TOWARDS A GENERIC NAVIGATION AND LOCOMOTION CONTROL SYSTEM FOR LEGGED SPACE EXPLORATION

Alexander Dettmann[1], Steffen Planthaber[1], Vinzenz Bargsten[1], Raul Dominguez[1], Gianluca Cerilli[2], Marco Marchitto[2], Geoff Fink[2,3], Michele Focchi[2,4], Victor Barasuol[2], Claudio Semini[2], and Róbert Marc[5]

[1] *1Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 28359 Bremen, Germany*
[2] *Dynamic Legged Systems (DLS), Fondazione Istituto Italiano di Tecnologia (IIT), 16163 Genova, Italy*
[3] *Thompson Rivers University, 835 University Dr, Kamloops, BC, V2C 0C8, Canada*
[4] *Università di Trento, Via Sommarive, 9, 38123 Trento, Italy*
[5] *Airbus Defence and Space Ltd., SG1 2AS Stevenage, Great Britain*

ABSTRACT

The exploration of lunar craters is of high interest, but their rugged and inclined terrain also exceeds the mobility capabilities of current rovers, opening up a field of application for legged exploration systems. This paper presents a navigation and locomotion control system that enables legged robots to be able to perceive the terrain, to plan a path to a desired goal, and to control the path execution while traversing unconsolidated, inclined, and rugged terrain. The navigation system is closely coupled with the robotic motion control to be able to exploit the full potential of the flexible locomotion system. In addition, the followed approach introduces a suitable level of abstraction to achieve a modular and generic software that can support different types of walking robots. This will allow, depending on specific requirements of future space missions, to use energy-efficient four-legged as well as more stable six-legged robots. The paper describes the guidance, navigation, and control approach and shows first experimental results.

Key words: **Planetary Robotics, Space Exploration, Legged Locomotion, Guidance, Navigation, and Control** .

1. INTRODUCTION

Moon exploration has come back on agendas after several decades of silence from major space agencies. Both, ESA and NASA, have ambitious plans for the coming years for both, scientific and industrial purposes. Also the Chinese Space Agency launched multiple rover missions in recent years to the Moon. ESA is planning for EMRS (European Moon Rover System) – one rover for multiple use cases (i.e. science of polar regions, deployment of radio telescope on a far side, In Situ Ressource Utilization). Plans from private sector (e.g. ISpace, Astrobotics) include rovers as part of their lander missions for demonstrating various industrial endeavours.

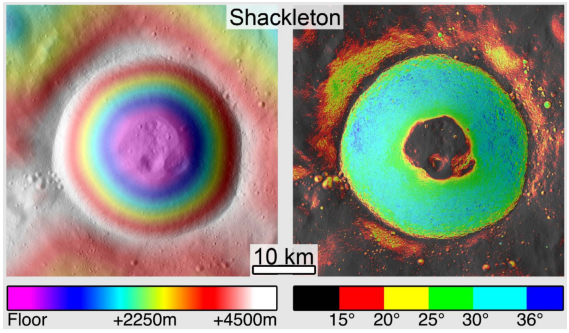


Figure 1: Maps of the Shackleton Crater [18] with a 5 m/px resolution - Left: Colored shaded relief map. Right: Derived slope map.

Craters are one of the most interesting places of the Moon: the majority of them were formed more than two billion years ago and some of them are in permanent darkness being a potential source of water ice. On Moon, the Shackleton crater (Figure 1) is an example of a potential mission for challenging terrain for any exploration system. The crater itself has been formed due to an impact and lies around the lunar south pole. The rims are exposed and are receiving direct sunlight while the interior of the crater receives no sunlight at all in some areas. The average wall slope is 31°, and very rarely exceeds 35° [18]. Both top and bottom of the walls show a gentle change in slope from the 35° wall to the nearly-flat floor over the course of over a kilometer.

However, these places of high scientific and economic interest are hard to explore due to their rugged terrain with many obstacles and high inclines. On Mars, the MSL (Mars Science Laboratory) rover Curiosity has been able to traverse challenging slopes of 20° to 30°)[1]. However, the process from JPL (NASA Jet Propulsion Laboratory) mission planners was to use routes that are similar to ser-

[1] https://astronomynow.com/2020/03/23/curiosity-climbs-its-steepest-slope-so-far

pentines, instead of tackling them directly in the direction of the slope vector. Such approaches have been validated and tested by NASA/JPL on various missions on Mars, but also increase length of the path and thus the time and energy required. Since the demanding mobility requirements to directly ascend along the incline exceed the capability of current exploration rovers, legged systems could be an alternative. Their flexible locomotion system can shift the trunk to increase their stability and their point-contact feet can be used to overcome rocks, or even further, utilize them to increase traction in inclines. On Earth, robots like Spot [2], Anymal [8], Lauron V [13], and HyQ [15] already showed extraordinary mobility in unstructured environment. Also specific developments for space applications such as Spacebok [9] or SpaceClimber [3] are emerging and demonstrate their applicability in fine sand and inclines.

