Experience-Based Adaptation of Locomotion Behaviors for Kinematically Complex Robots in Unstructured Terrain

Alexander Dettmann[1], Anna Born[1], Sebastian Bartsch[2], and Frank Kirchner[1,2]

Abstract— **Kinematically complex robots such as legged robots provide a large degree of mobility and flexibility, but demand a sophisticated motion control, which has more tunable parameters than a general planning and decision layer should take into consideration. A lot of parameterizations exist which produce locomotion behaviors that fulfill the desired action but with varying performance, e.g., stability or efficiency. In addition, the performance of a locomotion behavior at any given time is highly depending on the current environmental context. Consequently, a complex mapping is required that closes the gap between robot-independent actions and robot-specific control parameters considering the environmental context and a given prioritization of performance indices.**

In the proposed approach, the robot learns from experiences made during its interaction with the environment. A knowledge base is created which links locomotion behaviors with performance features for visited contexts. This *behavior library* is utilized by a case-based reasoner to select motion control parameters for a desired action within the current context. The paper provides an overview of the control approach, the algorithms used to determine the current context and the robot's performance, as well as a description of the reasoner which selects appropriate locomotion behaviors. In experiments, different *behavior libraries* were automatically built when operators had to control a walking robot manually through obstacle courses. Afterwards, the collected experiences and a trajectory follower were used to traverse an obstacle course autonomously. The provided experimental evaluation shows the performance dependency of the autonomous control with respect to different sizes and qualities of utilized *behavior libraries* and compares it to manual control.

I. Introduction

In common control architectures for robotic systems, e.g., [1], [2], hierarchical layers exist which have differences in their representation form, response time, predicting capabilities, and dependence on accurate world models [3]. The purpose of the lowest layer (action layer) is to generate motion commands for the actuators on basis of desired actions and processed sensor data. Since short response time is crucial to act in real world scenarios, reactive control paradigms are often used. An action, in this sense, is a command with associated attributes generated by the layer on top (task layer) where mission-specific tasks are decomposed into a sequence of actions often realized through deliberative planning. Since these actions are usually robot-independent, the reactive action layer needs to decompose each action into robot-specific control parameters resulting in a specific robot behavior. When using simple wheel driven systems, a simple mapping is needed, e.g., mapping a desired robot velocity to a desired actuator speed. But kinematically complex robots provide manifold possibilities to execute the same action in different ways by tuning numerous control parameters. In addition, the required mapping is highly depending on the current environmental situation which varies heavily in the field of locomotion in unstructured terrain.

One way to adapt the control parameters autonomously is to equip each behavior-producing module in the reactive layer with an intelligent mechanism that determines the module's relevance and reliability and adapts its influence accordingly [4], [5]. Other approaches try to build a causal model, which maps actions to control parameters. Since most applications are too complex to create models based on physical analysis, rules derived from expert knowledge [6], [7] or learned mappings [8], [9] are used. The former need deep domain knowledge which is not desired for developing a general motion control interface. The latter need a lot of training data to derive a policy which covers a broad spectrum of the possible problem space prior to execution time. Integrating new experiences during execution in real-time is rather complicated [10]. Instance-based regression techniques, e.g., case-based reasoning [11] which solves problems by reusing experiences from similar, previously solved problems, have the advantage that they can work on a sparse knowledge base and that new experiences can directly be integrated.

The idea in the proposed approach is to build up a knowledge base which holds all relevant robot-dependent information about known behaviors and their performance in various state contexts, which are environmental conditions in the target application of locomotion. This, so called *behavior library* (BL), can then be utilized by a behavior configurator that maps scenario-specific actions to known parameterizations of the motion control, thus generating different behaviors (Fig. 1). Since all robot-dependent information are implicitly stored in the BL, a general robot-independent mapper can be implemented as interface to the action generating task layer encapsulating the robot-specific motion control.

In the proposed approach, the robot learns from experiences made during its interaction with the environment. Therefore, the robot needs to continuously self-evaluate its actions. Section II briefly describes which performance features are necessary to characterize locomotion behaviors. Furthermore, the robot has to determine the significant fea-

[1]The author is with the Faculty of Mathematics and Computer Science, University of Bremen, 28359 Bremen, Germany firstname.lastname@uni-bremen.de

[2]The author is with the German Research Center for Artificial Intelligence - Robotics Innovation Center (DFKI RIC), 28359 Bremen, Germany firstname.lastname@dfki.de

