Towards an Autonomous Walking Robot for Planetary Surfaces

Martin Görner, Annett Chilian, Heiko Hirschmüller *

* DLR German Aerospace Center, Institute of Robotics and Mechatronics, Germany
e-mail: martin.goerner@dlr.de, annett.chilian@dlr.de, heiko.hirschmueller@dlr.de

Abstract

In this paper, recent progress in the development of the DLR Crawler - a six-legged, actively compliant walking robot prototype - is presented. The robot implements a walking layer with a simple tripod and a more complex biologically inspired gait. Using a variety of proprioceptive sensors, different reflexes for reactively crossing obstacles within the walking height are realised. On top of the walking layer, a navigation layer provides the ability to autonomously navigate to a predefined goal point in unknown rough terrain using a stereo camera. A model of the environment is created, the terrain traversability is estimated and an optimal path is planned. The difficulty of the path can be influenced by behavioral parameters. Motion commands are sent to the walking layer and the gait pattern is switched according to the estimated terrain difficulty. The interaction between walking layer and navigation layer was tested in different experimental setups.

1 Introduction

The exploration of planetary surfaces with mobile robots is a very challenging task because long signal round trip times prevent the robots to be remotely controlled from earth. Thus, they must be able to navigate and fulfill tasks autonomously in an unknown and unstructured environment. Possible tasks are sample collection, manipulation and analysis as well as mapping of interesting sites. Promising areas for retrieving useful samples are craters and canyons. Such areas are usually very difficult to traverse due to steep slopes, changing substrates and rugged ground. In this kind of environment walking robots are expected to show superior performance to wheel driven rovers. Their advantages are, no need for paths of continuous contact with the ground and the ability to step over or on obstacles as well as to climb various rock formations. However, walking robots suffer from limited payloads, which prevents them from carrying heavy instruments. Thus, using a group of robots consisting of a large wheeled rover for supply and long distance transport and a team of highly mobile legged robots for local exploration tasks seems to be a very promising solution. Such a configuration poses many challenging problems of which walking itself, crossing obstacles and autonomous navigation are the most immediate.

In our opinion an exploration robot should use a layered control architecture [1] consisting of a walking layer, a navigation layer and a high-level task planner. The walking layer should be responsible for the basic tasks of stable standing and walking, but also provide reflexes for non-flat terrain to overcome obstacles within the walking height reactively. The walking layer should only use proprioceptive sensors. The task of the navigation layer is to guide the robot along an optimal path to a goal point. For this, motion commands have to be computed and sent to the walking layer. Furthermore, the walking layer could send feedback data to the navigation layer, e.g. containing information about how often reflexes are triggered as a measure of the ground properties. The navigation layer needs to use sensors for perceiving the environment in order to detect obstacles and plan paths. On the top level, a task planner needs to generate tasks in order to fulfill the mission goal. For example, the task planner should compute goal point coordinates for the navigation layer and adjust behavioral parameters according to the robot's state. That means, if the robot is carrying heavy loads, the task planner is in charge of adjusting the path planning parameters so that the navigation layer generates only easy paths. Furthermore, if the navigation layer reports that the desired goal point cannot be reached, the task planner has to choose a different goal point. A similar concept of a gait control system and an autonomy and perception system was used in the four-legged robot BigDog [14].

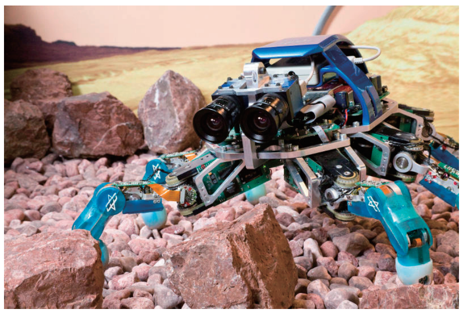


Figure 1: The DLR Crawler within the gravel testbed

In this paper we present the recent progress in the development of the DLR Crawler (Fig. 1) - a six-legged walking robot prototype with a walking layer employing a biologically inspired gait and a navigation layer, which enables the robot to autonomously find a path to a predefined goal point. The paper is organized as follows. Section 2 gives a system overview and describes the hardware and control of the DLR Crawler. In section 3 the implemented walking algorithms are explained and section 4 presents the stereo vision based navigation algorithm. Experimental results are evaluated and discussed in section 5 and comments on future work are given in section 6.

2 System Overview

The DLR Crawler is a prototypic, six-legged, actively compliant walking robot that is based on the fingers of DLR Hand II. It is a study for future exploration robots that is intended to be used as laboratory testbed for the development of gait and navigation algorithms. Following, a brief overview of the hardware and the control algorithm is given while a detailed description can be found in [7].

