Development and Control of the Multi-Legged Robot MANTIS

Sebastian Bartsch[1], Marc Manz[1], Peter Kampmann[1], Alexander Dettmann[2], Hendrik Hanff[1], Malte Langosz[1], Kai von Szadkowski[2], Jens Hilljegerdes[2], Marc Simnofske[1], Philipp Kloss[1], Manuel Meder[2], and Frank Kirchner[1,2]

[1]DFKI GmbH, Bremen, Germany
[2]University of Bremen, Bremen, Germany

Abstract

This paper presents the multi-legged robot MANTIS which is developed within the project LIMES at the DFKI RIC and the University of Bremen. In particular, we describe the mechanical design, the sensor setup, electronics, and computing hardware. Furthermore we give a short introduction to the software framework for simulation-based motion behavior generation and optimization for such kinematically complex robots as well as to the online locomotion control and evaluation approach for context-based utilization and adaptation of these behaviors. Finally, applied methodologies and experiments allowing to assess and reduce the difference between observed and simulated behavior of the robot and its subsystems are presented.

1 Introduction

Thus far, the field of mobile robots for space exploration has been dominated by wheeled systems, avoiding highly complex locomotion control and high energy consumption on flat terrain while building on experience gained in applications on Earth. However, depending on the planned mission, the limited mobility of wheeled rovers in unstructured terrain can outbalance their advantage on flat terrain, which is why walking systems might at least be competitive if not required in the first place [1]. Furthermore, future missions will include more complex manipulation tasks than in-situ investigation or sample collection, as for instance the building up and maintenance of infrastructure will be required to support and sustain human presence on extraterrestrial bodies [6].

MANTIS [1] (**Figure 1**) is developed with the aim to provide high mobility and manipulation capabilities in uneven and unstructured terrain and thus features six legs, two of which double as arms and are equipped with grippers, enabling the robot to perform dual-arm manipulation. The head and end-effectors of the extremities are equipped with various sensors to acquire data on the environment while most of the electronics for power management, high-level processing and overall robot control are housed in the rear section of the system. MANTIS is controlled by a network of FPGAs and micro-controllers on the subsystem level whereas the central high-level control is implemented using Rock [2] (Robot Construction Kit), a software framework based on Orocos RTT (Real Time Toolkit).

Figure 1 The robot MANTIS in upright manipulation posture

This paper explains the mechanical design inspired by the praying mantis insect, provides an overview of the electronics development and describes the software architecture used to control locomotion and adapt the robot's behavior to its current environmental, task-dependent and internal conditions.

[1]The presented system is part of the Project LIMES (Learning Intelligent Motions for Kinematically Complex Robots for Exploration in Space) funded by the Space Agency of the German Aerospace Center with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament, grant no. 50RA1218.

[2]`http://rock-robotics.org`

2 Mechanical Design

MANTIS is designed following the main idea to create a robot which is able to walk statically stable while manipulating with two arms based on the concept presented in [11]. Although statically stable walking is possible on two extremities, walking on four extremities while maintaining ground contact with at least three of them provides higher stability, which is particularly useful when additional forces arise during manipulation. A system like MANTIS is well-suited for applications requiring high manipulation payloads during locomotion, e.g., clearing disaster sites or setting up infrastructure in a planetary exploration setting.

MANTIS possesses six extremities for locomotion, each having six active degrees of freedom (DOF). In addition, MANTIS is able to erect its body and free the two foremost extremities to use them as arms, both featuring three-fingered hands for dual arm manipulation and a bracket to walk on it. The main electronic compartment is located in the the rearmost body segment, the abdomen, providing a counterweight for the upper body and thus shifting the center of mass towards the frame articulation. This feature facilitates switching between the locomotion and manipulation postures. In the former, the actuated frame articulation allows the shifting of the center of mass along the robot's longitudinal axis if both linear actuators extend or retract simultaneously, while in the latter two types of movement are possible: simultaneous movement of the actuators leads to an alteration of the robots height, contrary movement allows the robot to lean to the left or right.

The sensor head is actuated by three joints and contains a stereo camera system as well as an inertia measurement unit and a lidar sensor. The first joint is used to compensate the torso's pitch, simplifying control algorithms and allowing intuitive teleoperation of the system. Altogether, MANTIS has 61 active DOF used to control the movements of its body. The system with an overall weight of 109 kg is able to carry 40 kg of payload (see **Table 1**).

