KDL, IkFast)
- Environment awareness through 3D perception(PointCloud)
- Kinematics constrained motion planning


ROS Param Server
URDF
SRDF
Config
User Interface
move_group_interface (C++)
moveit_commander (Python)
GUI (Rviz Plugin)
Other Interfaces
MoveGroupAction
PickAction
PlaceAction
Get CartesianPath Service
Get IK Service
Get FK Service
Get Plan Validity Service
Plan Path Service
Execute Path Service
Get Planning Scene Service
AttachedObject
CollisionObject
PlanningSceneDiff
move_group
JointTrajectoryAction
Robot Controllers
Point Cloud Topic
Robot 3D Sensors
Joint States Topic
Robot Sensors
TF
Robot State Publisher

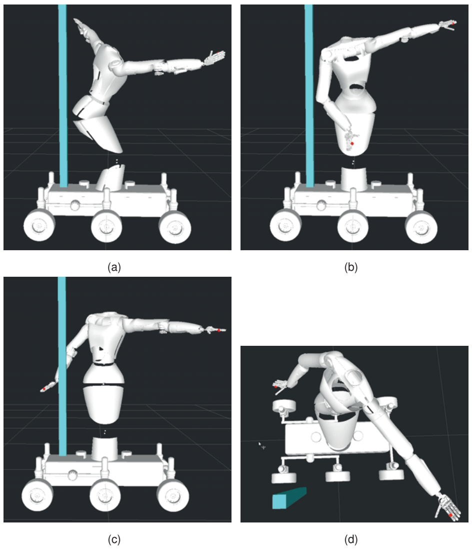
MoveIt Architecture

MoveIt in Rock: The Rock framework can interoperate with ROS at dataflow level. This feature helps to use MoveIt in Rock.

Motion Planner Library

Due to the dependency of the *MoveIt!* motion planner with *ROS* framework, a motionplanner library which is independent of any framework has been developed.

- The library supports sampling based planner such as *RRT*, *SBL*, *EST*, *LBKPIECE*, *BKPIECE*, *KPIECE*, *RRT*, *RRTConnect*, *RRTstar*, *PRM* and *PRMstar*.
- Planning in a dynamic environment.
- The library also supports dual arm manipulation.
- Planning under Kinematics constraints.


(a) (b) (c) (d)

Incorporating Foot, Knees, Hip and Waist into the motion planning.

Next steps

- Integrate the library to Rock framework.
- Including Optimization based planner to the library.
- Representing the environment from the sensor data like PointCloud.

[ŞMK12] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. `http://ompl.kavrakilab.org`.

[PCM12] Jia Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3859–3866, May 2012.

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

3.9 ‘Distributed Dynamics Computation’ (MC-P-03)

Vinzenz Bargsten(1)

(1) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `bargsten@uni-bremen.de`

Abstract

This poster presents a method to compute robot motion dynamics distributedly at joint level. In this method, FPGAs controlling the joint actuators are utilized to compute the recursion steps of the Newton-Euler-Algorithm.

3.9 ‘DISTRIBUTED DYNAMICS COMPUTATION’ – *Vinzenz Bargsten*

**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

Distributed Dynamics Computation

FPGA-based method to compute robot dynamics at joint level

Vinzenz Bargsten

Introduction

- Robot Motion Dynamics
  - relates actuator torques/forces with the resulting motion
  - often highly coupled and non-linear system

⇒ linear controllers require high gains to compensate for unmodelled dynamics

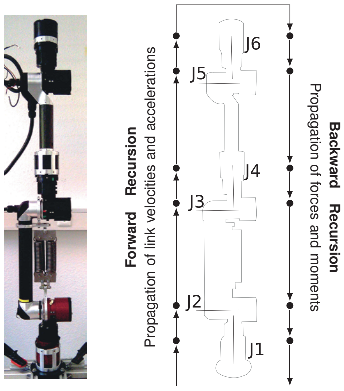
Motivation

- Taking the robot dynamics into account in the control system
  - simplifies control problem
  - allows more *compliant* control schemes instead of stiff position control
- Simulation of motion dynamics

