Stereo Vision Based Navigation of a Six-Legged Walking Robot in Unknown Rough Terrain

Annett Stelzer, Heiko Hirschmüller, Martin Görner

Abstract

This paper presents a visual navigation algorithm for the six-legged walking robot DLR Crawler in rough terrain. The algorithm is based on stereo images from which depth images are computed using the Semi-Global Matching (SGM) method. Further, a visual odometry is calculated along with an error measure. Pose estimates are obtained by fusing inertial data with relative leg odometry and visual odometry measurements using an indirect information filter. The visual odometry error measure is used in the filtering process to put lower weights on erroneous visual odometry data, hence, improving the robustness of pose estimation. From the estimated poses and the depth images, a dense digital terrain map is created by applying the locus method. The traversability of the terrain is estimated by a plane fitting approach and paths are planned using a D* Lite planner taking the traversability of the terrain and the current motion capabilities of the robot into account. Motion commands and the traversability measures of the upcoming terrain are sent to the walking layer of the robot so that it can choose an appropriate gait for the terrain. Experimental results show the accuracy of the navigation algorithm and its robustness against visual disturbances.

1 Introduction

Research on autonomous robots has gained more and more importance over the past years. Following the recent natural disasters and nuclear threats, the public desire for autonomous robots to support humans has grown significantly. In applications such as search and rescue and planetary exploration mobile robots would be valuable to prevent humans from being exposed to danger. In such applications, robots have to deal with rugged terrain, steep slopes and changing substrates. Under these conditions, walking robots are expected to show superior performance to wheel driven ones. Their advantages are, no need for paths of continuous contact with the ground, 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 heterogeneous team of robots seems to be a promising solution. Such a team could consist of large wheeled robots for supply and long distance transport, aerial robots for overviewing the terrain and a team of highly mobile multi-legged robots for local search and exploration tasks. Before tackling the problem of multi-robot cooperation, first the important challenges of robust locomotion and robust autonomous navigation need to be solved.

We believe, that each robot in a team should be able to fulfill its task independent of the other team members. This implies that each robot carries all necessary sensors to perceive its own state as well as its immediate environment. If available, it should use information provided by other robots, such as map parts or localization data, but it should not be reliant on it. Furthermore, it should be independent of a priori maps and external infrastructure such as GPS, but it should be able to benefit from such information when accessible. The independence of other team members is an important feature since the failure of a single robot must not compromise successful task completion.

In this paper we will focus on multi-legged robots as rough terrain specialists in a heterogeneous robot team. In our opinion a stereo camera is the appropriate sensor for perceiving the environment. It is light weight, versatile as well as passive and can be used for motion estimation and creating geometrical models of the environment. Further, its data could in future enable cognitive processes relevant to the task. Additionally, for the walking task itself, a legged robot already embeds many proprioceptive sensors, such as joint angle sensors and joint torque sensors. Besides delivering information about the internal state of the robot, they are also a valuable source of information about the environment. Thus, including foot force sensors and an inertial measurement unit (IMU), the perception of a walking robot is mainly based on a combination of visual, inertial and tactile information. Some of these sensors give redundant information which can be combined to improve the data quality and which can also enhance robustness against sensor failure.

We believe that a legged exploration robot should use a layered control architecture [4] consisting of a walking layer, a navigation layer and a high-level task planner. The walking layer is responsible for the basic tasks of stable standing and walking, but also provides reflexes for non-flat terrain to overcome obstacles within the walking height reactively. For this purpose, the walking layer mainly uses the proprioceptive sensors. The task of the navigation layer is to guide the robot along an optimal path to a goal point using visual information of the environment. In order to gain the largest benefit from visual and tactile data, both layers should be strongly coupled. On the one hand, the navigation layer should provide high-level motion commands and information that helps the walking layer to anticipate upcoming terrain. On the other hand, the walking layer should send current motion information and tactile clues about the ground to the navigation layer. Further, it should provide information about the current motion capability of the robot. For example, if the robot is heavily loaded or damaged, the navigation layer needs to react by adapting the path. On the top level, a task planner needs to generate tasks in order to fulfill the mission goal and to handle exceptions. For example, if the navigation layer reports that the desired goal point cannot be reached, the task planner has to choose a different goal point.

The objective of this paper is to present a visual navigation algorithm for the six-legged walking robot DLR Crawler (ref. Fig. 1). This robot is a prototypic study for future exploration robots. It serves as a test platform to gain experience in walking and navigation as a step towards an autonomous walking robot in a heterogeneous team of robots. The DLR Crawler implements a walking layer using different gait algorithms and reflexes which enable it to overcome obstacles within its walking height reactively. The navigation algorithm generates motion commands for the walking layer to lead the robot along

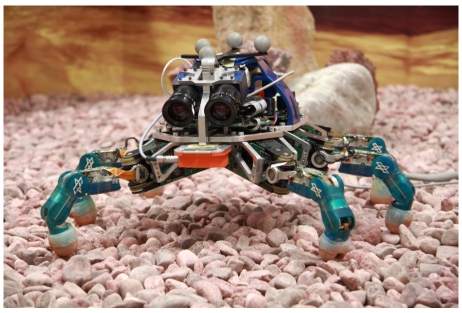


Figure 1: The DLR Crawler within the gravel testbed

a short and safe path to a goal point. It is designed for unstructured terrain and does not depend on external infrastructure such as GPS, nor a priori maps of the environment are required. Only measurements from an on-board IMU, leg odometry and the stereo camera are used to autonomously find and follow a path to given goal point coordinates. The main points this paper discusses are:

- Stereo visual odometry algorithm with error estimation
- Multisensor data fusion of inertial, leg odometry and visual odometry measurements for robust pose estimation using an indirect information filter
- Dense elevation mapping from stereo disparity images computed by Semi-Global Matching (SGM)
- Estimation of the terrain traversability according to the motion abilities of the robot
- Path planning considering the estimated terrain traversability and the current state of the robot
- Experimental results showing the accuracy of the navigation algorithm as well as its robustness against visual disturbances

The paper is organized as follows: Section 2 will provide a brief overview on related work in visual navigation. Section 3 introduces the DLR Crawler, describing the hardware as well as the implemented walking algorithms and reflexes. In Section 4 an overview of the visual navigation system is given. Section 5 describes the stereo image processing and the visual odometry calculation. The sensor data fusion algorithm for pose estimation is presented in Section 6. Section 7 deals with mapping rough terrain and Section 8 describes the traversability estimation. Path planning and motion control are presented in Sections 9 and 10. In Section 11 the on-line implementation of the navigation algorithm is explained briefly. Experimental results of the visual navigation system are shown in Section 12. Finally, Section 13 concludes this paper.

2 Related Work

In literature, several papers address autonomous rough terrain navigation. One of the first systems which was able to autonomously navigate on rough terrain was the Autonomous Land Vehicle (ALV) developed at the Hughes Artificial Intelligence Center in 1987 [8]. It used a laser scanner to build a navigation map and marked areas which would cause invalid vehicle configurations as untraversable.

NASA's Jet Propulsion Laboratory developed the Rocky rovers, which were test platforms for mostly sensor based navigation algorithms [30, 27]. They were prototypes for the Sojourner Mars rover, which arrived on Mars in 1997. Using a camera and a laser striper, it was able to detect and avoid obstacles along its way [31]. However, these systems only used a binary representation of the environment consisting of obstacles and free space.