Table 1 MANTIS specification

Attribute	Value	Unit
Mass	109	kg
Payload Capacity	40	kg
Dimensions [l x b x h] (Manipulation)	approx. 1,4 x 1,8 x 1,8	m
Dimensions [l x b x h] (Locomotion)	approx. 2,2 x 1,8 x 0,8	m
Number of DOF	61	
Operating Voltage	48	V
Quiescent Current	2.6	A

The structure is realized using cast aluminum parts and plastic tubes reinforced with carbon fibre. The structure of the abdomen and the lower arm of the manipulation extremities are realized with sheet metal to simplify upgrading individual parts such as computers and FPGA boards.

2.1 Arm Extremities

The arm extremities are used for both locomotion and manipulation (**Figure 2**). This is why each arm is equipped with a three finger gripper to manipulate objects and a covering bracket to walk on. Each finger has two DOF and is mounted on a force-torque (FT) sensor to enable force-controlled grasping. The outer two fingers can be rotated up to 180° around the palm. Furthermore, the whole gripper and the walking bracket are mounted on an FT sensor. The grasping surfaces of the finger phalanges are equipped with fiber-optic tactile sensors to allow a calculation of the pressure distribution. The gripper is designed as an integral part of the forearm, all actuators to drive the tendons for the finger actuation are located within it.

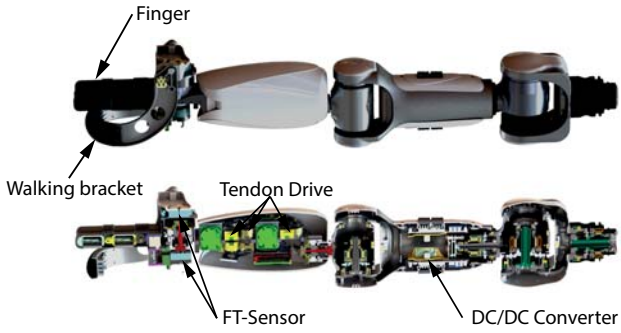

Figure 2 The arm in sectional view.

2.2 Leg Extremities

The leg extremities used for locomotion are designed to be collapsible to reduce the required stowage space. The leg has three DOF near the proximal mounting flange, one knee like DOF in the middle and two DOF in the distal ankle part (**Figure 3**).

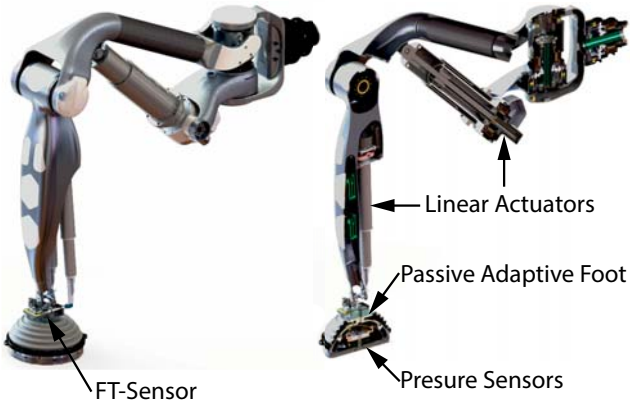

Figure 3 The leg in sectional view.

The lower leg consist of an active and a passive adaptive part. The active part is used to align the foot while the passive part is used for dealing with uneven terrain. The

adaptive characteristics are obtained with two hemispheres, the lower sliding on and being guided by the upper. The resulting center of rotation is below the ground so that this configuration is inherently stable.

2.3 Actuators

MANTIS has 61 actuators to move the body, consisting of six types of rotation actuators, three types of linear actuators, one tendon drive and one servo to tilt the lidar sensor (**Figure 4** and **Table 2**).

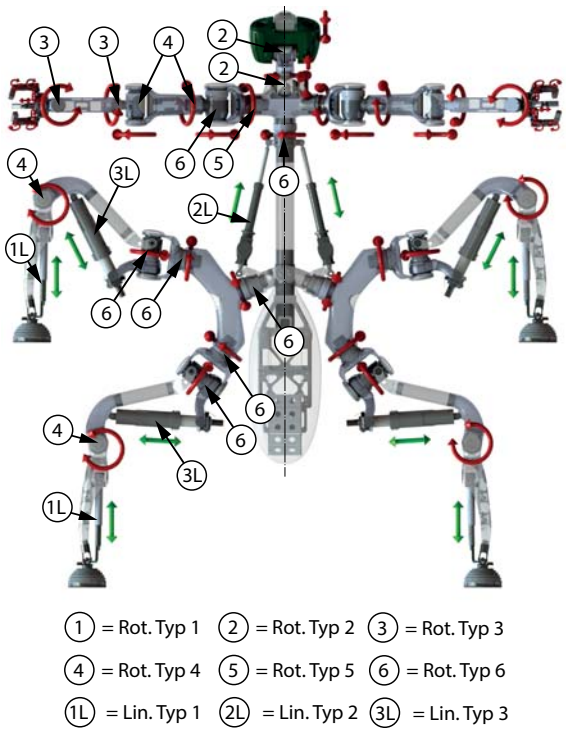

Figure 4 Actuated DOF of MANTIS, including both rotational (red) and linear (green) actuators, with arrows indicating motion axes.