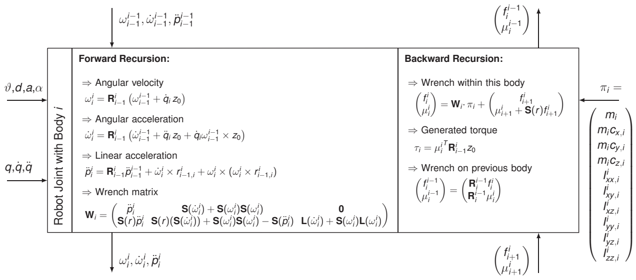
⇒ Require computation of the dynamic robot model, $\tau = f(q, \dot{q}, \ddot{q})$

Novel Approach – Distributed Dynamics Computation

- structure of the recursive Newton-Euler-Algorithm is exploited
  ⇒ distributed computation of the dynamic robot model
- each FPGA controlling an actuator computes a recursion step of the algorithm
- implemented as component programmed in VHDL
- linear dependence on the dynamic parameters is preserved



Left: SpaceBot robot arm. Right: Distributed computation of the recursive Newton-Euler algorithm.



Forward Recursion:

⇒ Angular velocity

$$\omega^i_i = \mathbf{R}^i_{i-1} \left(\omega^{i-1}_{i-1} + \dot{q}_i \, z_0 \right)$$

⇒ Angular acceleration

$$\dot{\omega}^i_i = \mathbf{R}^i_{i-1} \left(\dot{\omega}^{i-1}_{i-1} + \ddot{q}_i \, z_0 + \dot{q}_i \omega^{i-1}_{i-1} \times z_0 \right)$$

⇒ Linear acceleration

$$\ddot{p}^i_i = \mathbf{R}^i_{i-1} \ddot{p}^{i-1}_{i-1} + \dot{\omega}^i_i \times r^i_{i-1,i} + \omega^i_i \times \left(\omega^i_i \times r^i_{i-1,i} \right)$$

⇒ Wrench matrix

$$\mathbf{W}_i = \begin{pmatrix} \ddot{p}^i_i & \mathbf{S}(\dot{\omega}^i_i) + \mathbf{S}(\omega^i_i)\mathbf{S}(\omega^i_i) & \mathbf{0} \\ \mathbf{S}(r)\ddot{p}^i_i & \mathbf{S}(r)(\mathbf{S}(\dot{\omega}^i_i)) + \mathbf{S}(\omega^i_i)\mathbf{S}(\omega^i_i) - \mathbf{S}(\ddot{p}^i_i) & \mathbf{L}(\dot{\omega}^i_i) + \mathbf{S}(\omega^i_i)\mathbf{L}(\omega^i_i) \end{pmatrix}$$

Backward Recursion:

⇒ Wrench within this body

$$\begin{pmatrix} f^i_i \\ \mu^i_i \end{pmatrix} = \mathbf{W}_i \cdot \pi_i + \begin{pmatrix} f^i_{i+1} \\ \mu^i_{i+1} + \mathbf{S}(r) f^i_{i+1} \end{pmatrix}$$

⇒ Generated torque

$$\tau_i = \mu^{i\,T}_i \mathbf{R}^i_{i-1} z_0$$

⇒ Wrench on previous body

$$\begin{pmatrix} f^{i-1}_i \\ \mu^{i-1}_i \end{pmatrix} = \begin{pmatrix} \mathbf{R}^{i-1}_i f^i_i \\ \mathbf{R}^{i-1}_i \mu^i_i \end{pmatrix}$$

$$\pi_i = \begin{pmatrix} m_i \\ m_i c_{x,i} \\ m_i c_{y,i} \\ m_i c_{z,i} \\ I^i_{xx,i} \\ I^i_{xy,i} \\ I^i_{xz,i} \\ I^i_{yy,i} \\ I^i_{yz,i} \\ I^i_{zz,i} \end{pmatrix}$$

The computations carried out on each FPGA controlling a robot joint.