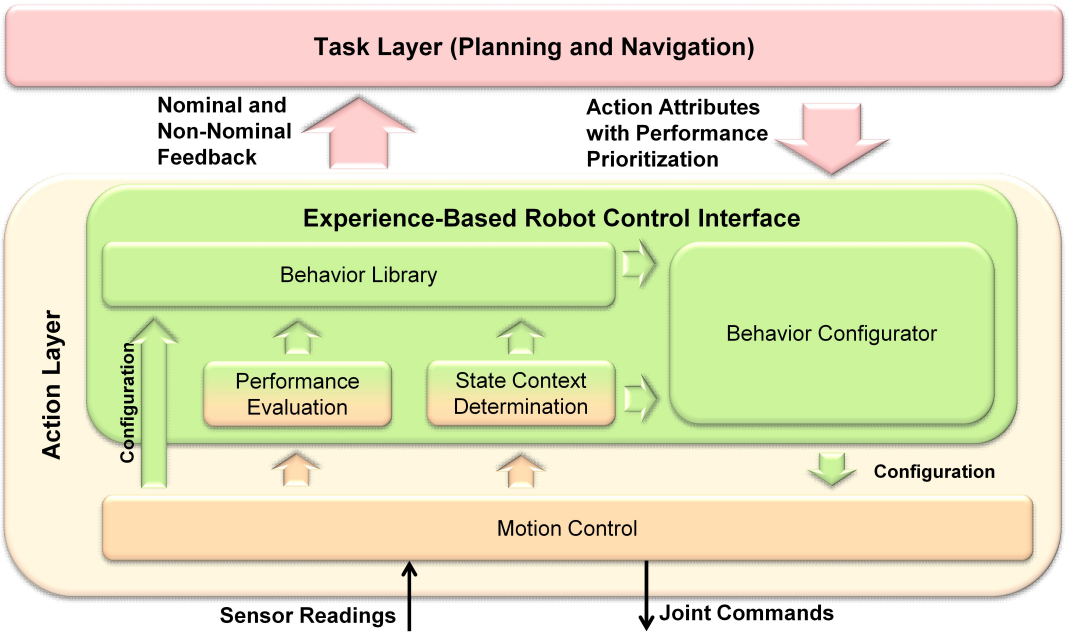


Fig. 1. Experience-based interface (green) which extends the motion control in the action layer (orange) to allow scenario-specific commands and to generate additional information for higher layers. Experiences, i.e. performance information of behaviors in various state contexts, are stored in a *behavior library* and are utilized by a behavior configurator to map scenario-specific actions to robot-specific control parameters of the motion control. The estimation of performance and state context features is based on the robot's processed sensor values and may already be part of the motion control itself.

tures of the currently investigated environment. A selection of these, so called state context features, is provided in Section III. Since especially at the beginning of a robot mission, only sparse domain knowledge and a few experiences are available, case-based reasoning is used to infer the best-suited configuration of the motion control for a certain context which is described in Section IV. The experiments in Section V show the performance of the proposed approach by an example of controlling the six-legged robot SpaceClimber [12] through a demanding obstacle course. In the last section, a conclusion is drawn and an outlook is provided.

II. Performance Evaluation

In the proposed experience-based control approach, evaluating the robot's performance during runtime is one key issue. For a meaningful performance metric, features are needed which build upon processed sensor values to characterize a robot's behavior and to allow a quantitative comparison. Here, a distinction was made between features which characterize the execution of the desired action, e.g., motion and posture commands, and features which deliver additional meta information, e.g., stability, energy efficiency, or undesired vibrations of the body. The former have variable and action-dependent target values whereas the latter always have a desired optimum. In the following, the performance features used to characterize locomotion behaviors of SpaceClimber are briefly introduced.

A. Action Performance Features

Most kinematically complex locomotion systems support longitudinal (*velocity x*) and lateral movement (*velocity y*) as well as turning motions around its geometrical center (*turn rate*). These features need to be known very precisely because they are also the basis for other algorithms like map building. Thus, it is always suggested to use accurate external reference systems, e.g, GPS or motion tracking systems. But since many real world applications do not support their usage, odometry information needs to be used. SpaceClimber uses a contact point odometry which takes the position of each foot during ground contact as input and averages their changes to the overall robot position difference. The orientation change is measured by an inertia measurement unit (IMU). For a more accurate velocity estimation, more sophisticated motion models [13], complete slam approaches [14], or fusion of visual odometry can be applied.

Besides motion commands, in some applications the operator needs to influence the robot's posture as well, e.g., to fit through doorframes or to enter flat confined spaces. Thus, *body width* and *body height* are estimated as well. Here, *body width* is defined as the largest lateral distance between any foot of the left and right side. The performance feature *body height* is the inverted average of all vertical foot positions with respect to the robot's coordinate system.

B. Meta Performance Features

Since kinematically complex robots have numerous possibilities to perform actions in various ways, additional criteria to compare them are required. Locomotion behaviors can be

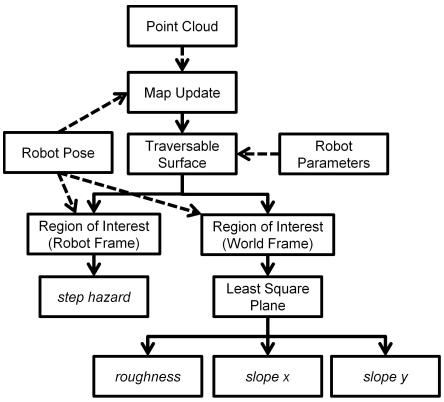


Fig. 2. Flow diagram of the procedure to determine state context features (*italic*) on the basis of visual data and the current robot pose