One drawback of legged systems is their complexity in control and their tailored software solutions for a specific hardware. On the one hand, the guidance, navigation and control system for legged robots that is presented in this paper, exploits the potential of legged locomotion to allow the exploration of hard-to-reach areas such as steep and rugged terrain. And, on the other hand, it provides a generic approach that can be applied on systems of different morphologies, i.e. quadrupeds and hexapods. The developed software shall be configured with reasonable amount of effort to the target system that meets the requirements for a specific future planetary mission.

The paper is structured as follows: Section 2 describes the overall generic approach. The main software stacks, the guidance and navigation as well as the motion control system, are described in more detail in Section 3 and Section 4, respectively. Section 5 introduces the hardware abstraction layer that defines the required generic interface for torque controlled robots and shows exemplary system implementations on the hexapod CREX[3] [11] and the quadruped Aliengo[4]. Both legged systems shall be capable of successfully passing mixed slopes, rubble, consolidated rubble piles on inclined terrain, and unconsolidated rubble piles on inclined terrain. Therefore, different demonstration scenarios as analogues for realistic use cases are realized to demonstrate the different mobility requirements. First experimental results are shown in Section 6. The final section concludes the paper and provides an outlook on future activities.

2. OVERALL CONTROL ARCHITECTURE

The navigation system for a legged robot needs to be closely coupled with the robotic motion control to be able to exploit the full potential of the flexible locomotion system. The presented solution is obeying the tight interaction while introducing a suitable level of abstraction to receive a modular and generic navigation and control system that can support different types of walking robots. This will allow, depending on specific requirements of future space missions, to utilize energy-efficient four-legged as well as more stable six-legged robots, the latter featuring additional redundancy and lowering the risk of mission failure in case of damaged actuators.

The general approach is depicted in Figure 2. The *Guidance and Navigation Layer* (NAV) generates a path to a given target based on the current location of the robot and the sensed environmental information that is represented in a sophisticated map. The generated path is then transformed by the *Motion Control System* (MCS) into precisely planned footholds that are reactively adapted to compensate for undetected irregularities. The footholds are used to plan the motion of the robot generating kinematic references for all actuated joints. A *Hardware Abstraction Layer* (HAL) is introduced to define a generic interface for different types of torque-controlled legged robots.

The single software stacks use different Robotic Middlewares: the NAV layer is implemented using ROCK[5], the MCS is implemented using ROS[6], and the driver layer may also be implemented in another framwork. To ease the integration, the Open Source, lightweight, and framework-independent *Robot Remote Control* (RRC) library [5] is used to interface the software modules of the single layers. The RRC library was originally created to define a control and telemetry interface for robots that can be used without a dependency to their Robotic Communication Middleware. But it is also suited to define interfaces inside a single software stack for robot control. Different channels can be used and differently configured to support the demands of commands and sensor messages. This way, a quality of service with acknowledged reception of messages for commands can be realized as well as high data throughput for high frequent or big sensor data.

In order to deploy, store, and update the software in different versions, docker images [12] with a special script collection for integrated development and deployment of docker images[7] are used for the NAV and MCS layers. Both layers are solely interfaced using the RRC library with a specifically designed interface. This means that they can be executed on any system that is able to run docker without further dependencies that have to be installed on the deployed computer.

3. GUIDANCE AND NAVIGATION

To generate the coarse path that should be followed to reach a given target destination, the NAV layer (i) perceives the state of the robot, (ii) senses the state of the environment and represents it in form of a map, and (iii)

[2] https://www.bostondynamics.com/products/spot

[3] https://robotik.dfki-bremen.de/uploads/tx_dfkiprojects/Systemblatt_CREX_en.pdf