2.1 Hardware

The DLR Crawler has six equal legs with a length of 155 mm each, a total mass of 3.5 kg and is able to carry a payload equal to its own mass. It is symmetric to its longitudinal axis and its feet span an area of 350 mm x 380 mm in a common configuration while its body stands within 10 mm and 120 mm above ground. Each leg has four joints and three active degrees of freedom (DOF). Two DOF are realized within the differential base joint while the third DOF results from the mechanical coupling of the medial and the distal joint. All joints are driven by permanent magnet synchronous motors in combination with harmonic drive gears and a tooth belt transmission.

The Crawler hosts a variety of proprioceptive sensors. Within each joint these are, a motor angle sensor, a link side joint angle sensor as well as a joint torque sensor. Additionally, each foot hosts a 6 DOF force-torque sensor and the body implements an inertial measurement unit (IMU). For the purpose of visual odometry and vision based navigation a stereo camera head is mounted.

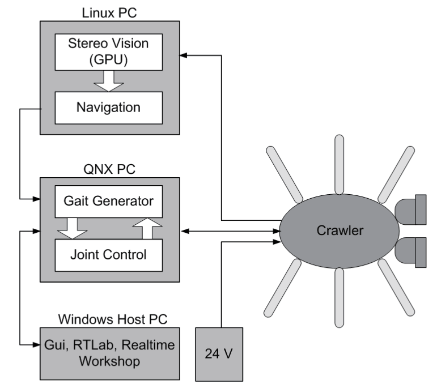


Figure 2: System setup of the DLR Crawler

Since the robot is a laboratory testbed, all control computation is done externally on a QNX realtime PC while the vision employs an external Linux computer (Fig. 2). This allows to quickly test different algorithms with varying computational complexity without caring about optimized implementation on specific on-board hardware at this stage. A 1 kHz control cycle is guaranteed by employing a fast IEEE 1355 based hierarchical serial communication system that uses FPGA nodes to collect and transmit all sensor data and control commands. Further, the robot has an external 24 V power supply and on-board power distribution.

2.2 Control

For joint control, the DLR Crawler employs an active joint compliance control algorithm. This algorithm acts like a virtual spring-damper system within the joint and, thus, allows to change the joint stiffness by means of control. On the upper level the control algorithm is similar to a PD control on joint position and velocity. But on the lower level it additionally implements a high gain joint torque control with friction compensation using the link side joint torque measurements. To initiate the motion of a joint its virtual equilibrium is shifted and, depending on the stiffness settings and the loading conditions of the Crawler, the joints will follow. The advantage of this control approach is its robustness allowing some deviation from the desired trajectory and reducing internal forces due to modeling errors. Using the large set of force-torque sensors, also Cartesian impedance control algorithms can be implemented on either leg or body level. Thus, in future, with a tool attached, a limb or even the body could be used as force controlled manipulator.

3 Walking Algorithm

To operate in various unknown environments a walking robot should only rely on information gained by its own proprioceptive sensors as well as its cameras. Since there is neither complete knowledge of the terrain nor the chance to foresee each possible situation the robot might encounter, a gait algorithm needs to be very flexible, adaptive and robust. Furthermore, it needs to ensure the robot's stability at all times. In order to achieve these goals of flexibility, stability and robustness, a variable gait coordination needs to interact with a multitude of local leg and joint reflexes. This interaction should generate the appropriate behavior to autonomously handle slopes and steps as well as obstacles and holes within the walking height. One goal of the gait algorithm is to reduce the load of the navigation layer that should only deal with large obstacles and a general terrain assessment but should not care about foothold and step planning during normal operation. A footstep and pose planner should only be invoked by the navigation layer in very challenging terrain that cannot be handled by a reactive gait.

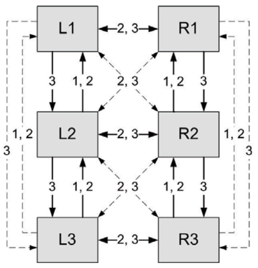
Since not all terrain is challenging, two different gait algorithms are implemented on the Crawler that differ in their capability but also in their computational complexity. The first gait is a tripod for moderate terrain with an underlying fixed coordination pattern requiring little computational power. The second gait, based on a biologically inspired, variable coordination and multiple reflexes, is more complex but allows to handle more challenging terrain and is even able to handle leg loss. The following sections will explain the gait algorithms. A more detailed description can be found in [8].

3.1 Tripod

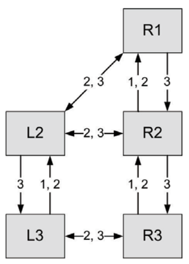
The tripod gait is the fastest gait observed on insects in nature. The main characteristic of this gait is that the front and hind leg on one side as well as the middle leg on the other side perform a step at the same time. The three remaining legs in ground contact form a stable tripod and push the body forward. Thus, all legs follow a fixed coordination pattern. The tripod implementation of the Crawler allows the robot to walk either forward, backward, sideways or to turn on the spot and to switch in between these at a central posture. The foot trajectories are calculated with respect to a body fixed frame using fourth order polynomials to allow smooth swing-stance transitions. They are normalized and stored in look-up tables allowing to vary speed, step height and step length. Using the tripod gait, the posture can be varied simultaneously and a stretch reflex, explained in section 3.3, can be used.