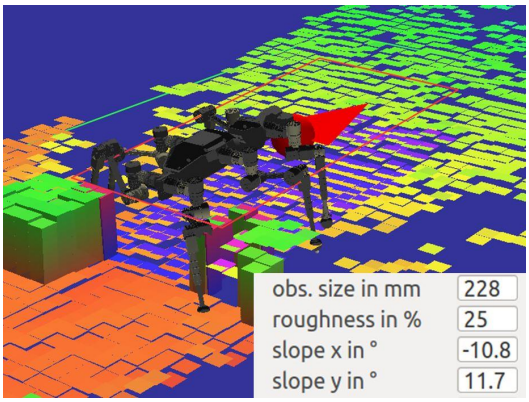


Fig. 3. Visualization of generated map, the region of interest (violett area beneath red rectangle), and estimated state context features

characterized by their stability or efficiency, both expressible by several features.

The static stability margin (*ssm*) [15] and force-angle stability margin (or dynamic stability angle, *dsa*) [16] are common stability metrics. The former measures the stability as a minimal distance between the projected center of gravity and the edges of the support polygon, which is defined as convex hull formed by the robot's footprints. This method was chosen for its simplicity and lower computational cost. The latter defines the stability as minimal angle between gravity vector and the vector from the robot's zero moment point to the closest edge of the support polygon. This approach takes, in comparison to *ssm*, dynamic effects and the robot height into consideration.

The overall *power* needed by all joints as well as all sensor and processing units is a feature that characterizes a behaviors efficiency. For locomotion, energy per distance (*epd*) is common, which multiplies the *power* with the processing period and divides it by the position change. The latter is computed by the average Euclidean distance each foot moves during ground contact. In this way, turning movements are also incorporated. The sum of the roll and pitch angular rotation speeds is used to measure the undesired *body vibration*.

III. Environmental Context

For safe and efficient locomotion in unstructured terrain, the robot should frequently adapt its motion control parameters to the current environmental context. Therefore, the environment should be characterized through meaningful features. In most common approaches, step hazards, inclination, and structure of the terrain are taken into consideration. In this section, an overview of determining these terrain characteristics on basis of point cloud data is provided (Fig. 2).

At first, a map is continuously updated by synchronized pose information and point cloud data. The environment is represented in form of a multi-level surface map [17], which has the possibility to handle overhanging and vertical objects, e.g., the representation of confined spaces. Fitting to the foot size of the robot, the size of a single cell patch was set to 8 cm x 8 cm which represents the mean of all point cloud data in this area. By considering the height of the robot and its maximum step height, the traversable surface is built from the environment map, which excludes all patches unaccessible to the robot. The relevant part of the environment, which should have an influence on the current locomotion behavior, is the area beneath the robot and the region in direction of movement during the next step cycle. For this region of interest (ROI), the environmental features are determined (Fig. 3).

A. Step Hazard

The state context feature *step hazard* represents the highest height difference between cells with respect to the robot. Therefore, the ROI is transformed to the robot's coordinate system making the step hazard independent from the ground inclination. Furthermore, the sign of the height difference is a valuable information. Positive steps should have more influence on the locomotion behavior because they can block the intended motion. So, a positive *step hazard* above a minimum threshold min_diff is preferred to any negative *step hazard*. The entire algorithm can be described as followed:

for each patch in ROI **do**
  $n\_patch = getNeighbor(patch, motion\_direction)$
  $diff = getHeight(n\_patch) - getHeight(patch)$
  if $|diff| < min\_diff$ **then**
    $diff = 0$
  end if
  if $diff > 0$ **then**
    $step\_hazard = max(diff, step\_hazard)$
  else
    if $step\_hazard < 0$ **then**
      $step\_hazard = min(diff, step\_hazard)$
    end if
  end if
end for

B. Inclination and Terrain Structure

To compute the these terrain characteristics, a least-square plane fitting algorithm is applied to the ROI [18], which is

independent from the terrain inclination and the robot orientation. To determine the inclination of the surface relative to the gravity vector, but in the longitudinal and lateral direction of the robot, only the rotation of the robot around the gravity vector is applied onto the ROI.

The context features *slope x* and *slope y* are used to represent the plane inclination in longitudinal and lateral direction of movement, respectively. Therefore, the angles between x and y component of the plane's normal vector and the gravity vector are calculated.

The state context feature *roughness* is computed by the standard deviation of cell patches along the plane's normal vector. It is normalized to the maximum possible step height of the robot to obtain a robot-meaningful coefficient.

IV. Behavior Adaptation

In the proposed experience-based control approach, the mapping of desired actions to locomotion behaviors is based on the current context and the experiences stored in the BL. This section provides an overview of the structure of this knowledge base and its utilization to select locomotion behaviors.