[4] https://www.unitree.com/products/aliengo

[5] https://www.rock-robotics.org

[6] https://www.ros.org

[7] https://github.com/dfki-ric/docker_image_development

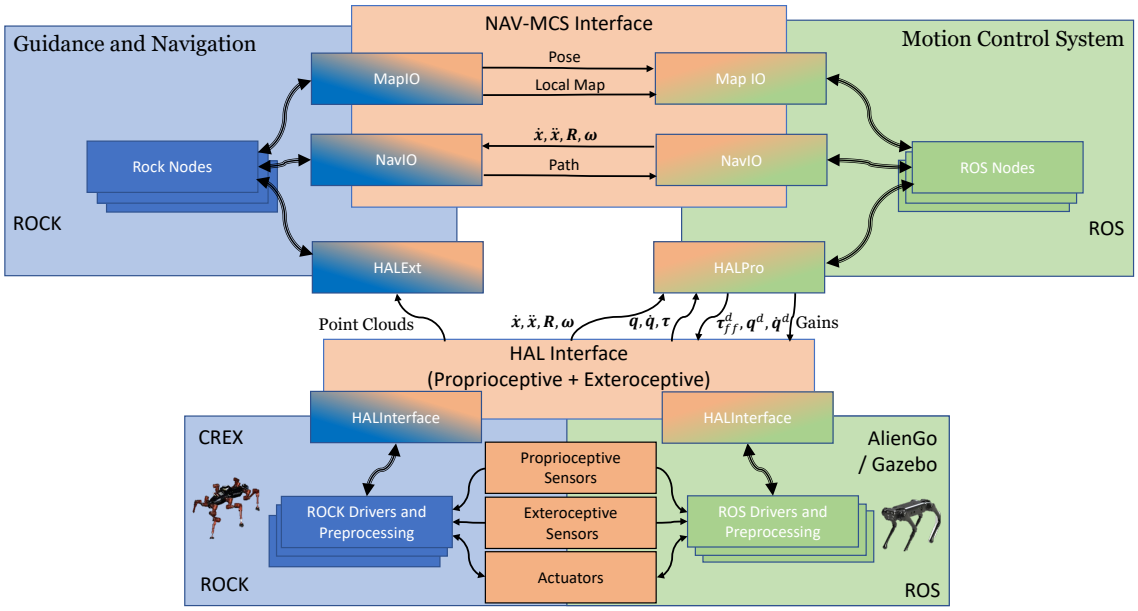


Figure 2: Overall control architecture and interfaces between main layers - The software can be deployed on one or several computers and supports implementations in different Middleware frameworks, in this case ROCK (blue) and ROS (green)

plans a path based on the acquired information. However, to fully utilize the potential of legged systems, besides using a traditional self-localization and mapping approach (SLAM), an additional local SLAM is introduced to provide a high resolution local map with high update rates to allow foothold planning in the control layer. The remainder of this section describes the utilized methods in more detail.

Local SLAM

The local SLAM is calculating a high resolution map that is locally consistent around the robot. It uses a representation of the environment based on Truncated Signed Distance Fields (TSDF) [4] that can efficiently be utilized to generate a grid map of the environment. In order to gain a high resolution map with fast update rates, only the area beneath and around the robot is modeled (Figure 3a). It is sent to the MCS layer to allow precise foothold planning. Another output for the MCS layer is a pose estimation that is based on the sensed robot state coming from proprioceptive data, i.e., linear velocity $\dot{\boldsymbol{x}}$, linear acceleration $\ddot{\boldsymbol{x}}$, orientation $\boldsymbol{R}$ and angular velocity $\boldsymbol{\omega}$, which is then accumulated and corrected through matching the incoming point clouds via ICP (Iterative Closest Point) [14].

Global SLAM

The global SLAM provides a consistent map of the complete area, i.e., handles loop closure and larger maps on the costs of lower update rates compared to the local SLAM. It is used for path planning and is based on *Slam3d*[8], a graph-based SLAM approach that uses ICP to include new point clouds. A global graph optimization [10] is used to correct errors introduced by drifting odometry. The output of the global SLAM is a Multi-Level Surface Map (MLS) [16] (Figure 3b). The MLS is a grid-based map with multiple entries per cell representing the height intervals of occupied volumes in the corresponding region. Every patch includes besides its cartesian position, also a inclination and can be extended to hold additional data such as estimated friction values.