Table 2 MANTIS actuator specifications

Rotational Actuators

Actuator Type	Nominal Torqe [Nm]	Nominal Speed [rpm]	Motortyp	Nominal Voltage [V]	Nominal Current [mA]
Rot. Type 1	2,5	29,4	EC 32 flat	24	500
Rot. Type 2	6,7	32,9	EC 45 flat	36	849
Rot. Type 3	36,8	43,75	ILM 50x14	48	3500
Rot. Type 4	54	21,87	ILM 50x14	48	3500
Rot. Type 5	92	21,87	ILM 70x10	48	7000
Rot. Type 6	176	13,125	ILM 70x18	48	7000

Linear Actuators

Actuator Typ	Nominal Force [N]	Nominal Speed [mm/s]	Motortyp	Nominal Voltage [V]	Nominal Current [mA]
Lin. Typ 1	804	81	EC 45 flat	24	3210
Lin. Typ 2	710	5	EC-max 30	36	970
Lin. Typ 3	1400	147	ILM70x18	48	7000

Each rotational and linear actuator is equipped with a stack of three PCBs which consist of a power module with motor driver, a logic board equipped with an FPGA and a connector board. All motors are commutated brushlessly and powered with direct current (BLDC). Most motors are equipped with magnetic off-axis absolute position encoders (working according to the nonius principle) which are capable to emulate both hall sensor signals for motor commutation as well as quadrature encoder signals for speed estimation. Identical magnetic absolute encoders are used to measure the rotational position of the output shaft of every DOF. The rotational actuators (Typ 3-6) provide a hollow shaft to feed through power and communication wires. The rotational actuators are depicted true to scale in **Figure 5**.

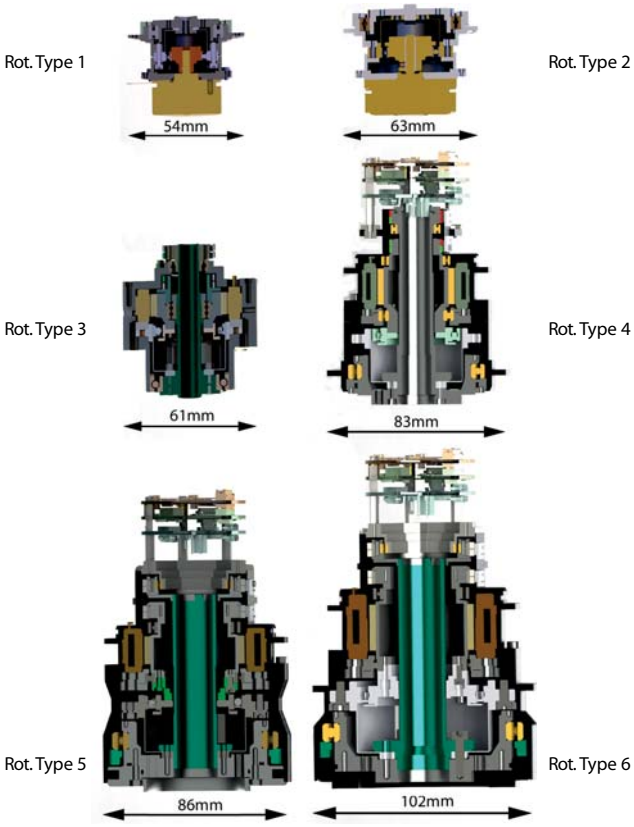

Figure 5 The rotational actuator types.

The smallest drive units are realized with external rotor motors in order to achieve the highest possible number of pole pairs and a large air gap diameter for a higher torque. The external rotor motors provide better torque density for this range of motors [14].

3 Electronic Design

Over the last several years in robot development, the topology of the hardware architecture changed from centralistic approaches to networks of heterogeneous processing units, an approach that has several advantages. It builds the foundation towards robustness against hardware failures, leads to increased signal quality due to analog-to-digital conversion right at the signal acquisition, and enables balancing of

processing load on the hardware via local pre-processing. Since the complexity of algorithms running on embedded hardware is increasing, the predominantly used term 'low-level processing' is often not fully applicable anymore which is why 'first-level processing' is used here for these types of electronics.