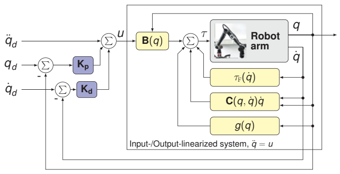
Distributed Computation

- Each robot joint communicates with its neighbours
- *Forward Recursion* starts with the first link, incorporating base velocity and acceleration including gravity
- According to joint motion and link geometry, the current link’s velocity and acceleration is propagated to the next one
- The last node initiates the *Backward Recursion*: the vector of forces and moments is propagated back to the previous node. Projection on the rotation axis yields the torque τ at the joint

Application Example: Arm Computed-Torque Control

In Computed-Torque-Control, the use of a dynamic model:

- decouples the motion dynamics with the inertia matrix $\mathbf{B}(q)$
- compensates for non-linear effects from gravity $g(q)$, friction $\tau_F(\dot{q})$, centrifugal and Coriolis forces $\mathbf{C}(q, \dot{q})\dot{q}$
- trajectory tracking is possible with much lower feed-back gains
  ⇒ basis for compliant motions
- contact forces at the end-effector can be estimated through the Jacobian



Structure of Computed-Torque-Control

Future Aspects

- Online adaption of dynamic parameters
- Experiments with different model-based control schemes
- Hybrid control of contact forces
- Online model adaption to changes in the link structure

References

- Bruno Siciliano, Lorenzo Sciavicco, and Luigi Villani. *Robotics: Modelling, Planning and Control (Advanced Textbooks in Control and Signal Processing).* Springer, Berlin, 2008

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

3.10 'Complex Numbers and Quaternions – A unified view on representations and metrics for rotations' (MC-P-04)

Bertold Bongardt(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `bertold.bongardt@dfki.de`

Abstract

On this poster, a novel view on complex numbers and quaternions is motivated by introducing a five-dimensional complex space which is defined as the union of the complex plane $\mathbb{C}$ with the quaternion space $\mathbb{H}$. For rotations with a fixed rotation axis, the complex 5-space can be visualized by $\mathbb{R}^3$ and by $\mathbb{R}^2$. In these visualizations, the representations of a rotation via a complex number, quaternions, and a rotation matrix appear in an elementary-geometric setup generalizing the unit circle. The definition of the complex 5-space is based on an explicit distinction of four different imaginary units. The poster illustrates one usage of these novel concepts with a comparison of distance measures for rotational displacements.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Complex Numbers and Quaternions

A unified view on representations and metrics for rotations

Bertold Bongardt

Introduction.

In mechanics, complex numbers and quaternions are used to encode rotations: Complex numbers for planar and quaternions for spatial rotations. How are both related – algebraically and geometrically?

A Precise Algebraic Notation.

A complex number is defined as $z = x + \mathrm{i} \cdot y$ with $\mathrm{i}^2 = -1$.
A quaternion, with $\boldsymbol{q} = (q_1, q_2, q_3)^T$ and $\mathbf{j} = (\mathrm{j}, \mathrm{k}, \mathrm{l})^T$, is:

$$q = q_0 + \mathrm{j} \cdot q_1 + \mathrm{k} \cdot q_2 + \mathrm{l} \cdot q_3 = q_0 + \mathbf{j} * \boldsymbol{q}$$

For the four *distinct* units i and j, k, l with

$$\mathrm{i} \neq \mathrm{j} \neq \mathrm{k} \neq \mathrm{l}\,, \tag{1}$$

the squares equal to $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{l}^2 = \mathrm{j} \cdot \mathrm{k} \cdot \mathrm{l} = -1$.