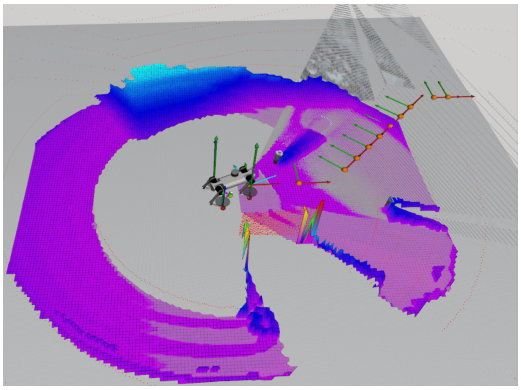
Path Planner

In order to execute path planning, the first step is to generate a 3D traversability map from the MLS output of the global mapper. The traversability map is parameterized by the properties of the robot, e.g. its dimensions, the maximum size of obstacles it can overcome, and the maximum slope it can traverse, and thus can easily be configured for the target application. The traversability map is a neighbor-based 3D representation of the map rather than a grid-based representation and thus better suited for the planning algorithm, as it can traverse directly to the neighbors without calculating them first.

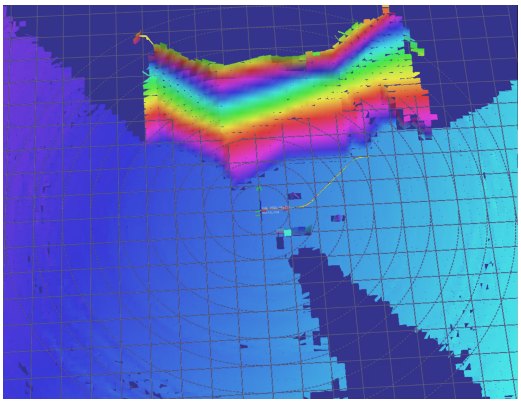
The planning approach allows 3D planning, e.g. planning a path to the same position (x,y) in a different floor (z) (e.g. in a cave or building). The algorithm itself is based on a search based planner (SBPL)[9] that includes the inclination of the terrain in its cost function. Besides using Ackermann-based motion primitives such as longitudinal velocities and turning motions, the planner also supports lateral velocities using the omnidirectional locomotion capabilities of legged systems. When the path planner receives a goal position, a suitable path for the robot is calculated, if possible, and transformed into the

[8] https://github.com/dfki-ric/slam3d

[9] https://github.com/sbpl/sbpl



(a) Local map for planning footholds



(b) Global map for path planning

Figure 3: Local and global map examples produced by the NAV layer

robot's base frame (yellow line in Figure 3b). The resulting path in form of way points, providing positions and desired headings, are sent to the MCS layer (markers in Figure 3a).

Exploration Module

The exploration module is a software that automatically computes intermediate goal positions for the path planner for long distance targets. When a new area has to be explored, the algorithm computes frontier positions of the map and estimates the information gain for the current map. After the next goal position is selected by a metric based on the expected information and the distance to the current position, the path planner is started. When the robot reaches the intermediate goal position, the next goal is generated, and so on, until a given area is either mapped or all unmapped areas are marked unreachable from the current region.

This iterative approach is required for unknown terrain, where no orbital maps can be obtained and the onboard sensors provide just a limited view, e.g. on cliffs, steep craters, or subsurface lava tubes. This component is combined with a re-planning mechanism that is triggered as soon as new map information is available, i.e., through newly discovered areas or changed known areas such as the detection of unstable ground.

4. MOTION CONTROL SYSTEM

The MCS takes care of generating the motion of the legs and to plan and control the robot Center of Mass (CoM) position and trunk orientation. The motion of the legs are generated to connect footholds that are planned based on the navigation references (e.g. a planned path or commanded velocities) and on the knowledge about the terrain and the robot capabilities. The CoM trajectory is planned so that the robot locomotion is statically stable, i.e., the CoM is always inside a support polygon (convex hull formed by the feet in contact). As depicted in Figure 2, the MCS makes use of proprioceptive and exteroceptive feedback. The proprioceptive feedback includes the actual robot joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, and torques $\boldsymbol{\tau}$, as well as measured body velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$, orientation $\boldsymbol{R}$, and angular velocity $\boldsymbol{\omega}$. As exteroceptive feedback, MCS receives a local terrain map around the robot. As output, the MCS provides the low-level control with desired joint positions $\boldsymbol{q^d}$, velocities $\dot{\boldsymbol{q}}^{\boldsymbol{d}}$, gains (proportional and derivative), and torques $\boldsymbol{\tau}_{\boldsymbol{ff}}^{\boldsymbol{d}}$.

The following subsections present a brief description of the main MCS modules used for motion generation and control and to add robustness to the locomotion on rough terrains.