The RANGER navigation system [23] is able to estimate the difficulty of traversable areas by computing the configuration a vehicle would have on certain terrain points. This knowledge is then included into the path planning process. Based on RANGER the Morphin algorithm [35] was developed. Using stereo images, it estimates the traversability of regular sized terrain patches by fitting a plane to the data. Morphin uses the traversability measures of these patches to evaluate the safety of possible steering arcs for the rover. These arcs are further evaluated on their use for reaching the specified goal point. The arc with the highest total vote is then commanded to the rover.

Morphin provided a basis for the development of the navigation software of the Mars exploration rovers Spirit and Opportunity. Visual odometry, wheel encoder readings and IMU data are combined for position estimation [28]. The automatic navigation mode uses the local path planner GESTALT [12]. By fitting a plane to a local terrain patch, the traversability of that grid cell is estimated and steering angles are commanded which lead the rover along a safe path in the direction towards the goal point. Since GESTALT is a local path planner an additional global Field D* planner was implemented [5].

Konolige et al. presented a stereo vision based outdoor navigation system using IMU, GPS and visual odometry [25]. Visual odometry, wheel encoder readings, IMU data and GPS are used for position estimation. A ground plane is extracted from the stereo depth images and obstacles are detected by thresholding the height above the ground plane. A modified gradient planner is used to plan global paths.

The four-legged robot BigDog implements a gait control system and an autonomy and perception system [38]. It uses proprioceptive force and position sensors, as well as GPS, an IMU, a stereo camera and a LIDAR scanner for navigation. Visual odometry, leg odometry and IMU data are used for pose estimation. Obstacles are detected using LIDAR and stereo vision and a 2D cost map is created. There is no traversability estimation of the terrain but cell costs are computed from obstacles and distances to obstacles. Using the cost map, paths are planned using an A* planner.

3 The DLR Crawler

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 [14] and [15].

3.1 Hardware

The DLR Crawler has a total mass of 3.5 kg and its feet span an area of 350 mm $\times$ 380 mm. Each of the six legs has four joints and three active degrees of freedom. 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 degrees of freedom (DOF) force-torque sensor and the body implements an IMU. For the purpose of visual odometry and vision based navigation a stereo camera head is mounted. Since the robot is a laboratory testbed, all control computation is done externally on a QNX realtime PC while the navigation algorithm employs an external Linux computer. This allows to quickly test different algorithms with varying computational complexity without caring about optimized implementation on specific on-board hardware at this stage. Further, the robot has an external 24 V power supply and on-board power distribution.

3.2 Walking Algorithms and Reflexes

Two different gait algorithms are implemented on the Crawler. They differ not only 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 is based on a biologically inspired variable coordination and multiple reflexes. This gait is more complex but allows to handle more challenging terrain and is even able to handle leg loss.

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 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.

The second 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.

The combination of these reflexes allows the Crawler to autonomously negotiate most obstacles within its walking height and to adapt to rough and uneven terrain. The elevator reflex requires a flexible gait coordination since its execution by a stepping leg causes extended power stroke phases in the supporting legs. 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 a good option for easy terrain due to its achievable speed and low computational complexity.

4 Navigation Algorithm Overview

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. An overview of the navigation algorithm for the DLR Crawler is shown in Fig. 2. It uses the IMU, leg odometry and the stereo camera running visual odometry for accurate and robust pose estimation. The estimated pose and the depth images computed from the stereo data are used to build a 2.5D terrain map. The traversability of the terrain is estimated from the map and a safe and short path is planned to the goal point taking the motion capabilities of the robot into account. Motion commands for following the path and the estimated difficulty of the terrain are sent to the walking layer. The single blocks will be described in detail in the following sections.

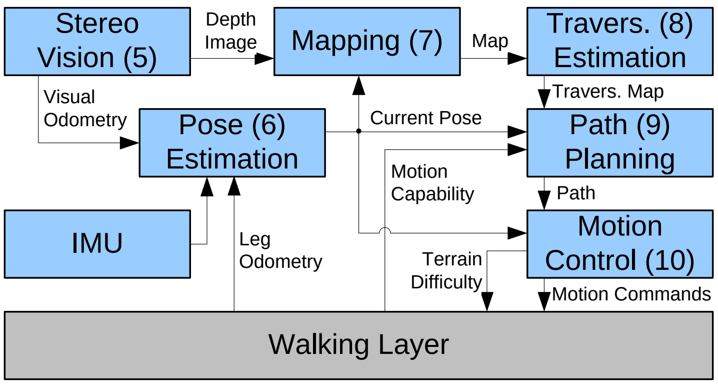


Figure 2: Overview of the navigation algorithm. Section numbers are given in brackets.

5 Stereo Vision

The stereo camera head consists of two AVT Guppy F080B grayscale firewire cameras. They contain a CCD chip with a resolution of 1032×778 pixel. The cameras are equipped with Theia wide angle lenses with a focal length of 1.3 mm. That results in a horizontal opening angle of about 123°. Sample images can be found in section 12. The stereo baseline is about 50 mm.

The cameras are calibrated at full resolution and after that the images are downscaled to 516×389 pixel which is sufficient for this application and saves computation time. The images are rectified by projecting both images onto a common plane that has the same distance to both optical centers [18]. Rectification enforces that the projection of any feature appears in the same image row in both images. This saves processing time of the following steps.

5.1 Stereo Matching

Dense stereo matching is performed for computing depth images of the scenery in front of the robot. A correlation based approach [20] is normally sufficient for obstacle avoidance and navigation. However, it has been found that using Semi-Global Matching (SGM) is more advantageous since it delivers denser results with far fewer outliers [19]. Furthermore, SGM can also reconstruct thin or small objects that are often undetected by correlation methods. For these reasons, SGM has also been used in a real-time FPGA implementation for driver assistance tasks [11].

SGM is based on the idea of pixel-wise matching, supported by a global smoothness constraint. The resulting global cost function is minimized along 8 path directions that originate at the image border. Thus, 8 paths meet at every pixel. They are combined by summing their costs and the disparity that minimizes the cost is chosen for each pixel separately. Occlusions and mismatches are identified and invalidated by a left/right consistency check that inverses the roles of both cameras and removes all disparities that differ. Census [39] has been chosen as matching cost, due to its robustness against many radiometric changes [22].

For real-time performance, an implementation on the GPU has been used. The original implementation [10] has been extended for supporting Census as matching cost. The GPU implementation runs with $4 - 5\,\mathrm{Hz}$ on VGA sized images (640×480 pixel) with 128 pixel disparity range on a GeForce GTX 275. The runtime depends linearly on the number of pixels and disparities. In this application, on images with 516×389 pixel with 128 pixel disparity range, a rate of about $6\,\mathrm{Hz}$ is achieved. A customized FPGA implementation which enables on-board processing for the DLR Crawler is anticipated in future.

5.2 Visual Odometry

Visual odometry is the determination of the camera movement with respect to the environment. Stereo camera methods permit computing all six degrees of freedom (i.e. translation and rotation) in contrast to mono camera methods that can only determine the direction, but not the scale of motion. Visual odometry is independent of wheel or leg slip, but it assumes a (mostly) static environment.