A driving aspect of shifting algorithms to controllers that are distributed across the robot are the equally distributed sensors. Generally, to support the possibility to design autonomous behavior with rising complexity and dealing with more than a limited set of tasks, sensors of different modalities are needed almost everywhere in a robot. **Figure 6** gives an overview of the distribution of MANTIS' sensors. In total, 88 position encoders, 14 six-axis force-torque sensors, 2 IMUs, 2 HD cameras, 1 lidar, 122 temperature sensors, 191 current measurement sensors, and 12 tactile sensors comprising 40 sensing elements can be found in the whole system. The overall data volume generated by this system amounts to 629 megabytes per second (compare **Table 3**).

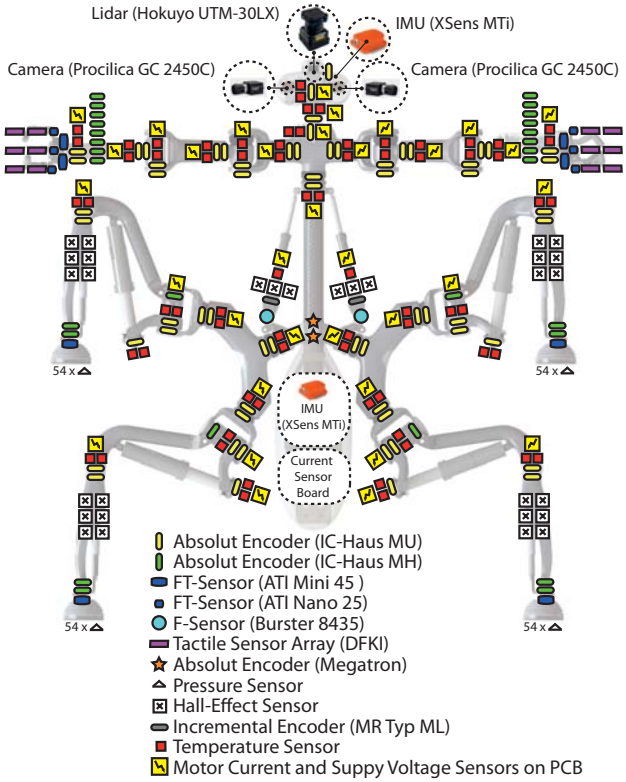

Figure 6 Sensor distribution within the MANTIS robot.

The designed hardware architecture that handles the processing of the sensor data and controlling of the robot is depicted in **Figure 7**. In total, there are 62 FPGAs, 14 microcontrollers, and one standard x86 central processing unit integrated into the system. Like for the homunculus, an artficial model of the senory or motor cortex in human beings, it can be seen that some areas of the robotic system are equipped with a higher density of processing units than others.

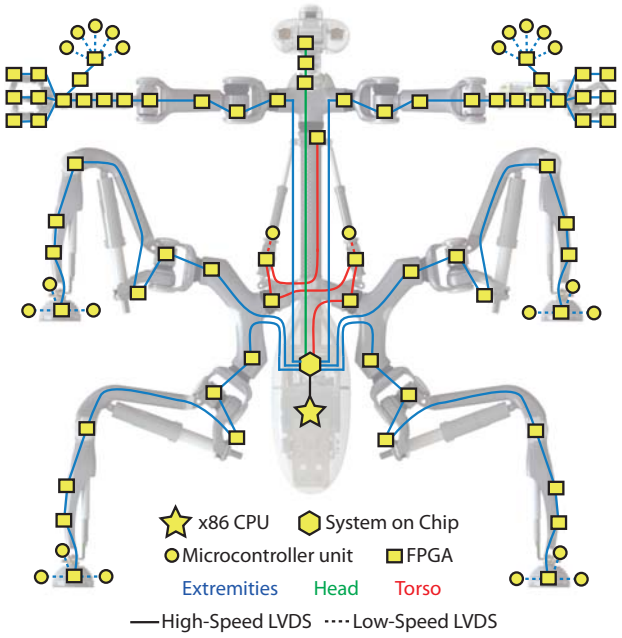

Figure 7 Architecture of the processing units within the MANTIS robot.

3.1 First-level control of BLDC signals

Most of the processing elements are placed in the system in direct neighbourhood of actuation units. Since the introduction of custom PCBs for BLDC motor control in the robots developed at the DFKI RIC [5], the motor actuation and control is realized locally. The technological advancement allowed us to move from Xilinx Spartan 3 FPGAs having roughly 17,000 logic elements to Spartan 6 modules containing 44,000 logic cells while staying roughly in the same form factor. This development supports the design approach to move as much signal processing and control as possible to first level processing units. Over the years the local processing evolved to hierarchical controllers and the implementation of advanced protocols for heterogeneous processing networks like Node-Level Data Link Communication (NDLCom) [16].