Planar Rotations with Complex Numbers and (2×2)**-Matrices.**

For a given angle, $\phi \in (-\pi, \pi]$, the corresponding unit complex number $\hat{z}$ is determined via the exponential map (*Euler's formula*) as

$$\hat{z} = \exp(\mathrm{i} \cdot \phi) = x + \mathrm{i} \cdot y = \cos\phi + \mathrm{i} \cdot \sin\phi\,, \tag{2}$$

the planar rotation matrix $\boldsymbol{R}_2 = \boldsymbol{R}_2(\phi)$ is determined as

$$\boldsymbol{R}_2 = \exp(\mathrm{I} \cdot \phi) = \boldsymbol{X} + \mathrm{I} \cdot \boldsymbol{Y} = \cos\phi \cdot \boldsymbol{I}_2 + \mathrm{I} \cdot \sin\phi \cdot \boldsymbol{I}_2 = \left(\begin{smallmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{smallmatrix}\right)\,, \tag{3}$$

where $\boldsymbol{X} = x \cdot \boldsymbol{I}_2$ and $\boldsymbol{Y} = y \cdot \boldsymbol{I}_2$ for $x = \cos\phi$ and $y = \sin\phi$. Matrix I is defined as $\mathrm{I} = \left(\begin{smallmatrix} 0 & -1 \\ +1 & 0 \end{smallmatrix}\right)$, with $\mathrm{I}^2 = -\boldsymbol{I}_2$ and analogue of i in Equation 2.

Spatial Rotations with Quaternions and (3×3)**-Matrices.**

For a rotation, given by an angle $\phi \in (-\pi, \pi]$ and an axis $\hat{\boldsymbol{\omega}} \in \mathcal{S}^2$, a corresponding unit quaternion $\hat{q}$ is determined via the equation [1]

$$\hat{q} = \exp(\tfrac{\phi}{2} \cdot \mathbf{j} * \hat{\boldsymbol{\omega}}) = q_0 + \mathbf{j} * \boldsymbol{q} = \cos\frac{\phi}{2} + \sin\frac{\phi}{2} \cdot \mathbf{j} * \hat{\boldsymbol{\omega}}\,, \tag{4}$$

the (3×3)-rotation matrix $\boldsymbol{R} = \boldsymbol{R}(\phi; \hat{\boldsymbol{\omega}})$ is determined as

$$\boldsymbol{R} = \exp(\phi \cdot \hat{\boldsymbol{\omega}}^{\otimes}) = \cos\phi \cdot (-(\hat{\boldsymbol{\omega}}^{\otimes})^2) + \sin\phi \cdot \hat{\boldsymbol{\omega}}^{\otimes} + \hat{\boldsymbol{\omega}}^{\boxslash}\,. \tag{5}$$

Complex 5-Space and Complex 3-Space.

Define a complex 5-space (see Equation 1):

$$\mathbb{R} \times \mathrm{i}\,\mathbb{R} \times \mathrm{j}\,\mathbb{R} \times \mathrm{k}\,\mathbb{R} \times \mathrm{l}\,\mathbb{R} \cong \mathbb{R}^5 \cong \mathbb{C} \cup \mathbb{H}$$

Define a complex 3-space (‘unit’ $\hat{\mathrm{j}} := \mathbf{j} * \hat{\boldsymbol{\omega}}$ and ‘plane’ $\hat{\mathbb{C}} = \mathbb{R} \times \hat{\mathrm{j}}\,\mathbb{R}$):

$$\mathbb{R} \times \mathrm{i}\,\mathbb{R} \times \hat{\mathrm{j}}\,\mathbb{R} \cong \mathbb{R}^3 \cong \mathbb{C} \cup \hat{\mathbb{C}}$$

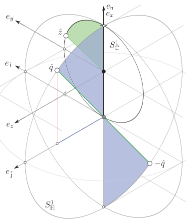


Fig. 1: Combined 3-space visualization of unit complex number and unit quaternion.

Extended Unit Circle Geometry.

To planarize the view of Figure 1, ‘fold’ the two vertical coordinate planes in Figure 1 so that they conincide, see Figure 2.