Since the anticipated frame rate is rather low, large motions can occur between consecutive images, especially if the robot rotates, i.e. turns left or right. Therefore, a method has been chosen that does not rely on feature tracking [21, 18]. Fig. 3 shows an overview of the method.

Feature points in consecutive left camera images are selected by the Harris corner detector [17]. A square patch around each corner is used as feature descriptor. The descriptor is extracted from Rank transformed [39] left images for making the comparison of descriptors with the sum of absolute differences

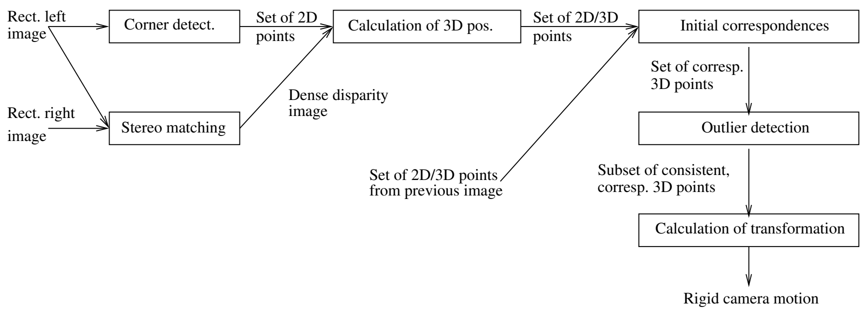


Figure 3: Overview of the used stereo visual odometry method

(SAD) robust against radiometric changes. The Rank transformation is slightly inferior to Census, but can be computed faster. It has been found that using such a non-parametric transformation is also robust against small rotations and perspective changes, since they are tolerant against outliers, in contrast to classical correlation methods like normalized cross correlation. It should be noted that using scale and rotation invariant features such as SIFT is not beneficial, since rotation around the optical axis will be minimal as well as scale differences. In contrast, using rotation and scale invariant descriptors can lead to worse results due to less discriminative power.

All corners of one image are compared against the corners of the previous image for finding initial correspondences. Thereafter, the corners of the previous image are compared to the corners of the current image and only those correspondences that agree in both directions are retained. All of the corresponding feature points are reconstructed in 3D in their own camera coordinate system using the depth image from stereo matching. It is assumed that there are many wrong correspondences.

For finding these correspondence outliers, the static scene assumption is used. If the scene is static, then the 3D distance of two points in the previous camera coordinate system must be the same as the distance of the corresponding points in the current camera coordinate system. For comparing distances, it is very important to take stereo reconstruction errors into account. Let x_1, y_1, z_1 and x_2, y_2, z_2 be two reconstructed feature points in the same camera coordinate system. The distance between both is obviously

$$L = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}. \tag{1}$$

An assumed uncertainty of $\varepsilon_p = 0.2$ pixel in the image plane propagates into an uncertainty in the length by

$$\varepsilon_L = \frac{\varepsilon_p}{Lft}\sqrt{z_1^2\,(A + B + C) + z_2^2\,(D + E + F)}, \tag{2}$$

$$A = ((x_1 - x_2)\,(t - x_1) - (y_1 - y_2)\,y_1 - (z_1 - z_2)\,z_1)^2\,, \tag{3}$$

$$B = ((x_1 - x_2)\,x_1 + (y_1 - y_2)\,y_1 + (z_1 - z_2)\,z_1)^2\,, \tag{4}$$

$$C = \frac{1}{2}\,(t\,(y_1 - y_2))^2\,, \tag{5}$$

$$D = ((x_1 - x_2)\,(t - x_2) - (y_1 - y_2)\,y_2 - (z_1 - z_2)\,z_2)^2\,, \tag{6}$$

$$E = \left((x_1 - x_2)\, x_2 + (y_1 - y_2)\, y_2 + (z_1 - z_2)\, z_2 \right)^2 , \tag{7}$$

$$F = \frac{1}{2} \left(t\, (y_1 - y_2) \right)^2 , \tag{8}$$

with f as focal length in pixel and t as baseline, i.e. distance between both cameras. The length L_1 can be considered equal to L_2, if

$$|L_1 - L_2| \leq 3\sqrt{\varepsilon_{L_1}^2 + \varepsilon_{L_2}^2}. \tag{9}$$

If this constraint is violated, then at least one of the correspondences *must* be wrong. Otherwise, one or both *may* be correct. It appears that finding the largest subset of corresponding points that satisfies the constraint for all pairs, is an NP problem. Therefore, not the largest, but just a large consistent subset is determined. All combinations of corresponding pairs are compared and stored. The consistent subset is constructed by starting with the point that is consistent to the most other points. Further points are incrementally added, if they are consistent with all points that are already in the consistent subset, by preferring the points with the highest number of further consistencies. This process ends if there are no more points that are consistent with all other points of the set. This method is similar to a random sampling approach (RANSAC), as it finds a large outlier free subset by assembling it bottom-up from a small consistent subset. However, in contrast to RANSAC, the method is deterministic as no form of randomization is used. It should be noted that there are other methods for finding a large outlier free set of correspondences, that could also be used [2, 32]. However, the method described above has proven to be well suited for this application because it is robust and has low computational costs.

The rigid motion is determined as the rotation and translation between corresponding reconstructed points. It is computed in closed form by singular value decomposition as shown by Haralick et al. [16]. A treatment for the special case with planar point sets is shown by Arun et al. [1]. With this initial result, Chauvenet's criterion is used to identify and eliminate all correspondences, with unexpected high errors.

The rigid motion is parametrized as $\boldsymbol{x} = \begin{pmatrix} t_x & t_y & t_z & n_1 & n_2 & n_3 \end{pmatrix}^T$, with $\boldsymbol{t}$ as translation vector, $\boldsymbol{n}$ as rotation axis and $\alpha = |\boldsymbol{n}|$ as rotation angle, with $0 \leq \alpha \leq \pi$. With the initial motion, corresponding points can be reconstructed and the reprojection error vector $\boldsymbol{y} = \begin{pmatrix} p'_{1x} - p_{1x} & \dots & p'_{ny} - p_{ny} \end{pmatrix}^T$ computed as the difference between feature point locations $\boldsymbol{p}_i$ in the image and the corresponding projected locations $\boldsymbol{p}'_i$ of their reconstructions. Alternatively, the error vector can be computed using the ellipsoid error model [29], which is a very good approximation, but faster to compute and therefore preferred. The function to be minimized (e.g. by Levenberg-Marquardt) is $\boldsymbol{y} = f(\boldsymbol{x})$ with $\boldsymbol{y}_0 = f(\boldsymbol{x}_0)$ as solution with the lowest reprojection error.