A. Behavior Library Structure and Management

The BL consists of behaviors which are defined through a motion control algorithm and parameter sets [19], evaluated contexts, and *behavior evaluations*. Each of the latter holds a reference to one behavior and E context evaluations which represent the experience stored in the BL (Fig. 4). Simulated data, e.g., produced by behavior learning algorithms, or real experiences from everyday usage can be stored in the BL. The type of experience needs to be indicated by a setup identifier, because these data must not be merged. Simulated experiences might be needed in certain situations, but after having real experiences collected for the same context, they loose importance due to the simulation reality gap inaccuracy. Information about the evaluated state context is mandatory as well, which is described by a list of I state context features $(s_1^{cur}, ..., s_I^{cur})$. Each of them is discretized according to a robot-specific value range to limit the possible number of state contexts which are then referenced in *behavior evaluations*. A list of J performance features $(p_1^{cur}, ..., p_J^{cur})$, each characterized by a mean value, a mean of its squared value (needed to calculate the standard deviation of the feature over the number of evaluations), a mean of the standard deviation of one step cycle, and a unit. The number of evaluations indicates the reliability of a context evaluation. It is also required when new experiences arrive and the mean values of an existing context evaluation need to be updated. Experiences are stored whenever a behavior was continuously executed through a whole step cycle. Only then, context and performance features can be averaged over this step cycle, matched to the current behavior, and finally stored in the BL (Fig. 5).

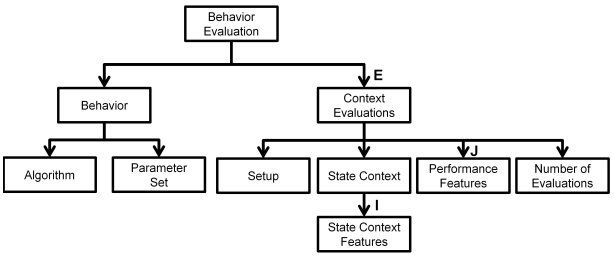


Fig. 4. Content of *behavior evaluations*

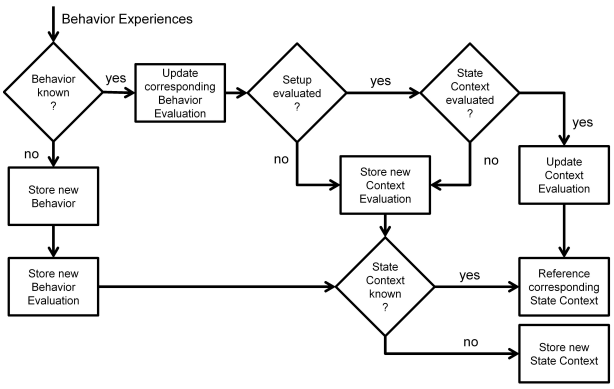


Fig. 5. Procedure to store new experiences in the *behavior library*

B. Behavior Configurator

Case-based reasoning is used for the real-time adaptation of the motion control. A *case* is represented by a *behavior evaluation* of the BL. The input problem can be described by two feature vectors representing the current environmental state with I state context features and the current desired action with J performance features, where each feature is a tuple consisting of value (s_i^{cur}, p_j^{cur}) and weight (w_i^S, w_j^P). The J performance features consist of A variable features describing the desired action and M features describing the meta information which are constant at their optimal values.

To identify the most suitable case, the proposed selection algorithm calculates for each *case* in the BL the similarity to the input query. To be able to compare feature values, every feature value is normalized according to robot-relevant limits. In the first step, the state similarity Sim_e^{State} between the current state features s_i^{cur} and stored reference state features $s_i^{ref,e}$ is calculated for each context evaluation e of each case (1). The weighted mean squared error is used, as it is more sensitive to large differences of one single feature than to small differences of several features compared to the weighted mean absolute error.

$$Sim_e^{State} = 1 - \frac{\sum_{i=1}^{I}(s_i^{cur} - s_i^{ref,e})^2 \cdot w_i^S}{\sum_{i=1}^{I} w_i^S} \qquad (1)$$

The weights for each feature variable w_i^S can be used to include the confidence or importance of the corresponding state context feature estimation. Finally, the state similarity for the entire case Sim^{State} is represented by the maximum state similarity of the E evaluations (2).

$$Sim^{State} = \max_{e \in E}(Sim_e^{State}) \qquad (2)$$

To ensure that only the most relevant performance evaluation is taken into consideration, the action similarity is calculated for the evaluation e_{max} with the highest state similarity (3), also using the weighted mean squared error (4),

$$e_{max} = \underset{e \in E}{\operatorname{argmax}}(Sim_e^{State}) \tag{3}$$

$$Sim^{Action} = 1 - \frac{\sum_{j=1}^{J}(p_j^{cur} - p_j^{ref,e_{max}})^2 \cdot w_j^P}{\sum_{j=1}^{J} w_j^P} \tag{4}$$

where w_j^P are the weights of a performance ratio which the operator or higher layers can define to influence the robot behavior. Finally, the overall similarity Sim of a case is the product of both single similarities (5).

$$Sim = Sim^{State} \cdot Sim^{Action} \tag{5}$$

After determining the similarity of each case, the behavior of the most similar case, which yields the probably best locomotion behavior, is applied.

V. Experimental Evaluation

To analyze the application of the proposed approach, a suitable BL is needed. In the first experiment, SpaceClimber was controlled manually through two different obstacle courses by setting its control parameters (22 were available). The generated locomotion behaviors and their performance in visited state contexts were recorded to fill BLs. It was evaluated how BLs evolve to analyze their maturity. In the second experiment, the generated BLs were used by the proposed behavior configurator and combined with a trajectory follower to generate motion commands to traverse an obstacle course autonomously.