3.2 Biologically Inspired Gait

The biologically inspired gait of the DLR Crawler is centered around inter-leg coordination mechanisms that biologists derived from stick insect experiments [3]. These mechanisms, shown in Fig. 3, act in between neighbored legs and excite or inhibit steps depending on the states of the legs. Mechanism 1 is directed towards the front of the Crawler and inhibits the start of the return stroke of a leg as long as the posterior leg performs a return stroke. Mechanism 2 also acts towards the front of the Crawler and additionally couples contralateral legs. It excites the return stroke of a leg for a certain time after a posterior leg finished its return stroke. Mechanism 3 is directed towards the rear of the Crawler and also couples contralateral legs. This mechanism excites the start of a return stroke of a leg with increasing strength the closer an anterior leg approaches its posterior extreme position (PEP). All mechanisms alter this PEP of a leg, which is the transition from power to return stroke. Thus, the power stroke of a leg can either be extended or shortened depending on the states of its neighbors. In our implementation the PEP can be imagined as a dynamically extending or shrinking cylinder centered in the Cartesian leg workspace, within which the leg can move freely during power stroke according to a desired body motion. Once it leaves the cylinder, it has to initiate a step. By this a coordinated gait emerges depending on the commanded velocity and walking direction. With increasing velocity this gait moves from just one leg stepping at a time via a tetrapod towards a tripod gait. Employing this coordination, it is possible to walk along continuously curved paths since forward and sideways walking as well as turning can be combined to any desired motion. The algorithm will decide itself at which time each leg needs to step. Important to the gait is that all legs step at an equally high velocity which poses a limit to the achievable walking speed. One interesting feature of the gait is its adaptability to leg loss scenarios as can be seen in Fig. 4. As soon as a leg is damaged or lost completely it is removed from the coordination network and previously suppressed inter-leg connections are activated. This results in immediate adaptation to the new situation and allows the Crawler to proceed with only five or in some cases even four legs just at a reduced speed. More details on this adaptability can be found in [6]. One underlying assumption is that the damaged leg does not remain in an outstretched locked configuration obstructing the overall motion. This assumption is valid since it is unlikely that all joints of a leg fail at the same time and due to the fact that the joints are backdrivable. Another nice feature of this algorithm is that it can easily be extended towards eight-legged robots, which has been shown in simulations. In addition to the stretch reflex used with the fixed tripod pattern, the biologically inspired gait allows to use the elevator and search reflexes described in the following section.



(a) All mechanisms



(b) Loss of L1

Figure 3: Inter-leg coordination mechanisms: solid arrows - active, dashed arrows - suppressed

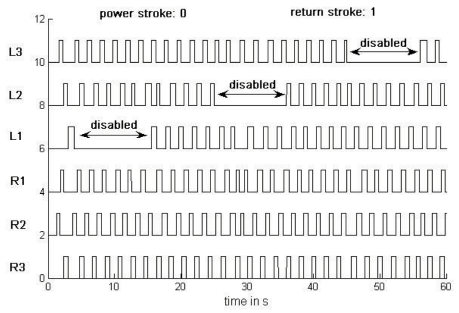


Figure 4: Emerging gait patterns in case of loss of the left front leg (L1), left middle leg (L2) or left hind leg (L3)

3.3 Reflexes

In order to autonomously and stably negotiate mid-size obstacles and holes which are within the walking height of the Crawler, reactive behaviors are needed. These are different reflexes that adjust the posture of the Crawler or react to collisions during stepping motions.

The first reflex of the Crawler is the above mentioned *stretch reflex*. The purpose of this reflex is to enforce the ground contact during the power stroke of a leg. If after a step the leg does not hit ground at the anticipated height or the leg looses ground contact due to a rolling stone, the reflex gets activated and tries to find contact by quickly extending the leg. It is triggered using torque thresholds of the proximal and medial joints and is switched off if the leg achieves a certain load or reaches some kinematic limit. If no contact is found, the reflex shuts off and activates the search reflex.

The *search reflex*, currently only used in simulations, is the second reflex, that tries to find a foothold by searching at distinct locations within the workspace. If this reflex finds some contact and can successfully load the leg then the robot can proceed. If the robot cannot find a foothold some more advanced searching behavior has to be activated that shifts the robot's body in order to extend the reach of the foot. If this is not successful, the navigation layer needs to command the robot to reverse its walking direction and to try to find a different path.