5.3 Visual Odometry Error Estimation

The original publication of the visual odometry method [21, 18] has been extended by estimating the motion error as well. This error depends not only on the number, but also on the distribution of feature points in the image. About $\varepsilon_p = 0.5$ pixel is a typical error in feature point localization (i.e. in the error vector $\boldsymbol{y}$), assuming that the correspondences are outlier free. The propagation of this error into the parameters $\boldsymbol{x}$ results in the parameter error ε_x. For computing this error, the function $f(\boldsymbol{x})$ must be inverted. Since it is not invertible in closed form (otherwise a non-linear optimization would not be needed), a linearization is computed at $\boldsymbol{x}_0$ as approximation,

$$\boldsymbol{y} = f(\boldsymbol{x}) \approx \boldsymbol{J}_0 (\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{y}_0, \tag{10}$$

with $\boldsymbol{J}_0$ as Jacobian matrix at the solution $\boldsymbol{x}_0$. The Levenberg-Marquardt optimization computes the Jacobian matrix internally, which may be reused, or it can be computed from scratch by numerical forward differentiation of $f(\boldsymbol{x}_0)$. For small values $\boldsymbol{x} - \boldsymbol{x}_0$, the linearization is a good approximation of the original function. This approximation can be inverted by $\boldsymbol{x} - \boldsymbol{x}_0 = \boldsymbol{J}_0^+ (\boldsymbol{y} - \boldsymbol{y}_0)$, with $\boldsymbol{J}_0^+$ as pseudo inverse of the Jacobian, computed by singular value decomposition. In this formulation, the error ε_p can be propagated individually, corresponding to each element of $\boldsymbol{y}$, by $\boldsymbol{\varepsilon}_x^i = \boldsymbol{J}_0^+ \boldsymbol{\varepsilon}^i$, with $\boldsymbol{\varepsilon}^i$ as null vector with only element i set to ε_p. If independent errors are assumed, then the individually propagated errors are simply the square root of the sum of squares according to the rules of error propagation. This is effectively the same as multiplying the pixel error with the L_2 norm over the rows of the inverse Jacobian, i.e.

$$\varepsilon_{xk} = \varepsilon_p \left| J_{0k}^+ \right|, \tag{11}$$

with ε_{xk} as k-th element of the error vector $\boldsymbol{\varepsilon}_x$ and J_{0k}^+ as the k-th row of $\boldsymbol{J}_0^+$. It is important to understand that the estimation of the visual odometry error $\boldsymbol{\varepsilon}_x$ implicitly includes all sources of errors due to bad conditioned scenes with weak texture or low contrast, like low number of correspondences, feature points that are clustered in an image area, etc. Therefore, it is a very good value for judging the quality of visual odometry for fusion with other ego-motion sensors.

To get an estimate of the absolute motion error, all relative error estimates $\boldsymbol{\varepsilon}_x$ have to be propagated. The motion $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{t} & \boldsymbol{n} \end{pmatrix}$ from one image to the next can be written as rotation matrix $\boldsymbol{R}(\boldsymbol{n})$, that is computed from the angle-axis notation $\boldsymbol{n}$, and a translation vector $\boldsymbol{t}$, such that a point $\boldsymbol{P}_{j+1}$ in the $(j+1)$-th camera coordinate system is transformed into the previous camera coordinate system by $\boldsymbol{P}_j = \boldsymbol{R}(\boldsymbol{n}) \boldsymbol{P}_{j+1} + \boldsymbol{t}$. The absolute motion (i.e. relative to the first camera coordinate system) is then computed as

$$\begin{aligned} \boldsymbol{n}_{j+1} &= \boldsymbol{R}^{-1} \left(\boldsymbol{R}(\boldsymbol{n}_j) \boldsymbol{R}(\boldsymbol{n}) \right), \\ \boldsymbol{t}_{j+1} &= \boldsymbol{R}(\boldsymbol{n}_j) \boldsymbol{t} + \boldsymbol{t}_j, \end{aligned} \tag{12}$$

with $\boldsymbol{R}^{-1}$ as the function that computes the angle-axis parameters from the given rotation matrix. To obtain the absolute motion error $\boldsymbol{\varepsilon}_{x_{j+1}}$ corresponding to the (j+1)-th camera position, the absolute motion error $\boldsymbol{\varepsilon}_{x_j}$ is combined with the relative motion error $\boldsymbol{\varepsilon}_x$ by adding each of the six elements of both error vectors individually to the corresponding elements of $\boldsymbol{x}_j$ and $\boldsymbol{x}$. Using (12), this results in 12 (erroneous) estimates $\boldsymbol{x}'_{j+1}$ of the absolute motion from which the unbiased absolute motion $\boldsymbol{x}_{j+1}$ is subtracted. The parameter difference between rotations $\boldsymbol{n}$ and $\boldsymbol{n}'$ in angle-axis form is computed as

$$\Delta \boldsymbol{n} = \begin{cases} \boldsymbol{n} - \boldsymbol{n}' & \boldsymbol{n}^T \boldsymbol{n}' \geq 0, \\ \min \begin{pmatrix} \boldsymbol{n} - \boldsymbol{n}', & \boldsymbol{n} \frac{|\boldsymbol{n}| - 2\pi}{|\boldsymbol{n}|} - \boldsymbol{n}' \end{pmatrix} & \text{otherwise}, \end{cases} \tag{13}$$

for considering the discontinuity of the angle-axis notation. It is important to note that the parameter difference of the rotation is *not* equal to the difference rotation. It is only used for specifying the error in the corresponding parameters. According to the rules of error propagation, the 12 individual differences/error vectors are combined by *component wise* computing the square root of the sum of the 12 squared elements. This leads to the error vector $\boldsymbol{\varepsilon}_{x_{j+1}}$, that corresponds to $\boldsymbol{x}_{j+1}$. In the whole discussion, covariances were ignored by always assuming independent errors for reasons of simplicity.

The absolute motion error estimate is used for another extension of the original work, which optionally computes the rigid motion not only to the previous image, but independently to all images of a set of previous images. The motion estimate with the minimum absolute motion error estimate is taken as result. For saving computation time, the set contains only a limited, small number of previous images. After computing the motion, the current image replaces either an image to which the motion could not be calculated (i.e. which is too old) or the one with the highest overall motion error estimate. With this strategy, solutions with lower overall motion error estimates are preferred and motion drift is minimized. The computation time is minimized by storing the 3D locations and rank signatures of the images as intermediate results, instead of the images themselves. Thus, the most expensive steps only need to be done once.

We found that this strategy makes visual odometry very robust and reduces drift, especially in situations with slow motion in comparison to the frame rate, e.g. the drift will be zero, if the system does not move. However, we do not utilize this option in the current work, because it would make motion fusion much more complicated as the visual odometry is computed to different images in the past. Instead, we only used incremental visual odometry, which calculates motion always to the previous image.

6 Pose Estimation

In this section, the multisensor data fusion algorithm as already presented in [7] is explained which uses an indirect information filter for fusing inertial measurements of the IMU with relative translation and rotation measurements from the 3D visual odometry and 3D leg odometry of the Crawler. Sensor data fusion is used to achieve more accurate and robust pose estimates than obtained by using motion measurements of a single sensor.

6.1 Motion Sensors

In addition to the stereo camera running visual odometry as described in the previous section, the DLR Crawler uses an XSens IMU and leg odometry as motion sensors.