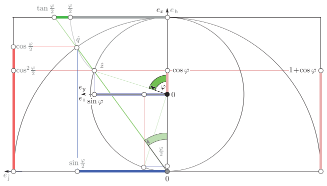


Fig. 2: Combined planar geometry of unit complex number $\hat{z}$ and unit quaternion $\hat{q}$.

Application: Relations of Rotation Metrics.

Metrics for rotations and displacements are used in several applications in mechanics. Recent works are, for example, [2, 3, 4].

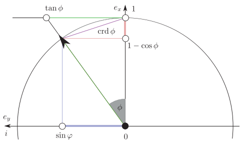


Fig. 3: Approximations of an angle ϕ.

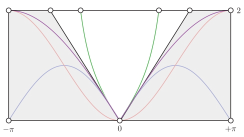


Fig. 4: Functions $f_i(\phi)$ of an angle ϕ.

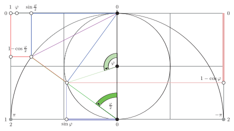


Fig. 5: Rotational displacement, $\phi = \frac{108^\circ}{360^\circ} \cdot 2\pi$.

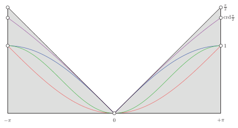


Fig. 6: Graphs of five distance functions.

Overview of Rotation Metrics.

With previous visualizations (Figures 3 – 6), five types of rotation metrics are observed ‘geometrically’, see Table 1.

	Planar Rotations	Spatial Rotations	
Metric Type	Complex Numbers	Quaternions	(3×3)-Matrices
Geodesic	$\lvert\log(\hat{z}_A^{-1} \cdot \hat{z}_B)\rvert = \lvert\phi\rvert$	$\lVert\log(\hat{q}_A^{-1} \cdot \hat{q}_B)\rVert_2 = \lvert\frac{\phi}{2}\rvert$	$\lVert\log(\boldsymbol{R}_A^{-1} \cdot \boldsymbol{R}_B)\rVert_F = \sqrt{2} \cdot \lvert\phi\rvert$
Chordal	$\lVert\hat{z}_A - \hat{z}_B\rVert_2 = \mathrm{crd}\,\phi$	$\lVert\hat{q}_A - \hat{q}_B\rVert_2 = \mathrm{crd}\,\frac{\phi}{2}$	$\lVert\boldsymbol{R}_A - \boldsymbol{R}_B\rVert_F = \sqrt{2} \cdot \mathrm{crd}\,\phi$
Versine	$1 - \hat{z}_A * \hat{z}_B = 1 - \cos\phi$	$1 - \hat{q}_A * \hat{q}_B = 1 - \cos\frac{\phi}{2}$	$1 - \frac{\boldsymbol{R}_A * \boldsymbol{R}_B}{\lVert\boldsymbol{R}_A\rVert_F \lVert\boldsymbol{R}_A\rVert_F} = \frac{2}{3} \cdot (1 - \cos\phi)$
Sine		$\lVert\,\mathbf{Im}\,(\hat{q}_A^{-1} \cdot \hat{q}_B)\rVert_2 = \sin\lvert\frac{\phi}{2}\rvert$	
Squared		$\lVert\,\mathbf{Im}\,(\hat{q}_A^{-1} \cdot \hat{q}_B)\rVert_2^2 = \sin^2(\frac{\phi}{2})$	

Tab. 1: An overview of the five considered classes of distance functions.

Conclusion.

The notation with four distinct imaginary units allows to unify the view of complex numbers and quaternions. Using this, rotation metrics are studied in analogy observations and two novel concepts are revealed.

References.

[1] Andrew J. Hanson. *Visualizing Quaternions.* Morgan Kaufmann, 2006.

[2] Inna Sharf, Alon Wolf, and M. B. Rubin. Arithmetic and geometric solutions for average rigid-body rotation. *Mechanism and Machine Theory*, 45(9):1239–1251, 2010.