Gait generation

The gait generation module produces a statically-stable crawl gait [6]. A state machine decides which leg is to be swung during locomotion. It was advanced to support walking gaits for quadrupeds and hexapods. The crawl is typically divided in two main phases called *swing phase* and *move-base phase*. During the swing phase, the robot does not move its trunk and only one foot at the time is allowed to lift off from the ground and move to a new foothold, while all the other feet have to be in stance. During the move-base phase, instead, all the feet are in stance and the robot moves its trunk to a target location and orientation. Before each move-base phase, a re-planning of the CoM and orientation trajectories is performed, considering the actual configuration of the robot. This allows the robot to recover from tracking errors that might appear during the motion, e.g. due to a loose rock collapsing under the feet. The swing trajectories connect the stance feet positions with the corresponding nominal footholds, computed from the desired user speed, the terrain inclination and terrain-map information.

Map-Based Terrain Adaptation

The swing trajectory of each leg is planned according to a desired foothold. When the robot is walking blindly, this desired foothold is called the nominal foothold, which is computed purely based on the desired robot velocity and the proprioceptive terrain estimation (using legs in contact, joint positions and the sensed trunk motion). When a terrain map is available around the nominal foothold, such portion of map is assessed to classify each candidate location as either a safe or unsafe foothold. To do so, the Visual Foothold Adaptation (VFA) technique pro-

posed in [17] was implemented. This technique assesses the terrain according to criteria like:

- leg workspace limits: the foothold must lie inside the leg workspace along the whole stance phase,
- leg collision: avoid leg collision with the environment during the whole leg swing and stance phases,
- terrain roughness: avoid locations that are close to discontinuities in the terrain surface.

After assessing every candidate location in the map patch, the closest to the nominal foothold is chosen as desired foothold.

Surface Reaching and Haptic Touch-down

The desired footholds are computed either using the information from the terrain map or, if only proprioceptive feedback is available, using a planar approximation of the terrain. Although the data from the map is expected to be more accurate, both sources of feedback contain inaccuracies that prevent a perfect contact match with the surface. Such inaccuracies might cause the leg touch-down event to happen earlier or not at all. Therefore, a module that adapts the leg swing to establish a contact with the surface is crucial for rough terrain locomotion. To adapt to the environment, two strategies are applied. First, a reaching motion strategy to search for the contact extends the foot trajectory along the terrain normal until a contact is firmly established. And second, a haptic touch-down strategy makes use of contact sensors to interrupt the swing motion of the foot in the case of an early touch-down event, thus preventing destabilizing forces on the trunk.

Whole-body Controller

To stabilize the robot trunk and drive its CoM to the planned positions, the whole-body controller (WBC) presented in [6] was implemented. The WBC optimizes the joint torques that lead to a desired stabilizing control action (forces and moments on the CoM). To do so, it considers the legs in stance phase, robot physical consistencies, and the constraints on the ground reaction forces (GRFs) due to surface inclinations and friction properties.

Terrain load-bearing assessment

One of the most challenging requirements in designing a locomotion strategy for terrain exploration is to make it robust against collapsing terrains. First, the robot must be able to select stable contact points and, second, recover in case of unexpected ground collapses. The strategy to probe and check, whether a given foothold is likely to be stable, consists of three main aspects: (i) maintain the robot CoM always inside a stable support polygon that does not depend on the foothold under probing; (ii) concentrate as much as possible the distribution of the ground reaction forces on the probed foothold; and (iii) the probing forces must be higher than the forces exerted at the foothold during nominal locomotion. The overloading of the probed foothold and the GRFs distribution during the probing phase is obtained through the WBC by providing

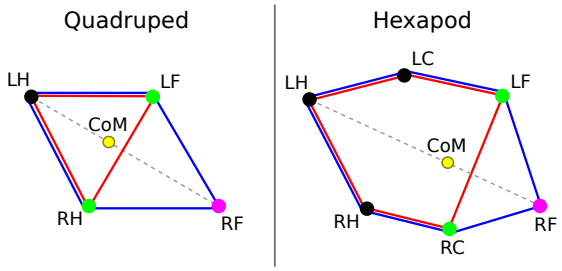


Figure 4: Example and key components of the load-bearing assessment strategy. Circles represent the robot CoM (yellow) and the footholds associated to each foot in stance: probing foothold (magenta), opposite footholds (black), and underloaded footholds (green). The dashed gray line represents the opposite diagonal over which the CoM is positioned. The blue polygon is the *target polygon*. The red polygon is the *safe polygon* that is used in case the probing foothold collapses during assessment.

a probing wrench and adjusting the penalization and limits associated to the forces exerted by each leg. Figure 4 illustrates the key components involved during the load-bearing assessment for quadrupeds and hexapods.