The XSens IMU is based on MEMS inertial sensors and consists of three accelerometers and three gyroscopes measuring the accelerations in x, y, z direction and the angular velocities around the three axis, respectively. 3D magnetometer data is also available but in the current work only calibrated accelerometer and gyroscope measurements are used at a rate of 120 Hz. The full scale accelerations and rates of turn are $50\,\mathrm{m/s^2}$ and 1200 deg/s, respectively. Accelerometer noise is $0.008\,\mathrm{m/s^2}$ and gyroscope noise is 0.006 rad/s. Since the accelerometers also sense the gravity, absolute roll and pitch angles can be derived. By integrating the accelerations and angular velocities, the velocity, position and orientation of the IMU can be computed. However, since the accelerometer and gyroscope measurements are biased, the errors in position and orientation will grow unbounded due to the integration. For this reason, the IMU needs to be corrected by other sensors with less drift to give good position estimates. The advantage of the IMU is that it only depends on the present gravity and apart from that is independent of environmental conditions.

Using the leg joint angle and joint torque measurements, a 6 DOF odometry of the DLR Crawler is computed. It estimates relative pose changes of the robot based on matching point clouds, which are represented by the positions of the supporting feet. The algorithm assumes rigidity of the configurations, which implies a no slip condition for the feet. Hence, the quality of the relative leg odometry measurements depends on the ground conditions. Since the basic odometry is subject to strong drift of the pitch and roll angles, the joint torque sensors are used to compute an estimate of the earth gravity direction which allows to stabilize the absolute roll and pitch angles using an error state Kalman filter.

For optimally combining the measurements of all available motion sensors, a multisensor data fusion filter was developed. Since the IMU sends data at the highest rate and is independent of environmental conditions, it was chosen to be the main sensor for pose estimation. The IMU is aided by relative visual odometry and relative leg odometry measurements. Leg odometry and visual odometry can be considered as complementary because usually rough terrain, where leg odometry is prone to slip, has good texture and allows accurate visual odometry measurements, and vice versa.

6.2 Filter Choice For Multisensor Data Fusion

For multisensor data fusion, usually, probabilistic estimators such as the Kalman filter or its inverse formulation, the information filter, are used. In this application, an indirect feedback information filter is used. The information filter has the advantage that fusing measurements of multiple sensors at the same time can be achieved very easily. The indirect or error state form works on an error state vector which contains the errors of the actual state rather than the state variables themselves. The advantage is that no model of the usually nonlinear robot dynamics is required but the filter is based on linear equations describing the error propagation in the inertial system. The feedback formulation means that the estimated error is fed back into the IMU navigation equations to correct the current position, velocity and orientation estimates. For this, the estimated error states are kept small and small angle approximations in the filter equations are possible. That also means that the error state can be predicted as zero for each new filter step. Furthermore, the indirect filter formulation allows the filter to be run at a lower frequency than the inertial navigation equations. For a more detailed discussion of the different filter formulations the reader is referred to Roumeliotis et al. [34].

The information filter is numerically equivalent to the Kalman filter but has inverse complexity properties. In particular, while the prediction step of the Kalman filter is computationally simple and the update step is complex, the information filter equations yield a complex prediction step and a computationally cheap update step. For transforming the indirect Kalman filter into the information form, the information matrix $\boldsymbol{Y}$ and the error information vector $\Delta\boldsymbol{y}$ are defined as

$$\boldsymbol{Y} = \boldsymbol{P}^{-1} \qquad \text{and} \qquad \Delta\boldsymbol{y} = \boldsymbol{Y} \cdot \Delta\boldsymbol{x}, \tag{14}$$

where $\boldsymbol{P}$ is the estimation covariance matrix and $\Delta\boldsymbol{x}$ is the error state vector. Transforming the Kalman filter equations such that $\boldsymbol{Y}$ and $\Delta\boldsymbol{y}$ are estimated results in the prediction step

$$\boldsymbol{Y}_t^- = (\boldsymbol{A}_t\boldsymbol{Y}_{t-1}^{-1}\boldsymbol{A}_t^T + \boldsymbol{Q}_t^p)^{-1} \tag{15}$$

$$\Delta\boldsymbol{y}_t^- = \boldsymbol{Y}_t^-(\boldsymbol{A}_t\boldsymbol{Y}_{t-1}^{-1}\Delta\boldsymbol{y}_{t-1}), \tag{16}$$

where $\boldsymbol{A}_t$ is the state transition matrix and $\boldsymbol{Q}_t^p$ is the process noise matrix. In the feedback form, the prediction (16) can be simplified to $\Delta\boldsymbol{y}_t^- = 0$ because it is assumed that the error is corrected after each filter step. The update step of the information filter becomes

$$\boldsymbol{Y}_t = \boldsymbol{H}_t^T(\boldsymbol{Q}_t^m)^{-1}\boldsymbol{H}_t + \boldsymbol{Y}_t^- \tag{17}$$

$$\Delta\boldsymbol{y}_t = \boldsymbol{H}_t^T(\boldsymbol{Q}_t^m)^{-1}\boldsymbol{z}_t + \Delta\boldsymbol{y}_t^-, \tag{18}$$

where $\boldsymbol{H}_t$ is the measurement matrix and $\boldsymbol{Q}_t^m$ is the measurement noise matrix. In the indirect formulation, the measurement vector $\boldsymbol{z}_t$ is the difference between the IMU measurements and the measurements of an aiding sensor. The update step can be written as

$$\boldsymbol{Y}_t = \boldsymbol{I}_t + \boldsymbol{Y}_t^-, \qquad \text{with} \quad \boldsymbol{I}_t = \boldsymbol{H}_t^T(\boldsymbol{Q}_t^m)^{-1}\boldsymbol{H}_t, \tag{19}$$

$$\Delta\boldsymbol{y}_t = \boldsymbol{i}_t + \Delta\boldsymbol{y}_t^-, \qquad \text{with} \quad \boldsymbol{i}_t = \boldsymbol{H}_t^T(\boldsymbol{Q}_t^m)^{-1}\boldsymbol{z}_t. \tag{20}$$

The term $\boldsymbol{I}_t$ is the amount of information in the measurement and $\boldsymbol{i}_t$ is the contribution of the measurement $\boldsymbol{z}_t$ to the state vector [9]. If there are several measurements $\boldsymbol{z}_{k,t}$ at a timestep t we get

$$\boldsymbol{I}_t = \sum_{k=1}^{n} \boldsymbol{H}_{k,t}^T(\boldsymbol{Q}_{k,t}^m)^{-1}\boldsymbol{H}_{k,t} = \sum_{k=1}^{n} \boldsymbol{I}_{k,t} \tag{21}$$

$$\boldsymbol{i}_t = \sum_{k=1}^{n} \boldsymbol{H}_{k,t}^T(\boldsymbol{Q}_{k,t}^m)^{-1}\boldsymbol{z}_{k,t} = \sum_{k=1}^{n} \boldsymbol{i}_{k,t}. \tag{22}$$

The simplicity of the update stage of the information filter originates from the fact, that the measurements of the single sensors are conditionally independent. Hence, the information form of the Kalman filter has computational advantages for multisensor data fusion. The routines for computing $\boldsymbol{I}_{k,t}$ and $\boldsymbol{i}_{k,t}$ for each measurement are independent of each other and independent of $\boldsymbol{Y}_t^-$ and $\Delta\boldsymbol{y}_t^-$ and can run in parallel and on distributed systems. The disadvantage is, that a matrix inversion is required to obtain the error state vector $\Delta\boldsymbol{x}_t$ from the information vector $\Delta\boldsymbol{y}_t$. However, the more external sensors are used, the higher the benefit of using the information filter.