The third reflex is the *elevator reflex* that is triggered once a stepping leg hits an obstacle. This reflex monitors all joint torques and gets activated after some thresholds are passed. In this case it retracts and raises the leg in order to surpass the obstacle. If an obstacle is too high and a leg reaches its kinematic limits trying to step over it then the leg is retracted and placed on the ground in front of the obstacle. In this case another behavior or the navigation algorithm must command the robot to walk into the opposite direction to circumvent the obstacle.

These three reflexes in combination allow the Crawler to autonomously negotiate most obstacles within its walking height and to adapt to rough and uneven terrain. It also allows the robot to master transitions from flat to sloped terrain or transition edges from upward to downward slopes that require different leg extensions and adjusted body postures. The necessary adjustments are solely achieved by the interplay of the reflexes without active balancing based on IMU data.

Most of the reflexes require a flexible gait coordination since they lead to extended power stroke phases of some legs or even cause the whole robot to stop if the situation cannot be resolved fast enough. Thus, all reflexes together with the biologically inspired gait give the Crawler the highest capability to master rough terrain. Nevertheless, the use of the tripod pattern together with the stretch reflex is also an interesting option for easy terrain due to its achievable speed and low computational complexity.

4 Navigation Algorithm

Autonomous navigation requires a robot to continuously estimate its current position in the environment and to plan and follow a path to a predefined goal point. Unknown unstructured terrain poses several challenges to these tasks: First of all, the robot cannot rely on odometry measurements because slip is very likely to occur. Furthermore, the robot has to estimate the traversability of the terrain which for rough terrain varies continuously between easily traversable and untraversable. Finally, the path planner must be able to cope with frequent map updates and fuzzy obstacle descriptions. The navigation layer of the DLR Crawler overcomes these problems. It uses a stereo camera as primary sensor. Stereo cameras are well suited for use in mobile robotic applications because they are light weight, passive sensors.

From each stereo image pair a depth image is computed. Based on the depth image and on natural image features, the position of the Crawler is estimated by visual odometry. A terrain model is incrementally built from the depth images and position data. This model is used to estimate the traversability of the terrain according to the robot's abilities. A path planner computes optimal paths based on the traversability map and sends motion commands to the walking layer. The following sections will explain the navigation algorithm in more detail. For further information on the navigation algorithm refer to [2].

4.1 Computation of Depth Images

Until now, the computation of depth images was done by a correlation based multiple-window approach [11]. However, we have recently switched to Semi-Global Matching (SGM), because this method is more accurate at fine structures and object borders. Furthermore, the depth images of SGM are denser, especially, in weakly textured areas, which is helpful in the current application.

SGM [9] is based on the idea of pixel-wise matching, supported by a global cost function. The matching cost is computed either by Mutual Information or Census [12] for robustness against radiometric differences, which are unavoidable in practice. The cost function is optimized pathwise from eight directions, symmetrically through the image. This can be computed efficiently. Furthermore, the basic algorithm is very regular and only contains simple operations like comparisons and additions. This allows an implementation on the CPU using vector commands for speeding up computation. However, SGM is also quite suitable for an FPGA implementation [5], which is very attractive for mobile systems.

At the moment, the SGM implementation on the Crawler is based on the graphics card [4], because the Crawler prototype is currently connected via cable to a standard desktop computer. The implementation computes almost 5 frames per second in VGA resolution with a large disparity range of 128 pixels. Much higher frame rates can be reached with reduced resolution or disparity ranges. An FPGA processing solution is anticipated in future, for fully autonomous robots.

4.2 Visual Odometry

The image of the left camera and the depth image are used for computing the relative movement of the robot via visual odometry. The method performs corner detection using Harris corners in subsequent left images [10]. In contrast to most other visual odometry methods, features are not tracked. Instead, each corner of one image is compared with all corners of the subsequent image. This allows large movements, i.e. either fast movements or low frame rates. The comparison of corners is done by Rank correlation [15]. Thereafter, corners are reconstructed using the depth image of SGM. An outlier detection, based on pairwise comparison of relative distances between two points, selects only those correspondences that fulfill the rigidity constraint, which is valid for static scenes. Small dynamic objects are tolerated and ignored as noise.

The relative movement is computed with 6 DOF in closed form as the transformation between the corresponding point clouds P and C. Finally, a refinement optimizes the transformation by minimizing the image based reprojection error ε_i of all corners. The relative transformation between subsequent camera coordinate systems is described by a rotation matrix R_r and a translation vector T_r as

$$P_i = R_r C_i + T_r + \varepsilon_i. \tag{1}$$

Summing up all relative movements gives the absolute transformation R, T between the start camera coordinate system and the current camera coordinate system.

The method is quite fast and would be able to compute the visual odometry on a standard computer in frame rate if the depth image was available at that rate. However, since feature tracking is avoided, the method is also able to work at low frame rates like in the current application.