The assessment procedure starts with the robot moving its CoM onto the line connecting the foothold to be probed and its most-opposite foothold. This line is called *opposite-diagonal*. The CoM position on the opposite-diagonal is computed so that the CoM remains as close as possible to the probed foothold while being inside the *safe polygon* and at a minimum distance to the closest polygon edge. The safe polygon is the convex hull formed by all foot contact positions excluding the one associated to the probed foothold. Once the CoM is positioned, the leg swings until the foot touches the ground at the selected foothold to be probed. This CoM positioning strategy allows the WBC to concentrate the GRFs on the probed foothold by penalizing the load at the other footholds. Underloaded footholds are barely loaded and are mainly used to keep the robot stable. Opposite footholds share almost the whole robot weight with the probed foothold and contribute to render the probing force. The probing force is a resulting pulsed force at the probing foothold from a desired pulsed moment around the CoM that is perpendicular to the gravity vector and to the opposite diagonal.

During the probing phase, the surface is considered to be *collapsed* in two cases: if the corresponding foot contact is not detected anymore; or if the mobility index [19] of the corresponding leg reaches a given minimum value (due to a leg extension caused by surface deformation). If the probed foothold fails, i.e. a collapse is perceived, the foot contacts that compose the *safe polygon* are used to recover the robot stability. After the stability recovery an algorithm selects an alternative foothold to be probed. Instead, if the probed foothold succeeds, the assessment procedure is finished and all the foot contacts are equally exploited to generate GRFs and move the robot CoM to a position inside the *target polygon* (convex hull formed

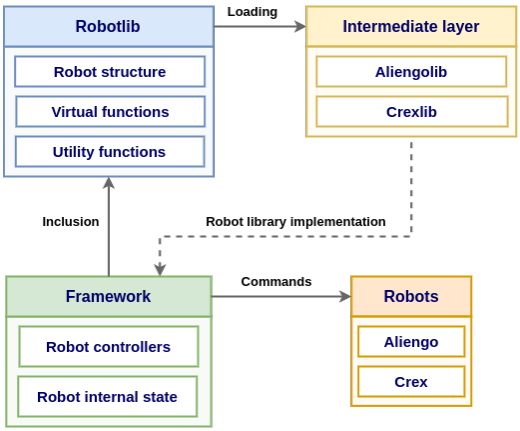


Figure 5: Robotlib loads specific robot libraries and provides generic access to their implementations.

by all the foot contacts).

Robotlib

The proposed locomotion control system is based on a modular and generic software framework that can be applied on robots of different morphologies, such as quadrupeds and hexapods. This software abstraction is achieved by *Robotlib*, a robot software interface that allows the same locomotion framework to run on many different robots. Robotlib provides a templated robot morphology that defines the hierarchical structure of joints and links. Additionally, it provides virtual and utility functions for the robot kinematics, dynamics, and Jacobians. Robot-specific libraries inherit the base robot interface to implement the particular morphology, kinematics, and dynamics of each different robot. In particular, we implemented libraries for Aliengo and CREX called Aliengolib and Crexlib, respectively (Figure 5).

There are several advantages of using an architecture abstraction layer like Robotlib. For example, with Robotlib, robot-specific structures are hidden from the controllers and state estimators, making them more modular and easier to implement. The structure allows controllers and state estimators to be written only once, and then the framework can dynamically load different robots. The abstraction layer also provides an easy way to switch backend libraries that compute the kinematics and dynamics without affecting the rest of the framework. Robotlib is written in C++17 to be fast and portable. It is compatible with the most adopted robotics libraries and is real-time safe.

5. HARDWARE ABSTRACTION

The HAL is introduced to define a generic interface for different types of torque-controlled legged robots. It defines the minimum sensor suite required by the framework. Each robot needs to be equipped with both, exteroceptive and proprioceptive sensors. The exteroceptive sensors provide point clouds of the environment, and the proprioceptive sensors provide body orientation $\boldsymbol{R}$, angular velocity $\boldsymbol{\omega}$, linear velocity $\dot{\boldsymbol{x}}$, and linear acceleration $\ddot{\boldsymbol{x}}$, as well as joint position $\boldsymbol{q}$, velocity $\dot{\boldsymbol{q}}$, and torque $\boldsymbol{\tau}$. It accepts desired joint positions $\boldsymbol{q^d}$, velocities $\dot{\boldsymbol{q}}^{\boldsymbol{d}}$, and feed forward torques $\boldsymbol{\tau}_{\boldsymbol{ff}}^{\boldsymbol{d}}$ as commands. In addition, an interface to adapt the respective control gains is provided.