Both experiments were conducted in simulation[1] for several reasons. First, operators of different skill level participated to increase the diversity of chosen locomotion behaviors. But especially unskilled operators can choose fatal parameterizations which could damage the robot. Second, complex test tracks with reproducible conditions could be generated which allow a high diversity of contexts. And third, the operators had best conditions to control the robot due to an experiment setup including a three-dimensional view on the robot in real-size from different perspectives as well as two touch screen monitors to visualize the robot's telemetry data and to conveniently set the control parameters.

The operators were evaluated to compare their skill level, to select appropriate operators for building BLs, to force the exploration of behaviors, and to compare their performance to autonomous control.

- Position Accuracy: A path was provided to guide the operator through the obstacle course. The average distance to the path was recorded, whereas a distance of more than 0.5 m (approx. half of the robot width) resulted in a score of 0% and a distance of less than 0.1 m in 100%. The small corridor for a full score was introduced to avoid continuous motion corrections allowing more behavior evaluations.
- Energy Efficiency: *epd* was used, whereas a value of more than 1.4 Wh/m resulted in a score of 0% and a value of less than 0.14 Wh/m in 100%, which corresponds to an average power consumption of 100 W (power needed to hold up the body and to power the senors and processing units) at a speed of 1/5th of the body length per second or 1/50th, respectively.
- Stability: The average *dsa* was used as metric whereas a positive angle scored up to the limit of 45° (100%) whereas instability (0°) returned no score.

Before conducting the actual experiment, each operator had to overcome a replica of the 2013th DARPA robotics challenge mobility test track[2] to analyze the operator's skills in controlling SpaceClimber. Only operators which completed the task with a overall score above 25% were selected for building a BL, since it was intended to apply the experience-based control approach on the real system in a third experiment.

A. Generating Behavior Libraries

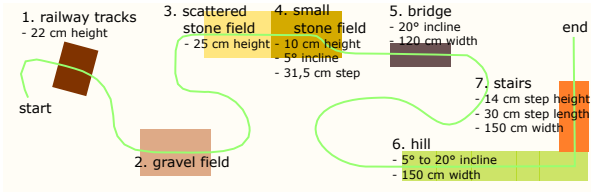
To build up BLs, two different obstacle courses were generated (Fig. 6). The *outdoor* test track includes obstacles commonly found outside whereas the *indoor* test track incorporates obstacles which can be recreated with basic elements. Both obstacle courses require many adaptations of the motion control to realize various translational and rotational movements and to pass one kind of obstacle without failing another.

Each obstacle course was traversed by five operators of comparable skill levels. During each experiment, a BL was built from scratch by recording the chosen locomotion behaviors, the average context, and the resulting behavior performance during each step cycle of constant parametrization. Fig. 7 shows the sizes of the generated BLs. After having all operator-specific BLs acquired, they were merged track-wise and all together to generate larger BLs. During the merge process, identical behaviors and state contexts are reused. Some evaluated contexts were identical in many libraries, due to the discretization before storing, resulting in a smaller size than all of the BLs combined. In contrast, no behavior was used in more than in one BL.

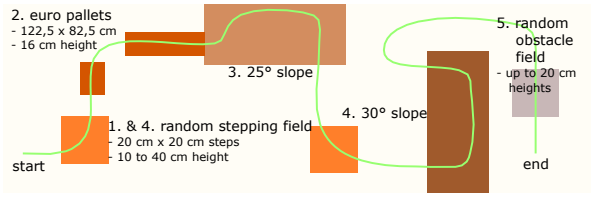
Since the library size alone does not provide an adequate measurement of the maturity of the BL, evaluated state contexts and the range of each performance feature need to be analyzed as well. A histogram per context feature is applicable to analyze the former. It provides a good estimate which contexts are evaluated and which ones need to be explored before delivering reliable information for autonomous behavior selection. The expected performance can be analyzed by boxplots. There, it needs to be distinguished between action performance features (*velocity x*, *velocity y*, *turn rate*, *body height*, and *body width*) and meta performance features (*ssm*, *dsa*, *epd*, *power*, *body vibration*).

[1] https://rock-simulation.github.io/mars/

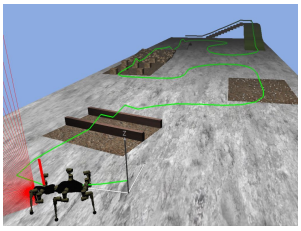
[2] http://archive.darpa.mil/roboticschallengetrialsarchive/