An extension of the original method [10] computes the relative movement not only to the previous image, but independently to a list of n previous images. This results in n new position estimates. A multi-modal median decides which of the n estimates is the best one. The list of n images is limited to a small number like 4-8 for making the computation time predictable. The current image, as well as the oldest image, from which a motion can be computed to the current image, is always stored in the list. Additionally, a heuristic adds *dissimilar* images *in between* the newest and oldest, depending on their time stamp and number of features for computing the motion to the current image. This strategy reduces accumulative errors to a minimum and increases robustness to failure of visual odometry.

However, errors are still accumulated over time. To prevent the calculated roll and pitch angles from drifting, they can be fused with absolute roll and pitch angle measurements from the IMU. This has been successfully tested on a commercial wheeled robot but has not been implemented for the DLR Crawler so far. The use of IMU position information for overcoming erroneous visual odometry data in case of blurred images or bad lighting conditions is also subject to future work.

4.3 Terrain Modeling

A digital terrain model (DTM), which is incrementally built from the depth images, was chosen as internal map. The DTM represents the environment as a regular grid and each grid cell stores a single height value. Although this model cannot be used to represent multiple height values per grid cell, it is sufficient for many applications where overhangs are rare, such as planetary surface exploration. DTMs need only little storage space in comparison to full 3D models and path planning algorithms can be applied easily.

The first step of creating a DTM is the reconstruction of the 3D coordinates P^C of the image points in the camera coordinate frame based on the depth image. Since the error in the calculated distance of an object point from the camera grows quadratically with the distance, only image points with a disparity value greater than a threshold $p_{d_{thresh}}$ are considered for map creation. The threshold is chosen according to the map resolution so that a disparity error of one pixel causes a maximal distance error of the size Δl of a grid cell as

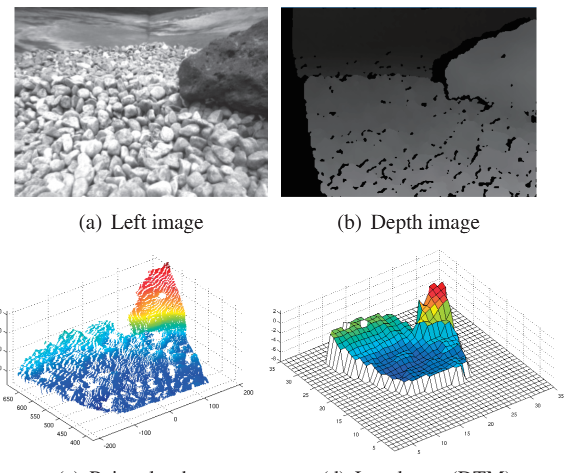
$$p_{d_{thresh}} \approx \sqrt{\frac{f \cdot b}{\Delta l}}, \tag{2}$$

where f is the focal length in pixels and b is the baseline width of the stereo camera pair. For the Crawler, the map resolution was set to $\Delta l = 20$ mm, resulting in a disparity threshold of $p_{d_{thresh}} = 31$ px which corresponds to a camera range of about 600 mm.

Knowing the rotation R_{WC} and the translation T_{WC} between the camera coordinate frame and the world coordinate frame, the 3D coordinates P^W of the image points in the world coordinate frame can be computed as

$$P^W = R_{WC} \cdot P^C + T_{WC}. \tag{3}$$

Of all points with equal P_x^W, P_y^W coordinates, only the points with the highest P_z^W values are stored. The resulting height value for each grid cell is computed as the mean of all P_z^W coordinates whose x, y coordinates are located in that grid cell. Using this method, a local map is created from each depth image (ref. Fig. 5).



(a) Left image (b) Depth image

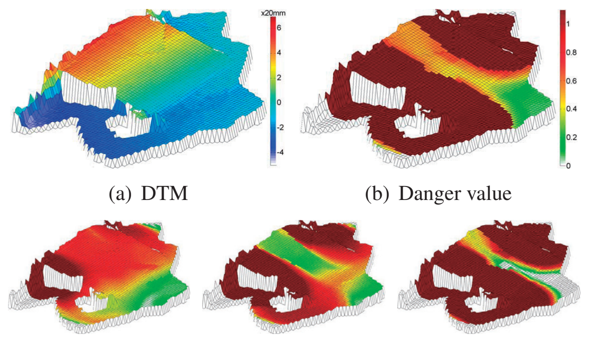
(c) Point cloud (d) Local map (DTM)

Figure 5: Local map creation from a stereo image

Each local map created from a depth image is attached to the global DTM using the estimated robot pose at the time of image acquisition. This approach is prone to errors from visual odometry, which can cause artifacts in the DTM. However, these errors remain small for small scale maps and can be considered in the traversability estimation process by taking the time into account when the height value was inserted into the global map [2].