3.2 First-level control of tactile sensor data

Further processing power is placed near the grippers where different modalities of contact sensing support the grasping of objects. Based on previous experiences with this technology [7], fiber-optic sensing principles were selected to form tactile sensor arrays in the palm.

The integrated sensor module can be seen in **Figure 8**. It consists of an upper layer where polymer-optical-fibers are placed in pairwise manner on a bended contact surface. The processing electronics at the lower layer consist of a small FPGA (Lattice MachXO2 4000) capable of extracting the tactile sensor information from a camera image that is capturing the endings of the sensing optical fibers of the sensor. The overall dimensions of the sensing module are 40 mm by 15 mm in width and length and 17 mm in height.

Table 3 Upper boundaries for sensor data generated every second in the MANTIS robot

Sensor	Type	Freq. [Hz]	Data [Bit]	Volume [MB/s]	Num. Sensors
IC-MU	Encoder	4000	12	0.005	58
IC MH	Encoder	4000	12	0.005	30
ATI FT	6 axis Force-torque	12	16	> 0.001	14
XSens MTi	IMU	2000	16		2
GC2450C	Camera	15.1	160.5M	287	2
Hoykuyo UTM	Lidar	40	26k	0.123	1
LM75	Temperature	10	9	> 0.001	61
DFKI Skin	Tactile Sensors	30	2.4M	1	12
Total:					180

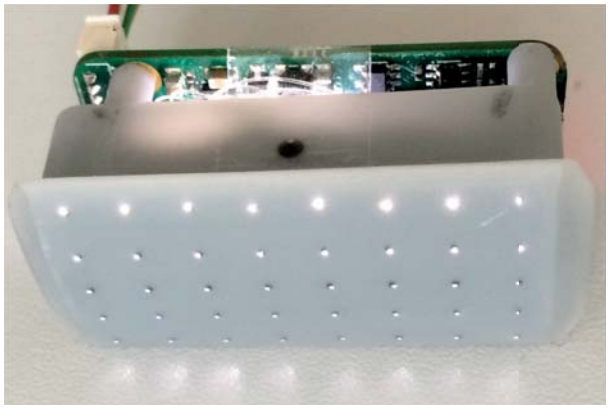

Figure 8 Integrated fiber-optic sensing module

Together with the 6 DOF force-torque sensors, these sensors implement the tactile sensing capabilities of the robot. Two FPGAs in the lower arm allow local pre-processing of the data (e.g. determination of force levers) and a direct reaction to processed stimuli. Examples for this are the realization of stable grasps directly at the first level without including higher-level computation units.

Assembling the data from first-level processing and forwarding it to standard processing units is a task where embedded communication signals have to be acquired from a considerable amount of input channels. Such preprocessing requires a system capable of massive parallel data handling in combination with a serial processing unit. The Xilinx Zynq System-on-a-Chip (SoC) is a perfect candidate for this task. Massive parallel processing power is delivered by an FPGA and the embedded dual core ARM Cortex A9 core provides enough processing power to do high level computing on a Linux operating system. Choosing this embedded two-in-one solution ensures that the overall footprint of the processing unit remains small when compared to discrete solutions. The lack of commercial off-the-shelf (COTS) products containing a Xilinx Zynq which are able to satisfy all of our requirements led to an in-house development. **Figure 9** shows the final result of the latter, the *ZynqBrain*. The ARM Controller permits the installation of a standard Debian Linux which has the advantage of providing all software packages from the Debian software repository. On top of that the installation of the ROCK software framework was easy compared to the installation on more exotic operating systems. The embedded FPGA is, among other things, responsible for preprocessing the data received from the limbs and controlling other hardware such as current measurement unit, emergency switch, relais board (for motor supply), and battery surveillance.

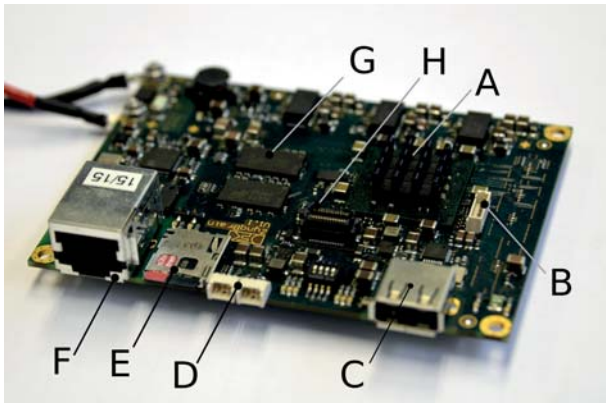

Figure 9 ZynqBrain containing the system-on-a-chip (SoC) Zynq device including a reconfigurable FPGA and a dual-core ARM A9 processor plus peripherals. A: Xilinx Zynq SoC with mounted heat sink, B: JTAG port, C: USB port, D: 2 UART ports, E: Micro SD Card Slot, F: Ethernet Connector, G: Memory (RAM), H: Inter-board connector. Not visible is the connector located on the back of the ZynqBrain which permits to connect and stack different extension boards.