The HAL also uses the RRC library to interface the actual hardware of the system to the NAV and MCS layers. Two instances of the library are used to separate the proprioceptive and exteroceptive sensors. This approach avoids interference of bigger and smaller data packages on the same ethernet ports, e.g., crucial high frequency joint values and commands are separated from large point clouds. This way, the required quality of service is reached and the control loop of the control layer can run with a control frequency of 1000 Hz. To test the generic guidance, navigation, and control system, two experimental platforms of different morphologies are utilized, which are described in more detail in the following.

Experimental Hexapod – CREX

CREX (CRater EXplorer) is a biologically inspired six-legged walking robot that weighs 27 kg at a nominal size of 0.8 m x 1.0 m. It is the successor of SpaceClimber [3] having the same morphology but updated sensors and electronics. CREX was designed to explore steep craters and rough terrain on rigid and loose surfaces. Every leg has four joints, each of them being equipped with position, velocity, current, and temperature sensors as well as with additional electronics and an FPGA to locally control the actuator, thus reducing the computational load of the main processing units. Each joint consists of an ILM50x8 brushless dc motor, and a 1:100 harmonic drive resulting in a maximum torque of ca. 28 Nm. In addition, the system is equipped with an embedded PC, running the navigation stack, and an Intel NUC, running the MCS as well as the robot-specific low-level drivers and controllers accessible through the HAL interface.

For the proposed control scheme of the MCS to be applicable on the CREX system, the low-level actuator controllers have been extended to achieve the required torque-based joint control. As these actuators do not possess dedicated joint torque sensors and in addition have a high gear ratio, a low-level controller based on motor current in combination with experimentally identified models of the leg dynamics and joint friction has been implemented based on the developments in [1, 2]. In particular, the joint friction can be compensated, directly using the electronics integrated into the actuators. Additionally, the six degrees of freedom force-torque sensors mounted between each leg and trunk are used to estimate the external forces acting on the feet, taking into account the leg mass and configuration. They are used together with the reading of the inertia measurement unit to implement a contact-based odometry [7] to provide the required state estimation.

Visual feedback is mainly provided by a Velodyne Puck VLP-16 that provides 360°-point clouds of the environ-

ment. In addition, two PicoFlexx time-of-flight cameras are attached to the head to provide additional depth information of the occluded region in front and beneath the robot.

Experimental Quadruped – Aliengo

The second experimental platform is the 21 kg quadrupedal robot Aliengo. This robot has 12 high-performance servo motors that allow it to achieve motions like forward, backward and side movements, jumping, running and walking at 1.5 m/s speed. In addition to this, its structure makes it suitable for tasks where the traversability of rugged and inclined terrains is crucial. The robot is equipped with a MiniPC running Ubuntu 16.04 with a real-time kernel, to which the high-level SDK communicates with. Two Intel Realsense cameras, the Tracking T265 and the Depth D435 ones, are attached to the MiniPC to get visual odometry-based state estimation and the point clouds of the environment, respectively. The robot has also a Jetson TX2 with Ubuntu 18.04, that is interfaced with another Intel Realsense Depth D435 camera, and a Controller Board used to run the Unitree locomotion controller to which the low-level SDK communicates with. An Ethernet switch allows all these components to communicate with each other. In order to improve the on-board computational power, an Intel NUC running Ubuntu 20.04, with an Intel i7-10710U CPU, has been additionally mounted on the robot and connected to the switch. This pc is used to run the locomotion and navigation frameworks. A Velodyne Puck LITE lidar has been mounted and connected to it, so that both the MiniPC camera and lidar pointclouds can be used for the path planning.

The robot control interface supports both C++ and ROS and the HAL is based on the Unitree legged SDK interacting with the Controller Board through UDP communication. The HAL is also using the RRC library to receive joint commands and send robot states and point clouds to the MCS and NAV stack, respectively.

6. EXPERIMENTAL EVALUATION

The development is divided into successive sprints, each providing a complete software stack to accomplish the traversal of terrain with increasing difficulty. First, traversing slopes was targeted. Second, tests to overcome rubble fields were conducted.

Walking in Inclines

In order to asend or descend steep craters, omnidirectional walking in inclines is a mandatory capability. In order to show omnidirectional motions, both systems were placed on adjustable ramps which they had to traverse along the slope upwards and downwards as well as sideways across the slope and diagonally. In every direction, they had to walk in line of direction as well as laterally and to turn on spot. This way, the ASTM standard test method for evaluating emergency response robot capabilities on mobility[10] were obeyed. Both systems, showed that climbing 30° slopes is possible, mainly limited by the friction between feet and surface. While Aliengo is flexible in all directions, CREX showed a limited range of motion when going laterally up or down the slope.