4.4 Traversability Estimation

For path planning, the traversability of the global DTM must be estimated according to the robot's abilities. In rough terrain it is beneficial to not only classify the terrain in obstacles and free space but to assign continuous traversability values to each part of the terrain. During traversability estimation a danger value is assigned to each grid cell. The danger value of a cell is a measure of how difficult it is for the robot centered on that cell to traverse the terrain in any direction, which means, how much safety risks the robot is exposed to and how much energy is required for the movement. By doing so, the robot could be able to choose paths according to internal states, e.g. prefer a difficult shortcut over a safe and easy detour if time critical tasks have to be fulfilled.



(a) DTM (b) Danger value

(c) Slope (d) Roughness (e) Step height

Figure 6: Danger value computation from the criteria slope, roughness and step height

Completely flat and smooth terrain is assumed to have a danger value of $d = 0$ because it is traversable for a robot with minimal energy costs and minimal safety risk. A grid cell is untraversable if the robot is exposed to hazards such as collisions or tilt over. The navigation algorithm calculates the danger value of a grid cell from the three criteria slope s, roughness r and step height h. Slope and roughness of a cell are computed by fitting a plane to a circular terrain patch of the size of the maximal robot diameter (600 mm for the Crawler) around that cell. The slope is calculated as the angle between the z-axis of the world coordinate system and the normal vector of the plane. The roughness is computed as the standard deviation of the terrain points from the plane. The step height is computed from local height differences of terrain points within the maximal robot diameter. Cells are untraversable if one of the three criteria exceeds a corrsponding critical value s_{crit}, r_{crit} or h_{crit}, which are chosen according to the robot's abilities. For the Crawler, values of $s_{crit} = 20°$, $r_{crit} = 30$ mm and $h_{crit} = 70$ mm are used. Untraversable cells are assigned a danger value of $d = \infty$. If all three criteria are below the critical values, a danger value of the grid cell between 0 and 1 is computed as

$$d = \alpha_1 \frac{s}{s_{crit}} + \alpha_2 \frac{r}{r_{crit}} + \alpha_3 \frac{h}{h_{crit}}, \tag{4}$$

where α_1, α_2 and α_3 are weight parameters which sum up to 1. Fig. 6 shows a DTM with assigned danger values computed from the three criteria. A danger value for a grid cell is only computed if there is sufficient information about the surrounding terrain.

4.5 Path Planning

Based on the DTM and assigned danger values, an optimal path to the goal point can be planned. Since the robot's knowledge of the environment changes over time, the path planner must be able to replan paths efficiently. Thus, a D* Lite [13] path planner was chosen, which basically works like an A* graph based planner, but allows efficient replanning by locally modifying the results of the previous path planning step. As A*, D* Lite finds the minimum cost path to a goal vertex in a graph by first searching those vertexes which most likely lead to the goal point. For the path planner, the grid map is treated as a graph where the grid cells are the vertexes and the edges connect vertexes which refer to adjacent grid cells. The design of the cost function $c(N, N')$ of moving between two grid cells N and N' defines the optimality of the path. Since the path planner should take the terrain traversability into account, the cost function was defined as the weighted sum of distance and danger value:

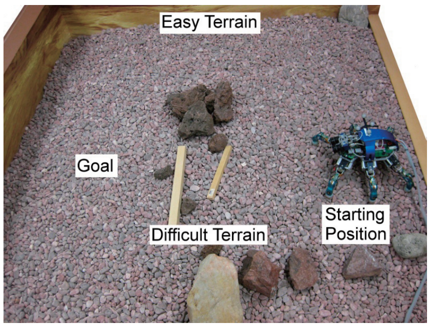
$$c(N, N') = \sqrt{(N_x - N'_x)^2 + (N_y - N'_y)^2} + \beta \cdot d(N'), \tag{5}$$

where $d(N')$ is the danger value of the target grid cell and β is the weighting factor. For small values of β, the path planner plans short and possibly more difficult paths, for larger values of β longer but safer paths are planned. The costs of moving to an untraversable cell are ∞.

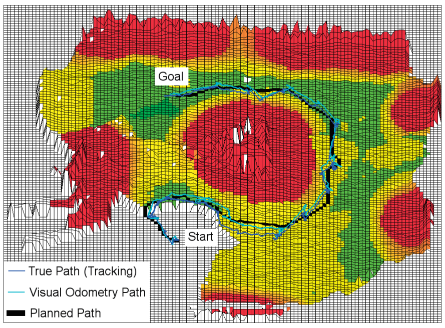
From the definition of the cost function follows approximately that paths are planned which are β times longer than the shortest path, if their average danger value is less than $\frac{1}{\beta}$ of the danger of the shortest path. That means, that only the relation between path length and path safety is considered but not the absolute danger value of a path. Therefore, the path planner additionally allows to define a danger value threshold $0 \leq d_{max} \leq 1$. If the danger value of a cell is higher than d_{max}, the cost of moving to that cell is set to ∞. Thus, the safety of the planned path can be improved without the need of changing parameters in the traversability estimation process. This is beneficial if the robot should avoid difficult areas for some reason, no matter how long the detour would be, e.g., if the robot has to carry heavy loads or if the robot's hardware is damaged. Currently, this threshold is set by the operator, but in future a high-level task planner should be in charge of adjusting this value.