6.3 State Vector and State Transition Model

For implementing the information filter we chose to use a state vector consisting of 15 variables: The position $\boldsymbol{p}$ (3), the velocity $\boldsymbol{v}$ (3), the orientation Euler angles $\boldsymbol{\varphi}$ (3), the bias of the gyroscopes $\boldsymbol{b}_g$ (3) and the bias of the accelerometers $\boldsymbol{b}_a$ (3). In the indirect formulation the error state vector

$$\Delta\boldsymbol{x} = (\Delta\boldsymbol{p}, \Delta\boldsymbol{v}, \Delta\boldsymbol{\varphi}, \Delta\boldsymbol{b}_g, \Delta\boldsymbol{b}_a)^T \tag{23}$$

is used. The position $\boldsymbol{p}$ and velocity $\boldsymbol{v}$ variables are given in world coordinates with the origin located at the IMU origin at the beginning of the data fusion process. The Euler angles $\boldsymbol{\varphi}$ are the angles of the rotation matrix that turns a point from the IMU coordinate system to the world coordinate system. The bias values $\boldsymbol{b}_g$ and $\boldsymbol{b}_a$ are given in IMU coordinates.

The use of Euler angles for representing the orientation of the robot is valid in this application, because configurations which cause the Euler angle gimbal lock problem (such as 90° pitch) will not be reached by the robot. Euler angles have been chosen because they provide an intuitive representation of orientation. For applications where gimbal lock can occur representations such as rotation vector or quaternions should be used. However, in the error state vector, the orientation error $\Delta\boldsymbol{\varphi}$ always contains small Euler angles, which are, thus, equivalent to the components of a rotation vector. This can easily be shown by applying small angle approximation when computing a rotation matrix from Euler angles and from a rotation vector.

The discrete time error state propagation originates from the inertial error dynamics [37] as

$$\Delta\boldsymbol{x}_t^- = \boldsymbol{A}_t \cdot \Delta\boldsymbol{x}_{t-1} \tag{24}$$

$$\boldsymbol{A}_t = \mathbf{I} - \begin{bmatrix} \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{R}_t^- \lfloor(\boldsymbol{a}_t - \boldsymbol{b}_{a,t}^-)\times\rfloor & \mathbf{0} & \boldsymbol{R}_t^- \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{R}_t^- & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \Delta t \tag{25}$$

$$\lfloor\boldsymbol{o}\times\rfloor = \begin{bmatrix} 0 & -o_z & o_y \\ o_z & 0 & -o_x \\ -o_y & o_x & 0 \end{bmatrix}, \tag{26}$$

where $\mathbf{I}$ is the identity matrix (not to confuse with the information amount $\boldsymbol{I}_t$), $\boldsymbol{a}_t = (a_{tx}, a_{ty}, a_{tz})^T$ is the acceleration measured by the IMU, $\boldsymbol{b}_{a,t}^-$ is the predicted accelerometer bias, $\boldsymbol{R}_t^-$ is the propagated rotation from the IMU coordinate system into the world coordinate system and Δt is the time difference between $t-1$ and t.

6.4 The Multisensor Data Fusion Process

An overview of one time step of the data fusion process is given in Fig. 4. First, the accelerations $\boldsymbol{a}_t$ and angular velocities $\boldsymbol{\omega}_t$ measured by the IMU are fed into a strapdown algorithm. Considering the state vector $\boldsymbol{x}_{t-1}$ from the previous filter step, this algorithm integrates the IMU measurements to velocity $\boldsymbol{v}_t^-$, position $\boldsymbol{p}_t^-$ and orientation Euler angles $\boldsymbol{\varphi}_t^-$. These values are the predicted

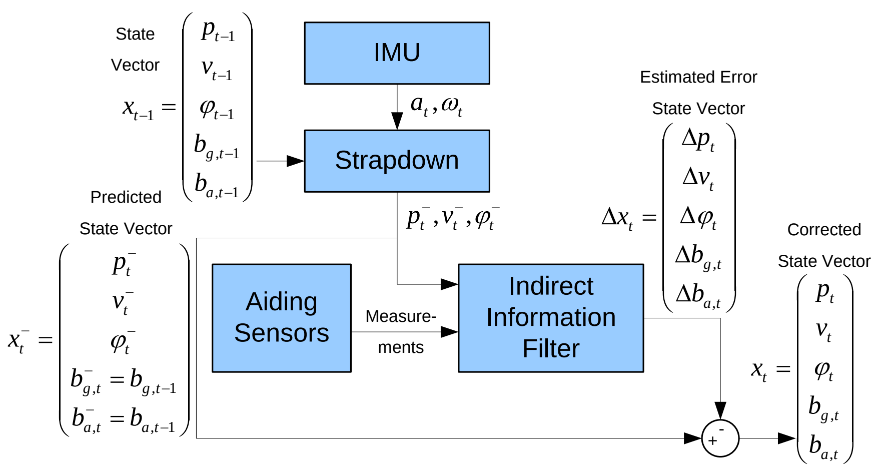

State Vector
$x_{t-1} = \begin{pmatrix} p_{t-1} \\ v_{t-1} \\ \varphi_{t-1} \\ b_{g,t-1} \\ b_{a,t-1} \end{pmatrix}$
IMU
a_t, ω_t
Strapdown
$p_t^-, v_t^-, \varphi_t^-$
Estimated Error State Vector
$\Delta x_t = \begin{pmatrix} \Delta p_t \\ \Delta v_t \\ \Delta \varphi_t \\ \Delta b_{g,t} \\ \Delta b_{a,t} \end{pmatrix}$
Predicted State Vector
$x_t^- = \begin{pmatrix} p_t^- \\ v_t^- \\ \varphi_t^- \\ b_{g,t}^- = b_{g,t-1} \\ b_{a,t}^- = b_{a,t-1} \end{pmatrix}$
Aiding Sensors
Measure-ments
Indirect Information Filter
Corrected State Vector
$x_t = \begin{pmatrix} p_t \\ v_t \\ \varphi_t \\ b_{g,t} \\ b_{a,t} \end{pmatrix}$
+
-

Figure 4: Overview of the multisensor data fusion process

state variables. The bias values $\boldsymbol{b}_{a,t}^-$ and $\boldsymbol{b}_{g,t}^-$ are predicted to be equal to the bias values of the last filter step. Every time one or more measurements of the aiding sensors are available, the indirect information filter is run and gives an estimated error state vector $\Delta\boldsymbol{x}_t$. This error state vector is then subtracted from the predicted state vector $\boldsymbol{x}_t^-$ to feedback the error. The result is the corrected state vector $\boldsymbol{x}_t$. If no measurements of the aiding sensors are available, the error state vector is zero and the corrected state will be the predicted state. The strapdown block and the information filter block are described in more detail in the following sections.