(a) CREX walking in DFKI's artificial lunar crater (b) Aliengo walking a 30° incline in the DLS Lab

Figure 6: Walking in inclined terrain

(a) CREX walking over a rubble field (b) Aliengo inside the Mars Terrain Simulator of ALTEC

Figure 7: Walking over unstructured terrain

Walking on Rubble Fields

Rubble fields is one particular terrain that reveals the benefits of legged systems. During the experiments, both systems showed that they can easily traverse obstacle heights of one third of the leg length, which is for both systems approx. 17 cm. The haptic terrain adaptation stabilizes the system on irregular terrain with sharp edges and stones, where wheel-driven systems can easily get stuck. Also loose rocks that move when stepping on them did threaten the system to the continuous control of the trunk's stability.

7. CONCLUSION AND OUTLOOK

The presented guidance, navigation and control approach for legged systems was successfully implemented on two robots, the hexapod CREX and the quadruped Aliengo. Key enablers for the rapid deployment are the (i) generic implementations of the NAV and MCS algorithms that can be configured for the target systems, (ii) a docker-based deployment of the single software stacks, and (iii) the framework-independent communication between the software layers.

[10] https://www.astm.org/e2803-11r20.html

Both systems showed high mobility in rough and inclined terrain. Mainly the torque-based control with haptic adaptation to the terrain guarantees stable locomotion, but the boundaries have not been exploited so far. In future, the vision-based adaptation of footholds and load-bearing assessment will be tested in rough inclines with and without load bearing issues. The goal is to demonstrate that legged robots can reliably be used to explore craters with rough inclines up 35°. Thus, unstructured slopes with and without load bearing issues are targeted next.

ACKNOWLEDGMENTS

The presented work is carried out in the frame of the ESA funded project ANT (ESA AO/1-10289/20/NL/RA) with the goal to develop a guidance, navigation and control system for legged robots that is capable of traversing unconsolidated rubble piles that are to be expected at the base of cave skylights or cliffs.

REFERENCES

[1] V. Bargsten and J. de Gea Fernández. COMPI: Development of a 6-DOF compliant robot arm for human-robot cooperation. In *Proceedings of the 8th International Workshop on Human-Friendly Robotics*, 2015.

[2] V. Bargsten and J. de Gea Fernández. Distributed computation and control of robot motion dynamics on FPGAs. *SN Applied Sciences*, 2020.

[3] S. Bartsch. *Development , Control , and Empirical Evaluation of the Six-Legged Robot SpaceClimber Designed for Extraterrestrial Crater Exploration*. PhD thesis, University of Bremen, 2013.

[4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[5] L. C. Danter, S. Planthaber, A. Dettmann, W. Brinkmann, and F. Kirchner. Lightweight and framework-independent communication library to support cross-plattform robotic applications and high-latency connections. In *International Symposium on Systems, Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*, 2020.

[6] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini. *Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality*. Springer International Publishing, 2020.

[7] R. Hartley, M. Ghaffari Jadidi, J. Grizzle, and R. M Eustice. Contact-Aided Invariant Extended Kalman Filtering for Legged Robot State Estimation. In *Robotics: Science and Systems XIV*, 2018.

[8] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, et al. Anymal-toward legged robots for harsh environments. *Advanced Robotics*, 2017.

[9] H. Kolvenbach, P. Arm, E. Hampp, A. Dietsche, V. Bickel, B. Sun, C. Meyer, and M. Hutter. Traversing steep and granular martian analog slopes with a dynamic quadrupedal robot. *arXiv preprint arXiv:2106.01974*, 2021.

[10] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[11] J. Machowinski, A. Bckmann, S. Arnold, C. Hertzberg, and S. Planthaber. Climbing steep inclines with a six-legged robot using locomotion planning. In *International Conference on Robotics and Automation. IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[12] D. Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014.

[13] A. Rönnau, G. Heppner, M. Nowicki, and R. Dillmann. Lauron v: A versatile six-legged walking robot with advanced maneuverability. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2014.

[14] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, 2001.

[15] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 2011.

[16] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. 2006.

[17] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini. Fast and continuous foothold adaptation for dynamic locomotion through cnns. *IEEE Robotics and Automation Letters*, 2019.

[18] R. Wagner, M. Robinson, E. Speyerer, and P. Mahanti. Topography of 20-km diameter craters on the moon. In *Lunar and Planetary Science Conference*, 2013.

[19] T. Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 1985.