[3] Du Q. Huynh. Metrics for 3D Rotations: Comparison and Analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[4] Qiao Lin and Joel W. Burdick. Objective and Frame-Invariant Kinematic Metric Functions for Rigid Bodies. *J. of Robotics Research*, 19:612–625, 2000.

SPONSORED BY THE
Federal Ministry
of Education
and Research

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

3.11 ‘iTaSC: Application and Rock Integration – Specification and Control of Complex Sensor-Based Tasks’ (MC-P-05)

Dennis Mronga (1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `dennis.mronga@dfki.de`

Abstract

On this poster an introduction to the iTaSC Software (**i**nstantaneous **Ta**sk **S**pecification using **C**onstraints) is given. ITaSC helps the user to specify complex sensor-based robot tasks by separating them into more easily defined sub-problems, which are the constraints to an hierarchical optimization problem.

The poster illustrates use-cases, mathematical background and experimental results, as well as the integration of iTaSC in the Rock framework.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

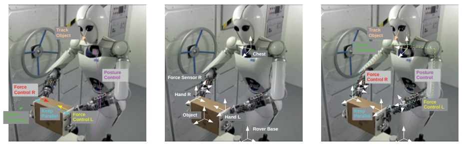
iTaSC: Application and Rock Integration

Specification and Control of Complex Sensor-Based Tasks

Dennis Mronga

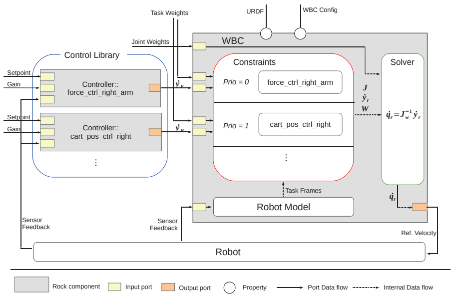
What is iTaSC?

- iTaSC - **i**nstantaneous **Ta**sk **S**pecification using **C**onstraints [1, 2]
- Can be used to describe *reactive* robot control tasks by specifying constraints between robot frames and the environment
- Automatic computation of the control solution
- Prioritization and weighting between constraints



(a) Specification of the motion constraints (b) Involved Frames (c) Kinematic chains that are controlled by each constraint

Example: Moving an object with two arms along a pre-defined trajectory



Control Library and WBC Component in Rock

Establishing Constraint Hierarchies

Given: A proportional controller $\dot{\mathbf{y}}_r = \mathbf{k}_p \cdot (\mathbf{x}_r - \mathbf{x})$ with gain $\mathbf{k}_p$ that controls the pose between two robot frames. A solution in joint space $\dot{\mathbf{q}}_r$ that regulates the *constraint velocity* $\dot{\mathbf{y}}_r$ to zero is:

$$\dot{\mathbf{q}}_r = \mathbf{J}_W^+ \cdot \dot{\mathbf{y}}_r \tag{1}$$

$\mathbf{J}, \mathbf{J}_W^+$ - Constraint Jacobian, Weighted LS Inverse[1]of **J**
$\mathbf{x}, \mathbf{x}_r$ - Actual and reference pose

[1] to improve readability, the index *W* is omitted in the following lines

In the redundant case, a secondary constraint ξ can be optimized in the Nullspace of the primary one:

$$\dot{\mathbf{q}}_r = \mathbf{J}^+ \cdot \dot{\mathbf{y}}_r + (\mathbf{I} - \mathbf{J}^+\mathbf{J}) \cdot \xi \tag{2}$$