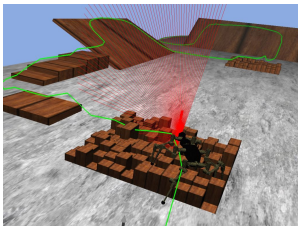
(a) Outdoor test track schematic



(b) Indoor test track schematic



(c) Outdoor test track simulation



(d) Indoor test track simulation

Fig. 6. Test tracks to generate *behavior libraries*

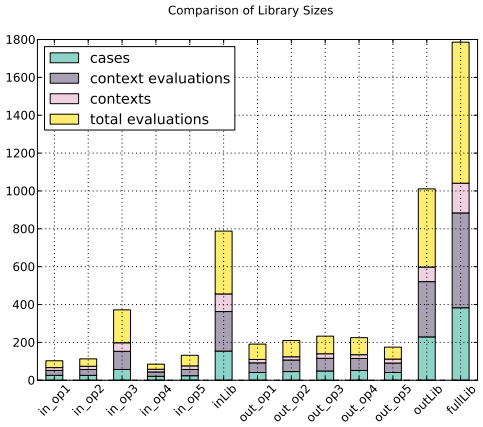
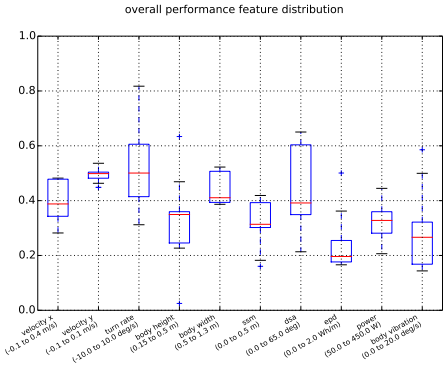
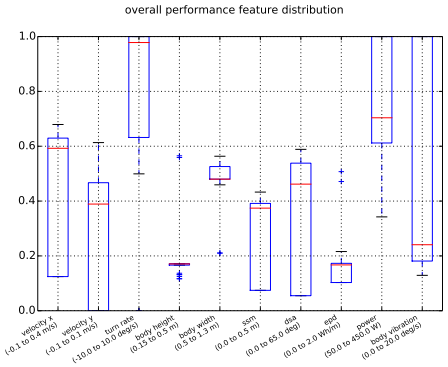


Fig. 7. Comparison of individual and merged *behavior libraries*

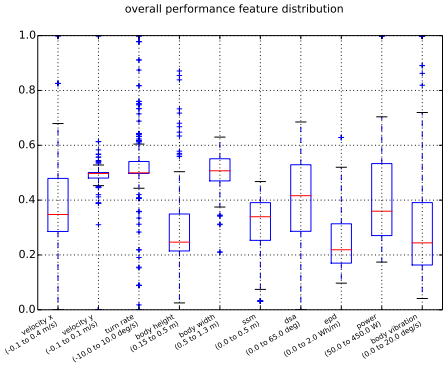
Fig. 8(a) shows the boxplot of *out_op3* which has small boxes and few outliers, indicating a library of small exploration which is less general applicable but very confident within a certain range of action. In contrast, Fig. 8(b) shows the library of *out_op2* which covers a broad range of performance. But since the number of evaluated behaviors is similar, its confidence over the whole range of performance is less. In addition, asymmetries at the action performance features show an unbalance, e.g., when building the BL of *out_op2* no locomotion behavior to turn leftwards could be stored. The merge of all BLs (Fig. 8(c)) shows a plot with more symmetric boxes covering a broader spectrum of each action performance feature, indicating higher flexibility and confidence. As long as a BL provides sufficient behavior for the requested actions, its performance is better the closer the boxes of the meta performance features are to their optimums.



(a) *Behavior library* with few performance exploration



(b) *Behavior library* with large performance exploration, but few confidence



(c) Mature *behavior library*

Fig. 8. Maturity compare of *behavior libraries*

B. Fully Autonomous Traversing of an Obstacle Course

In this experiment, the operator was replaced by a trajectory follower to create motion commands with a maximum of 150 mm/s of longitudinal translation and 10°/s rotational speed. Furthermore, the behavior configurator utilized the previously generated BLs to autonomously map the incoming motion commands to control parameters of the motion control. The task was still to traverse the obstacle course along

the given trajectory. For this application, the performance prioritization was chosen as follows. The weight of *turn rate* was set to 1.0 and of *velocity x* to 0.8 to underline path following characteristics. Since *velocity y* is not commanded by the trajectory follower, its weight was set to 0.2 to degrade behaviors which move sidewards unintendedly. The task does not require a specific *body height* or *body width*. Consequently the weight of these action features was set to zero. The meta performance weights were set to 0.1 so that behaviors of similar action performance but with more meta performance are rated higher. Exceptions are *ssm* and *power* which were set to zero since stability and energy efficiency are incorporated by *dsa* and *epd*, respectively.