4 Software Design

As described in the previous sections, MANTIS is a system with an extensive sensory and actuatory equipment by which the robot should gain extensive capacities to interact with and perceive its environment in different ways. To implement this abilities to its full extend, the control software has to enable the robot to identify the current situation in order to decide on and execute appropriate motions to achieve the intended objective at the best possible performance.

4.1 Locomotion

MANTIS possesses a behavior-based control architecture, combining various behaviors such as central pattern generators with postural behaviors and reflexes to generate stable locomotion. Underlying the behaviors executed on the robot are *behavior graphs* (BGs), computation networks featuring elements from neural networks as well as genetic programming, previously described by Langosz et al. ([8, 9]). The use of BGs allows the graph-based development of locomotion control algorithms while providing an interface for machine learning algorithms. These can either be used to optimize control parameters of existing graphs for specific actions or to develop entirely new control graphs or embedded subgraphs. As learning has to be conducted for a wide variety of environments and over extensive sets of repetitions, a simulation environment[3] is used, facilitating repeatability and providing accurate measurements of evaluation criteria such as stability, energy efficiency, or body vibration to calculate a fitness value.

4.1.1 Behavior Graphs

Nodes in BGs (**Figure 10**) - in contrast to nodes in classic artificial neural networks (ANNs) - can have an arbitrary number of inputs and outputs and a freely definable *transfer function* to map the former to the latter. While this allows emulating ANNs and similarly defined networks made from simple nodes for use with graph-building machine-learning algorithms such as NEAT ([15]) or SABRE ([9]), it further enables manual definitions of complex computations such as inverse kinematics in a single node as well as wrapping subgraphs into single nodes and using numerically intuitive global control parameters.

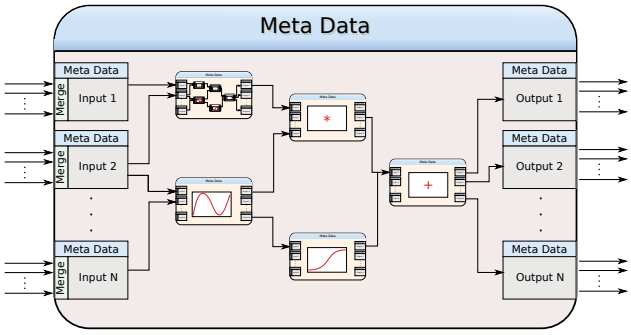

Figure 10 Structure of a behavior graph such as the ones used for MANTIS (from [8]).

All of this greatly simplifies the use of machine learning algorithms to improve existing and develop entirely new graphs. For instance, parameter optimization algorithms such as CMA-ES [4] can be both used to tune manually predefined control parameters of an engineered graph and to adapt the non-intuitive parameters of an evolved graph without changing the interface between control structure and machine learning components. This becomes especially important when combining switching of behaviors with online-adaptation in the future [3].

[3]MARS (http://www.mars-sim.org)

4.1.2 Behavior Library and Configuration

Human engineering as well as automated learning in simulation yields a number of graphs defining robot behaviors, necessitating mechanisms to catalog behaviors and to choose which behavior to execute for a given task in a given environment considering the observed system state. For this purpose, a *behavior library* was created for MANTIS, where both types of behaviors are stored after their performance was evaluated in a variety of contexts, the latter being defined as a combination of a provided task and detected environment. A *behavior configurator* (see **Figure 10**) then automatically chooses behaviors based on the context the robot operates in. This system is combined with an interface to a deliberative navigation layer, allowing to use knowledge from past behavior evaluations in planning processes [2]. Crucial requisites for a proper functioning of such an approach are the capabilities of the system to perform online context identification as well as performance self-evaluation using the available sensory equipment.