4.6 Path Following

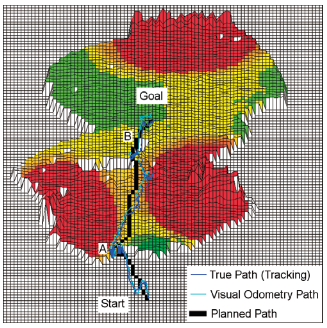
The planned path is followed by sending the simple motion commands "move forward", "turn left" and "turn



(a) Testbed setup



(b) Map and paths for $d_{max} = 0.2$



(c) Map and paths for $d_{max} = 1$

Figure 7: Test of the navigation algorithm with different danger value thresholds d_{max}

right" to the walking layer. Furthermore, the gait pattern can be chosen. Thus, in easy, smooth terrain the robot can use the simple and fast tripod gait, and in rough and more difficult terrain the gait can be switched to the computationally more expensive biologically inspired gait with elevation reflexes for overcoming higher obstacles.

5 Experimental Results

Different experimental setups were created to test the interaction between the walking layer and the navigation layer. An external tracking system was used to track a target body mounted on the Crawler, which provided ground truth measurements in comparison to estimated positions by visual odometry.

A first experiment should show the effects of different danger value thresholds d_{max} on the planned paths. The robot was given a goal point located at 1.20 m ahead of its starting position. The direct path to the goal point lead through difficult terrain. An easier but significantly longer path existed to the right of the Crawler. The testbed setup is depicted in Fig. 7(a). For the first run, the danger value threshold of the path planner was set to a low value of $d_{max} = 0.2$. Using this threshold, all paths leading through grid cells with a danger higher than 0.2 are marked as blocked by the path planner. Hence, the direct path to the goal was avoided and the Crawler chose the longer but safer path to the goal point. Figure 7(b) shows the resulting navigation map as well as the planned path, the true path and the path estimated by visual odometry. The error of the visual odometry estimate is 4.0 cm at the goal point.

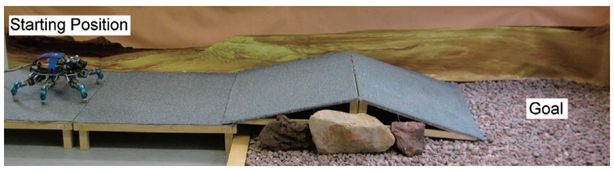
For the second run, the danger value threshold was set to 1, which means, that only paths are marked as blocked by the path planner which contain untraversable cells. Hence, the Crawler chose the direct but difficult path to the goal point. The resulting map and the paths are shown in Fig. 7(c). Since the robot had to traverse difficult terrain, the navigation layer switched the gait pattern. The easy areas were traversed via the simple tripod gait. At point *A* the gait pattern was switched to the biologically inspired gait with reflexes to overcome higher bumps. After traversing the difficult area, the tripod gait was chosen at point *B* to walk to the goal point. At the goal point, the error of the visual odometry was 5.5 cm compared to the true position.

A second test environment was built of several ramps of different slopes. The Crawler is not able to traverse the transitions between the different slopes using the tripod gait. Thus, the gait pattern had to be switched to the biologically inspired gait at the transitions. Fig. 8 shows the test environment and the navigation map. The weight parameters for calculating the danger value (ref. Eq. 4) were chosen so that terrain roughness had a higher influence on the danger value than terrain slope and step height. Thus, the difficulty of the transitions was estimated higher than the difficulties of the slopes. The elevation profile of the true path is shown in Fig. 8(c). The colors indicate the used gait pattern. As can be seen, the gait switched to the biologically inspired gait for crossing the transitions and to the tripod gait for traversing the slopes and the flat parts of the terrain.

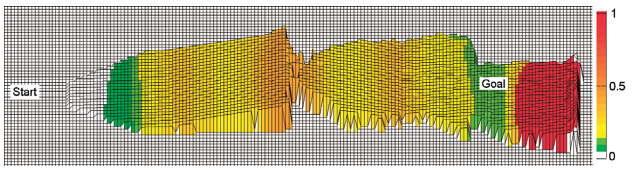
6 Conclusions and Future Work

This paper provides an overview of the work on the DLR Crawler towards the future goal of an autonomous walking robot for planetary exploration. For this purpose, the hardware setup, different gait coordination methods incorporating reflexes as well as a stereo vision based navigation algorithm have been presented. It has been shown in experiments that the Crawler is able to navigate in unknown terrain only using its on-board sensors and the capabilities inherent to its walking and navigation layer. Further, a first approach to influence the navigation towards taking a shorter, more difficult path on the one hand or a longer, safer path on the other hand depending on the robot's state or mission goal has been presented and demonstrated experimentally. Additionally, a preliminary adaptation of the gait based on the terrain assessment has been shown.