6.4.1 The Strapdown Algorithm

The accelerations and angular velocities of the IMU are measured in the IMU coordinate system. Since the IMU moves, the accelerations have to be transformed into the world coordinate system before integrating them. For this, the rotation matrix $\boldsymbol{R}_t$, which turns a vector from the IMU coordinate system into the world coordinate frame, has to be computed.

The rotation matrix can be propagated using the gyroscope measurements $\boldsymbol{\omega}_t$. Assuming a high sampling rate (Δt is small), the propagation of the rotation matrix can be performed as follows [3]:

$$\boldsymbol{R}_t^- = \boldsymbol{R}_{t-1}\boldsymbol{R}_{\Delta,t} \tag{27}$$

$$\boldsymbol{R}_{\Delta,t} = \mathbf{I} + \frac{\sin|\boldsymbol{\phi}_t|}{|\boldsymbol{\phi}_t|}\lfloor\boldsymbol{\phi}_t\times\rfloor + \frac{1-\cos|\boldsymbol{\phi}_t|}{|\boldsymbol{\phi}_t|^2}\lfloor\boldsymbol{\phi}_t\times\rfloor^2 \tag{28}$$

$$|\boldsymbol{\phi}_t| = \sqrt{\phi_{x,t}^2 + \phi_{y,t}^2 + \phi_{z,t}^2} \tag{29}$$

$$\boldsymbol{\phi}_t = (\boldsymbol{\omega}_t - \boldsymbol{b}_{g,t}^-)\Delta t. \tag{30}$$

$\boldsymbol{R}_t^-$ is the propagated rotation matrix, which is computed from the rotation matrix $\boldsymbol{R}_{t-1}$ of the last time step and a differential rotation $\boldsymbol{R}_{\Delta,t}$. The variable $\boldsymbol{\phi}_t$ is the rotation vector.

Knowing the rotation matrix, the IMU velocity $\boldsymbol{v}_t^-$ and position $\boldsymbol{p}_t^-$ can be computed. The acceleration measurements $\boldsymbol{a}_t$ have to be compensated for bias $\boldsymbol{b}_{a,t}^-$, transformed into the world frame using $\boldsymbol{R}_t^-$ and the gravity vector $\boldsymbol{g} = (0, 0, -9.80665)^T$ must be compensated:

$$\begin{aligned} \boldsymbol{v}_t^- &= \boldsymbol{v}_{t-1} + (\boldsymbol{R}_t^-(\boldsymbol{a}_t - \boldsymbol{b}_{a,t}^-) + \boldsymbol{g})\Delta t && (31) \\ \boldsymbol{p}_t^- &= \boldsymbol{p}_{t-1} + \boldsymbol{v}_{t-1}\Delta t + \tfrac{1}{2}(\boldsymbol{R}_t^-(\boldsymbol{a}_t - \boldsymbol{b}_{a,t}^-) + \boldsymbol{g})\Delta t^2. && (32) \end{aligned}$$

6.4.2 The Indirect Information Filter

Within the indirect information filter relative and absolute measurements are used to compute the estimated error state vector. While absolute measurements only depend on the current state of the system, relative measurements contain a difference between the current system state and a previous state. Since Kalman filter theory assumes that a measurement only depends on the current state of the system, relative measurements have to be treated in a special way. When this fact is ignored, the data fusion filter might also give good results at the first glance. However, viewed more closely, the resulting estimated variances are not feasible: For example, when fusing an IMU with only relative position measurements, one would expect that the estimated position variance grows with time, because summing up relative position measurements results in a drifting position estimate. However, if the relative character of the position measurements is ignored, the resulting position variances are estimated small and constant over time. This can cause serious problems when using another position sensor such as GPS, which gives absolute but noisy position measurements. These measurements will not influence the estimated position to the expected extent because the position estimate after fusing only the relative measurements is overconfident. Even if no absolute measurement was available for a longer time and position drift is significant, the estimated position variance would be small.

To avoid this problem, the state vector and covariance matrix have to be augmented to also contain the previous state which is part of the relative measurement. This approach was described by Roumeliotis et al. [33] and termed "Stochastic Cloning". This method introduces the correlations between the current and the previous state and hence allows to estimate a correct covariance matrix, with growing variances over time if only relative measurements are available.

To keep the augmented covariance matrix small, we chose to only clone the covariances associated to the states $\boldsymbol{p}_t$ and $\boldsymbol{\varphi}_t$, because only relative position and rotation measurements are used. At each time $t = t_{\text{Start}}$ when at least one relative measurement starts, the covariance matrix is augmented as follows:

$$\check{\boldsymbol{x}}_t = \begin{bmatrix} \boldsymbol{p}_t \\ \boldsymbol{\varphi}_t \end{bmatrix} \qquad \check{\boldsymbol{P}}_t = \mathbf{Cov}(\check{\boldsymbol{x}}_t, \check{\boldsymbol{x}}_t) \tag{33}$$

$$\boldsymbol{P}_t^{\text{aug}} = \begin{bmatrix} \check{\boldsymbol{P}}_{t_{\text{Start}}} & \mathbf{Cov}(\check{\boldsymbol{x}}_{t_{\text{Start}}}, \boldsymbol{x}_t) \\ \mathbf{Cov}(\boldsymbol{x}_t, \check{\boldsymbol{x}}_{t_{\text{Start}}}) & \boldsymbol{P}_t \end{bmatrix}, \tag{34}$$

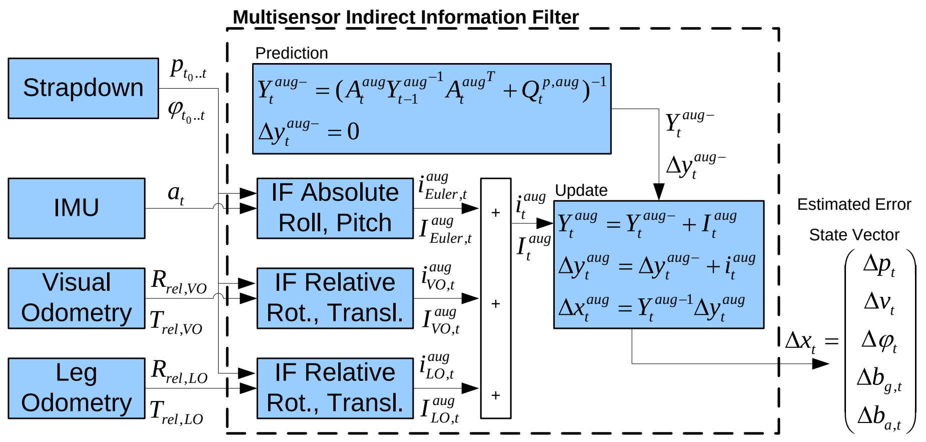
where $\mathbf{Cov}(\boldsymbol{x}_t, \check{\boldsymbol{x}}_{t_{\text{Start}}})$ is the covariance between the states at time t and the cloned states at t_{Start}. Since the covariance $\check{\boldsymbol{P}}_{t_{\text{Start}}}$ must not change during prediction of the filter, the system matrix $\boldsymbol{A}_t^{\text{aug}}$ and the process noise matrix

$\boldsymbol{Q}_t^{p,\mathrm{aug}}$ become

$$\boldsymbol{A}_t^{\mathrm{aug}} = \mathbf{blkdiag}\left[\mathbf{I}, \boldsymbol{A}_t\right] \tag{35}$$
$$\boldsymbol{Q}_t^{p,\mathrm{aug}} = \mathbf{blkdiag}\left[\mathbf{0}, \boldsymbol{Q}_t^{p}\right], \tag{36}$$

where $\mathbf{blkdiag}\left[\boldsymbol{U}, \boldsymbol{V}\right]$ stands for a block diagonal matrix with the matrices $\boldsymbol{U}, \boldsymbol{V}$ on its main diagonal.