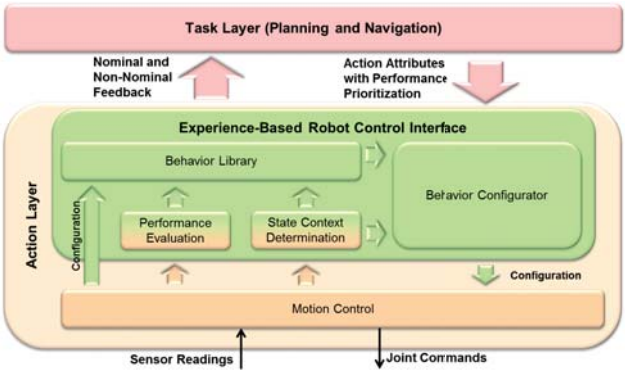

Figure 11 The control architecture used for MANTIS (from [2]).

4.2 Manipulation

A whole body control approach is used to manipulate objects while observing constraints such as stability, thus considering all body joints to bring the hands into desired grasping poses. The contact surface of the manipulated object can be determined by the contact arrays on each finger. The huge amount of data is directly processed in the FPGA of each hand to reduce computation and communication load of the main CPU.

For good grasping poses the robot uses color and depth data from the cameras and lidar integrated in its head. This data gets fed to an ANN model that was pre-trained on labeled RGB-D images. For this pre-training of the neural network, the Cornwell data set and the data preprocessing methods from Lenz et al. [10] are used. In order to reduce the vast search space for possible grasps Bayesian Optimization is used. The thus determined grasp poses

are ranked using the ANN's output as a score. Before initiation of the actual closing movement of the gripper MANTIS' manipulator has to be moved towards the object. This approaching which is part of the grapsing procedure is learned using imitation learning. A human demonstrates the grasping trajectory which then gets represented by a Dynamic Movement Primitive and is transferred onto the robot using the learning platform described by Metzen et al. [12].

5 Reference Experiments

Given the extensive use of simulation for the development and optimization of behaviors of MANTIS, it is essential that its simulation model resembles the real system as closely as possible. This is especially important since some scenarios such as reduced gravity conditions, e.g., on the Moon can only be tested in simulation. However, when creating and revising MANTIS' simulation models along with the ongoing mechanical design, it is also important to weigh accuracy and detail against model complexity, especially for the purpose of optimizing behaviors in simulation, where computation speed is paramount. For instance, as a rigid body model is used in MANTIS' simulation, structural flexibilities can only be emulated using additional passive joints, inflating the model's complexity and thus leading to slow computation and possible instabilities.

For creating a complete model, a two-step approach is implemented. First, individual components such as motors and structural elements are modeled after data obtained from their real counterparts. For structures, this comprises obtaining values for masses and inertia from CAD, but can also include testing bending stiffness on an appropriate bending test bench. For motors, a specially designed motor test bench is used to characterize the controller reaction and power consumption across the range of speed and torque of different motor types. The results obtained from these measurements enable the use of black-box function approximation to recreate the same behavior of motors in simulation, e.g., for power consumption which is an important evaluation factor for behavior optimization. Similarly, since controllers in simulation run with a much slower frequency than in reality, determined by the iteration step duration of the simulation (often as long as 10 ms), different sets of parameters have to be found approximating the real motor behavior, which is a parameter optimization problem in itself and can be tackled with similar methods as parameter optimization of behaviors.

In a second step, the whole body behavior of the robot can be compared to the simulation model to further tune overall simulation parameters, such as contact softness and friction, both essential to correctly recreate the behavior of a walking robot. For this purpose, a walking test bench was created, comprised of a treadmill large enough for MANTIS with its wide stance to walk on and thus allowing to use a stationary motion tracking system to monitor the robot's movements.

Finally, it is important to note the virtual impossibility of completely closing the simulation-reality-gap due to unaccountable variance both in controlled measurements and even more so in the field, with often unpredictable influence of environmental conditions (e.g. temperature or soil humidity) and interaction (e.g. sinking or slipping on the surface).

6 Conclusion and Outlook

The development of robotic systems with numerous DOF as well as high sensor data quality and quantity leads to an ever-increasing amount of data that needs to be transformed into useful information for high-level decision making. With the ongoing effort to acquire sensory information with higher spatial density, higher resolution or different modalities, this trend will only continue in future robotics development. Handling the bottleneck of processing power by dynamically distributing tasks among the processing hardware [13] or applying concepts from disciplines like Big Data and high performance computing will become increasingly necessary to effectively handle the growing amount of data and processing nodes in future robots.