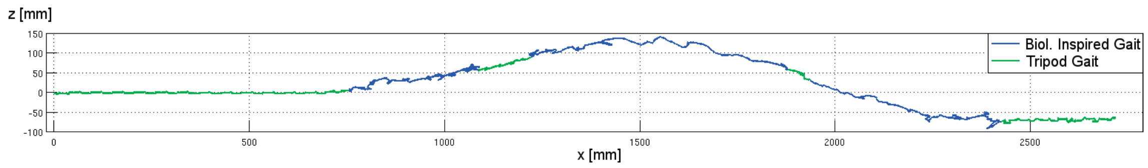
In order to be completely autonomous many different problems still have to be solved. First, a two-way data exchange between the walking and the navigation layer has to be implemented to provide both layers with environmental information gained by the large set of sensors. These information could be pre-filtered by the data collecting layer and contain additional information about the



(a) Test environment



(b) Navigation map



(c) True path and gait patterns

Figure 8: Test of the gait switching behavior

effort of a layer negotiating various situations. Thus, an adaptation of the interplay of the layers should lead towards more efficient solutions. Further, a planner has to be implemented that plans footholds and body poses to handle very difficult terrain. Additionally, some mission planning capability has to be provided. On the hardware side a more robust, highly integrated robot has to be developed that is power and computation autonomous. It should allow to try out uncertain paths and should be able to recover from falls, using this experience of difficulties to adapt parameters of its walking and navigation layer following the concept "learning by doing".

References

[1] R. Brooks, "A robust layered control system for a mobile robot", IEEE journal of robotics and automation, 2(1), (1986), pp. 14–23.

[2] A. Chilian and H. Hirschmüller, "Stereo Camera Based Navigation of Mobile Robots on Rough Terrain", in IROS, International Conference on Intelligent Robots and Systems, 2009, pp. 4571–4576.

[3] H. Cruse, "What Mechanisms Coordinate Leg Movement in Walking Arthropods", Trends in Neuroscience, 13, (1990), pp. 15–21.

[4] I. Ernst and H. Hirschmüller, "Mutual Information based Semi-Global Stereo Matching on the GPU", in International Symposium on Visual Computing (ISVC08), volume LNCS 5358, Part 1, Las Vegas, NV, USA, 2008, pp. 228–239.

[5] S. Gehrig, F. Eberli, and T. Meyer, "A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching", in International Conference on Computer Vision Systems (ICVS), volume LNCS 5815, Liege, Belgium, 2009, pp. 134–143.

[6] M. Görner and G. Hirzinger, "Analysis and Evaluation of the Stability of a Biologically Inspired, Leg Loss Tolerant Gait for Six- and Eight-Legged Walking Robots", in IEEE 2010 International Conference on Robotics and Automation, 2010, pp. 1525 – 1531.

[7] M. Görner, T. Wimböck, A. Baumann, M. Fuchs, T. Bahls, M. Grebenstein, C. Borst, J. Butterfass, and G. Hirzinger, "The DLR-Crawler: A Testbed for Actively Compliant Hexapod Walking Based on the Fingers of DLR-Hand II", in IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems, 2008, pp. 1525 – 1531.

[8] M. Görner, T. Wimböck, and G. Hirzinger, "The DLR Crawler: evaluation of gaits and control of an actively compliant six-legged walking robot", Industrial Robot: An International Journal, 36(4), (2009), pp. 344–351.

[9] H. Hirschmüller, "Stereo Processing by Semi-Global Matching and Mutual Information", IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2), (2008), pp. 328–341.

[10] H. Hirschmüller, P. R. Innocent, and J. M. Garibaldi, "Fast, Unconstrained Camera Motion Estimation from Stereo without Tracking and Robust Statistics", in Proceedings of the 7th International Conference on Control, Automation, Robotics and Vision, Singapore, 2002, pp. 1099–1104.

[11] H. Hirschmüller, P. R. Innocent, and J. M. Garibaldi, "Real-Time Correlation-Based Stereo Vision with Reduced Border Errors", International Journal of Computer Vision, 47(1/2/3), (2002), pp. 229–246.

[12] H. Hirschmüller and D. Scharstein, "Evaluation of Stereo Matching Costs on Images with Radiometric Differences", IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(9), (2009), pp. 1582–1599.

[13] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain", in Proceedings of the International Conference on Robotics and Automation, 2002, pp. 968–975.

[14] D. Wooden, M. Malchano, K. Blankespoor, A. Howard, A. Rizzi, and M. Raibert, "Autonomous Navigation for BigDog", in Proc. of the IEEE International Conference on Robotics and Automation, 2010.

[15] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondance", in Proceedings of the European Conference of Computer Vision, Stockholm, 1994, pp. 151–158.