Fig. 9 shows the performance after traversing the *outdoor* obstacle course manually (*operator*) and autonomously on basis of different BLs. The individual operator-specific libraries were tested independently and their results than merged (*opLibs*). Furthermore, the BLs generated on the outdoor test track (*outLib*), the ones generated on the indoor test track (*inLib*), and all together (*fullLib*) were merged and tested. In addition to the operator evaluation, the plot lists also the percentage of how often the test track was completed. The graph and the observation of the experiments yield following conclusions:

- The performance of the autonomous control using an individual operator library is performing worst since none of them accomplished the task. Every BL has its own weakness, e.g., the library of *out_op1* has no behavior which sets the turn rate in slopes resulting in falling from the hill (obstacle 6), the library of *out_op2* has no adequate leftwards turn behavior resulting in exceeding the map limits (at obstacle 2), the libraries of *out_op3*, *out_op4*, and *out_op5* have no point turn behavior which resulted in falling from the edge between hill and stairs (between obstacle 6 and 7). The lack of behaviors results from the requirement that a behavior needs to stay constant for a step cycle to get evaluated, e.g., an operator who continuously tunes the control parameters allows less behavior evaluations.
- Merging of all individual outdoor libraries results in BL of higher maturity because the knowledge of the single Bls is combined, thus, the performance was increased. The energy efficiency was better compared to manual control because there were less situations where the robot got stuck due to an unsuitable parametrization. In 60% of the runs, the obstacle course could be completed. But variances of each run, originated from multi-tasking computation, paired with comparably low path following capability led to falling from the edge on top of the hill.
- The BL merged from the indoor libraries lead to a control which works in principle even trained on different obstacles but with similar feature representation. But it has no behavior to cope with the tight point turn on top of the hill (between obstacle 6 and 7). Still the transferability of the proposed approach is shown.

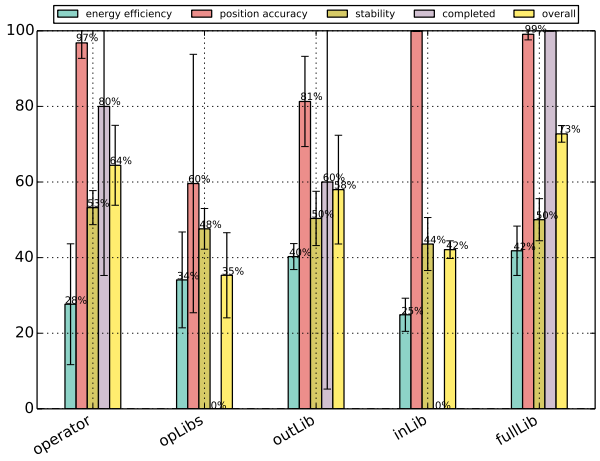


Fig. 9. Performance comparison while traversing the *outdoor* obstacle course. Note that *operator* represents the average scores of all five operators who traversed this terrain manually and *opLibs* shows the same for autonomous traversing while using the single five individual operator libraries, respectively. The other three plots are averaged over five autonomous runs of each merged library.

- The autonomous control using the library containing all data performed best. It completed the obstacle course and the position accuracy was increased compared to the outdoor library.
- In the cases where the autonomous control did not complete the course, missing maneuverability was the reason, which could be improved by intervening with manually setting motion commands. In contrast, the failure reason during manual control was a bad parametrization during a descend (at the end of obstacle 4) causing the robot to tip over. The latter reason is more serious and may occur more often in situations where the operator's task load is very high.

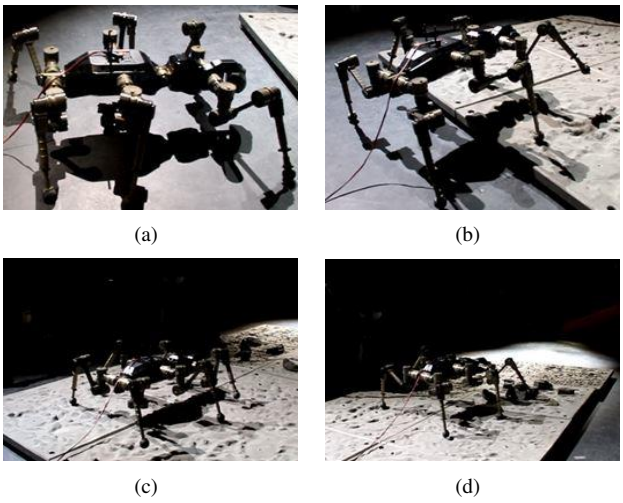
C. Transfer to Real System

To test the applicability of the experience-based adaption of the motion control in reality, it was used to apply in simulation leaned behaviors on the real SpaceClimber. The task was to overcome an obstacle course consisting out of a step of 12 cm height, followed by a 2 m plane and a random obstacle field of up to 20 cm tall obstacles.

The pictures of Fig. 10 show, that the robot was performing its motion as intended. Besides other parameters of less impact, body and step height were autonomously lifted when encountering the step. After passing it, SpaceClimber was returning to its energy-efficient lower locomotion behavior. Then, a higher walking pattern was chosen to overcome the random obstacle field. Further investigations on the transfer from simulation data to the real system is provided in [20].

VI. Conclusion and Outlook

This paper presents an experience-based interface layer for the motion control of kinematically complex robots. It autonomously maps scenario-specific actions to robot-specific control parameters by incorporating the current context and



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

Fig. 10. SpaceClimber autonomously switching between energy-efficient walking in flat terrain and slower walking pattern with taller body and step height to traverse a step and a random obstacle field.

collected experiences, i.e. performance of behaviors in previously visited contexts. Thus, the probably best locomotion behavior is selected which takes an adjustable performance prioritization into consideration. The determination of appropriate state context and performance features is shown on the example of the walking robot SpaceClimber. The structure of the BL, in which all experiences are stored, is described. A case-based reasoner utilizes this information to derive control parameters generating various locomotion behaviors.