Iterative extension to *N* constraints

$$\dot{\mathbf{q}}_{r,i} = \dot{\mathbf{q}}_{r,i-1} + \mathbf{J}_i^+ \cdot \mathbf{P}_i \cdot (\dot{\mathbf{y}}_{r,i} - \mathbf{J}_i \cdot \dot{\mathbf{q}}_{r,i-1}) \tag{3}$$

$$\dot{\mathbf{q}}_{r,i-1} = \mathbf{0}, \quad i = 0 \tag{4}$$

$$\mathbf{P}_i = \mathbf{P}_{i-1} - (\mathbf{J}_{i-1}\mathbf{P}_{i-1})^+(\mathbf{J}_{i-1}\mathbf{P}_{i-1}), \quad \mathbf{P}_0 = \mathbf{I} \tag{5}$$

$\mathbf{J}_i$ - Jacobian of the i-th priority level
$\dot{\mathbf{y}}_{r,i}$ - Constraint velocity of the i-th priority level
$\mathbf{P}_i$ - Nullspace projector of priority level i
$\dot{\mathbf{q}}_{r,i}$ - Overall solution on priority level *i*

Integration in Rock

- Whole body control component (WBC):
  - Configuration via URDF (robot model) and a WBC configuration (constraints, priorities, task frames, ...)
  - Dynamically creates port interfaces for each constraint
  - Sorts constraints by priority, creates equation system
  - Solver implements the iTaSC algorithm
- Control Library: Collection of generic controllers that provide the input for the WBC component

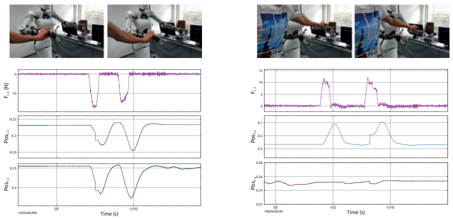
[1] Joris De Schutter et al.: *Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty.* The International Journal of Robotics Research, 2007.

[2] http://www.orocos.org/wiki/orocos/itasc-wiki

Experimental Results

Name	Constraint Description	TF Root	TF Tip	Priority
Zero Force R	Maintain zero force on right wrist	Chest	Force Sensor R	0
Zero Force L	Maintain zero force on left wrist	Chest	Force Sensor L	0
Hold Pose R	Hold the pose of the right wrist	Rover Base	Hand R	1
Keep Parallel	Keep left & right wrist parallel	Hand R	Hand L	2

Constraints used in the experiment, ordered by priority (0 = highest)



(a) External force acting on the right arm: The right arm pose is shifted since the Null Force constraints have a higher priority than the Hold Pose R constraint

(b) External force acting on the left arm: The right arm is stationary (bottom plot), as the priority of the Hold Pose R constraint is higher than the one of the Keep Parallel constraint

Experiment on the robot AILA: An external force is applied and measured using wrist-mounted force-torque sensors.

Conclusion

- iTaSC allows specification of complex sensor-based tasks
- Easier configuration and monitoring of the constraints in Rock

Next Steps

- Execution of task sequences with context-specific adaptation of task hierarchies, weights and controller configurations
- Extension to torque controlled motions
- Connection of sensor-based reactions to global motion planning

Universität Bremen

Contact:
DFKI Bremen & University of Bremen
Robotics Innovation Center

Director: Prof. Dr. Frank Kirchner
E-mail: robotics@dfki.de
Website: www.dfki.de/robotics

German Research Center for Artificial Intelligence (DFKI) GmbH

DFKI Bremen
Robert-Hooke-Straße 1
28359 Bremen
Germany
Phone: +49 421 178 45 0
Fax: +49 421 178 45 4150

DFKI Saarbrücken
Stuhlsatzenhausweg 3
Campus D3 2
66123 Saarbrücken
Germany
Phone: +49 681 875 75 0
Fax: +49 681 857 75 5341

DFKI Kaiserslautern
Trippstadter Straße 122
67608 Kaiserslautern
Germany
Phone: +49 631 205 75 0
Fax: +49 631 205 75 5030

DFKI Projektbüro Berlin
Alt-Moabit 91c
10559 Berlin
Germany
Phone: +49 30 238 95 0

E-mail:
reports@dfki.de

Further information:
http://www.dfki.de