Similar considerations apply to the behavior-based locomotion control approach utilizing a behavior-library to store optimized behaviors and gained experiences represented as corresponding performance evaluations in certain contexts. Here, the number of behaviors and evaluations is currently still relatively small, keeping the library as well as the selection process manageable in terms of memory usage and computing time. It is desirable, though, for the robot to evaluate the performance of its behavior online and thus to continuously improve its behavior library in a life-long learning approach. This includes the exploration of novel behaviors in unknown contexts through interaction with the environment as well as the alteration of past evaluations according to a worsening performance resulting from wear out of the system. Consequently, strategies to keep the library manageable yet capable of storing as much of the gained experiences as possible, while including the possibility to diminish previous evaluations and to remember bad experiences, are required.

In future walking experiments reference data will be acquired and subsequently used for further optimization of the overall simulation model. This will allow to develop behaviors for walking and other tasks purely in simulation that can be transferred on the real robot with minimal adaptation effort. Another advantage is that the simulation can be used to generate, evaluate, and optimize locomotion behaviors for other gravity conditions such as on the lunar surface to set up an suitable initial behavior library for a real extraterrestrial mission.

7 Literature

[1] Sebastian Bartsch, Timo Birnschein, Malte Römmermann, Jens Hilljegerdes, Daniel Kühn, and Frank Kirchner. Development of the Six-Legged Walking and Climbing Robot SpaceClimber. *Journal of Field Robotics*, 29:506–532, 2012.

[2] Alexander Dettmann, Anna Born, Sebastian Bartsch, and Frank Kirchner. Experience-Based Adaptation of Locomotion Behaviors for Kinematically Complex Robots in Unstructured Terrain. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*, Hamburg, Germany, September 2015.

[3] Alexander Dettmann, Malte Langosz, Kai Alexander von Szadkowski, and Sebastian Bartsch. Towards Lifelong Learning of Optimal Control for Kinematically Complex Robots. In *Proceedings of IEEE ICRA14 Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots*, Hong Kong, China, June 2014.

[4] Nikolaus Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[5] Jens Hilljegerdes, Peter Kampmann, Stefan Bosse, and Frank Kirchner. Development of an Intelligent Joint Actuator Prototype for Climbing and Walking Robots. In *12th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR-2009)*, pages 942–949, Istanbul, Turkey, September 2009.

[6] International Space Exploration Coordination Group (ISECG). The Global Exploration Roadmap. Online at http://www.globalspaceexploration.org, August 2013. last visit 2016/03/30.

[7] Peter Kampmann and Frank Kirchner. A Tactile Sensing System for Underwater Manipulation. In *Proceedings of the workshop on: Advances in Tactile Sensing and Touch based Human-Robot Interaction to be held in conjunction with the 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2012)*, Boston, Massachusetts, USA, March 2012.

[8] Malte Langosz, Lorenz Quack, Alexander Dettmann, Sebastian Bartsch, and Frank Kirchner. A Behavior-based Library for Locomotion Control of Kinematically Complex Robots. In *Proceedings of the 16th International Conference on Climbing and Walking Robots, (CLAWAR-2013)*, Sydney, NSW, Australia, July 2013.

[9] Malte Langosz, Kai Alexander von Szadkowski, and Frank Kirchner. Introducing Particle Swarm Optimization into a Genetic Algorithm to Evolve Robot Controllers. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, pages 9–10, New York, NY, USA, July 2014. ACM.

[10] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep Learning for Detecting Robotic Grasps. *International Journal of Robotics Research (IJRR)*, 34, 2014.

[11] Marc Manz, Sebastian Bartsch, and Frank Kirchner. MANTIS - A Robot With Advanced Locomotion And Manipulation Abilities. In *Proceedings of the 12th Symposium on Advanced Space Technologies in Robotics and Automation. ESA/Estec Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2013)*, Noordwijk, Netherlands, Mai 2013.

[12] Jan Hendrik Metzen, Alexander Fabisch, Lisa Senger, José Gea Fernández, and Elsa Andrea Kirchner. Towards Learning of Generic Skills for Robotic Manipulation. *KI - Künstliche Intelligenz*, 28(1):15–20, 2013.

[13] Sascha Roloff, Stefan Wildermann, Frank Hannig, and Jürgen Teich. Invasive computing for predictable stream processing: a simulation-based case study. In *Proceedings of 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia 2015)*, pages 1–2, Oct 2015.

[14] Jonathon W. Sensinger, Stephen D. Clark, and Jack F. Schorsch. Exterior vs. interior rotors in robotic brushless motors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2764–2770, Shanghai, China, May 2011.

[15] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[16] Martin Zenzes, Peter Kampmann, Tobias Stark, and Moritz Schilling. NDLCom: Simple Protocol for Heterogeneous Embedded Communication Networks. In *Proceedings of the Embedded World Exhibition & Conference. Embedded world Exhibition & Conference, located at Embedded World 2016*, Nürnberg, Germany, February 2016.