The experiments show that a suitable BL can be generated by recording the chosen parameters as well as the passed state context and evaluated behavior performance features while an operator is manually controlling a walking robot with his or her expert knowledge. The proposed experience-based control approach could then be used to traverse a demanding obstacle course autonomously. Different BLs with respect to size and maturity are compared. It was shown that the real SpaceClimber could also autonomously adapt its locomotion behavior to traverse another obstacle course by using the experience-based approach with knowledge generated in simulation.

In the current implementation, more evaluated behaviors lead to better performance since only the best behavior is selected. In the future, more advanced regression techniques are planned to be used to generate better behaviors with less knowledge. Furthermore, autonomous setting of the performance prioritization needs to be analyzed. Lifelong learning including wear out of the robot is also a future topic as well as the incorporation of more features, e.g, soil characteristics. In addition, the information stored in the BL are also valuable areas in other areas, e.g., to improve cost calculation in path planning or to detect anomalies.

ACKNOWLEDGMENT

The presented work was carried out in the project LIMES, a collaboration between the DFKI Robotics Innovation Center and the University of Bremen, funded by the German Space Agency (DLR, Grant numbers: 50RA1218, 50RA1219) with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament.

REFERENCES

[1] T. Köhler, C. Rauch, M. Schröer, E. Berghöfer, and F. Kirchner, "Concept of a biologically inspired robust behaviour control system," in *Intelligent Robotics and Applications*, 2012, vol. 7507, pp. 486–495.

[2] P. Putz and A. Elfving, "Control techniques 2," Dornier GmbH, Tech. Rep. CT2/CDR/DO/BL, August 1991, issue: 1.1, ESTEC Contract 9292/90/NL/JG (SC).

[3] R. Arkin, *Behavior-based robotics*. MIT press, 1998.

[4] E. Burattini and S. Rossi, "Periodic activations of behaviours and emotional adaptation in behaviour-based robotics," *Connection Science*, vol. 22, no. 3, pp. 197–213, Sept. 2010.

[5] M. N. Nicolescu and M. J. Mataric, "Experience-based learning of task representations from human-robot interaction," in *Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on*. IEEE, 2001, pp. 455–460.

[6] B. Gassmann, "Modellbasierte, sensorgestützte navigation von laufmaschinen im gelände," Ph.D. dissertation, University Karlsruhe, 2007.

[7] F. Michaud, "Emib - computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules," *Cognitive Science Quarterly, Special Issue on Desires, Goals, Intentions, and Values: Computational Architectures*, vol. 3-4, pp. 340–361, 2002.

[8] C. E. Rassmussen, "Gaussian processes in machine learning," *Int. journal of neural systems*, vol. 14, no. 2, pp. 69–106, Apr. 2004.

[9] H. Hoffmann, G. Petkos, S. Bitzer, and S. Vijayakumar, "Sensor-assisted adaptive motor control under continuously varying context," *Int. Conf. on Informatics in Control, Automation and Robotics*, 2007.

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[11] J. L. Kolodner, "An introduction to case-based reasoning," *Artificial Intelligence Review*, vol. 6, no. 1, pp. 3–34, 1992.

[12] S. Bartsch, T. Birnschein, M. Römmermann, J. Hilljegerdes, D. Kühn, and F. Kirchner, "Development of the six-legged walking and climbing robot spaceclimber," *Journal of Field Robotics*, vol. 29, pp. 506–532, 2012.

[13] J. Hidalgo-Carrio, A. Babu, and F. Kirchner, "Static forces weighted jacobian motion models for improved odometry," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.

[14] J. Schwendner and F. Kirchner, "eslam - self localisation and mapping using embodied data," *KI-Künstliche Intelligenz*, vol. 24, no. 3, pp. 241–244, 2010.

[15] R. McGhee and G. I. Iswandhi, "Adaptive locomotion of a multilegged robot over rough terrain," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 9, no. 4, pp. 176–182, April 1979.

[16] E. Papadopoulos and D. a. Rey, "The Force-Angle Measure of Tipover Stability Margin for Mobile Manipulators," *Vehicle System Dynamics*, vol. 33, no. 1, pp. 29–48, Jan. 2000.

[17] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2276–2282.

[18] J. Gu, Q. Cao, and Y. Huang, "Rapid traversability assesment in 2.5 d grid based map on rough terrain," *Int. Journal of Advanced Robotics*, vol. 5, no. 4, pp. 389–394, 2008.

[19] M. Langosz, L. Quack, A. Dettmann, S. Bartsch, and F. Kirchner, "A behavior-based library for locomotion control of kinematically complex robots," *Int. Conf. on Climbing and Walking Robots*, pp. 495–502, 2013.

[20] A. Dettmann, M. Langosz, K. von Szadkowski, and S. Bartsch, "Towards lifelong learning of optimal control for kinematically complex robots," *ICRA14 Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots*, 2014.