Since in the information filter the inverse covariance is used, it must be ensured that in the prediction step (15) $\boldsymbol{A}_t^{\mathrm{aug}}\boldsymbol{P}_t^{\mathrm{aug}}(\boldsymbol{A}_t^{\mathrm{aug}})^T + \boldsymbol{Q}_t^{p,\mathrm{aug}}$ is invertible. For that reason, if two different relative measurements start at the same time, cloning is applied only once to keep the covariance matrix full rank. If measurements start at different times, the covariances between the different previous states also have to be cloned correctly. At the end of each relative measurement, the corresponding covariances are deleted from the augmented covariance matrix because they are not needed any longer after the relative measurement was processed. However, in this application, usually a relative measurement starts at the same time the previous measurement ends. Thus, after deleting a previous state, the current state is cloned for augmenting the covariance matrix again. In this application, relative measurements from visual odometry and leg odometry are usually taken at different rates. Thus, the augmented state vector usually contains two different previous positions and orientations, each corresponding to the starting time of a relative measurement.



Multisensor Indirect Information Filter
Prediction
$Y_t^{aug-} = (A_t^{aug} Y_{t-1}^{aug^{-1}} A_t^{aug^T} + Q_t^{p,aug})^{-1}$
$\Delta y_t^{aug-} = 0$
Strapdown
$p_{t_0..t}$
$\varphi_{t_0..t}$
Y_t^{aug-}
Δy_t^{aug-}
IMU
a_t
IF Absolute Roll, Pitch
$i_{Euler,t}^{aug}$
$I_{Euler,t}^{aug}$
i_t^{aug}
I_t^{aug}
Update
$Y_t^{aug} = Y_t^{aug-} + I_t^{aug}$
$\Delta y_t^{aug} = \Delta y_t^{aug-} + i_t^{aug}$
$\Delta x_t^{aug} = Y_t^{aug-1} \Delta y_t^{aug}$
Estimated Error State Vector
Visual Odometry
$R_{rel,VO}$
$T_{rel,VO}$
IF Relative Rot., Transl.
$i_{VO,t}^{aug}$
$I_{VO,t}^{aug}$
Leg Odometry
$R_{rel,LO}$
$T_{rel,LO}$
IF Relative Rot., Transl.
$i_{LO,t}^{aug}$
$I_{LO,t}^{aug}$
+
+
+
$\Delta x_t = \begin{pmatrix} \Delta p_t \\ \Delta v_t \\ \Delta \varphi_t \\ \Delta b_{g,t} \\ \Delta b_{a,t} \end{pmatrix}$

Figure 5: Overview of the multisensor data fusion information filter

The data flow within the indirect information filter for multisensor data fusion is shown in Fig. 5. First, the augmented information matrix is predicted. Then, for each available sensor measurement k at time step t the values for $\boldsymbol{i}_{k,t}^{\mathrm{aug}}$ and $\boldsymbol{I}_{k,t}^{\mathrm{aug}}$ are computed using the differences between the strapdown algorithm results and the sensor measurements. Then, all available information amounts $\boldsymbol{I}_{k,t}^{\mathrm{aug}}$ and information contributions $\boldsymbol{i}_{k,t}^{\mathrm{aug}}$ are summed up and the update equations are performed. In the end, the resulting information vector is transformed into an error state vector from which the cloned states are deleted. These steps are described in more detail in the following sections.

Prediction Using the state transition matrix $\boldsymbol{A}_t^{\text{aug}}$ as given in (25) and (35), the information matrix $\boldsymbol{Y}_t^{\text{aug}-}$ is predicted using (16). The prediction of the information vector simply becomes $\Delta\boldsymbol{y}_t^{\text{aug}-} = 0$ because in the indirect feedback information filter the error is corrected after each filter step.

Absolute Roll and Pitch Angle Measurements Since the accelerometers of the IMU sense the gravity, which is known in size and direction with respect to the world frame, it is possible to determine the absolute roll and pitch angles γ_{abs} and β_{abs} of the acceleration measurement $\boldsymbol{a} = [a_x, a_y, a_z]^T$ as follows:

$$\gamma_{\text{abs}} = \text{atan2}(a_y, a_z), \tag{37}$$
$$\beta_{\text{abs}} = \text{atan2}(-a_x, a_y \sin\gamma_{\text{abs}} + a_z \cos\gamma_{\text{abs}}). \tag{38}$$

From the absolute roll and pitch angles, an absolute rotation matrix $\boldsymbol{R}_{\text{abs}}$ can be computed using

$$\boldsymbol{R}_{\text{abs}} = \begin{bmatrix} \text{c}\beta\text{c}\alpha & \text{s}\gamma\text{s}\beta\text{c}\alpha - \text{c}\gamma\text{s}\alpha & \text{c}\gamma\text{s}\beta\text{c}\alpha + \text{s}\gamma\text{s}\alpha \\ \text{c}\beta\text{s}\alpha & \text{s}\gamma\text{s}\beta\text{s}\alpha + \text{c}\gamma\text{c}\alpha & \text{c}\gamma\text{s}\beta\text{s}\alpha - \text{s}\gamma\text{c}\alpha \\ -\text{s}\beta & \text{s}\gamma\text{c}\beta & \text{c}\gamma\text{c}\beta \end{bmatrix}$$
$$\text{s}\varphi = \sin\varphi_{\text{abs}} \qquad \text{c}\varphi = \cos\varphi_{\text{abs}} \tag{39}$$

For this, the yaw angle α_{abs} is set to be equal to the yaw angle of the propagated rotation matrix $\boldsymbol{R}_t^-$ because it cannot be determined from the acceleration measurements.

The absolute roll and pitch angles obtained from the acceleration measurements contain a high level of noise. The noise is caused by additional accelerations that occur when the robot moves. Hence, the absolute noisy angles must be fused with low-noise angular measurements. The propagated rotation matrix $\boldsymbol{R}_t^-$ as computed in (27)-(30) contains the roll and pitch angles from integrating the gyroscope measurements. These angles do not suffer from high noise but from a drift caused by integrating the sensor values. By fusing $\boldsymbol{R}_{\text{abs}}$ and $\boldsymbol{R}_t^-$, the roll and pitch Euler angles can be determined quite accurately without drift and high noise. The difference rotation matrix between the propagated rotation $\boldsymbol{R}_t^-$ and the absolute rotation $\boldsymbol{R}_{\text{abs}}$ is computed as

$$\boldsymbol{R}_{\text{diff}} = \boldsymbol{R}_t^- \cdot \boldsymbol{R}_{\text{abs}}^T. \tag{40}$$

Using the equations

$$\begin{aligned} \alpha &= \quad \text{atan2}(\boldsymbol{R}_{(2,1)}, \boldsymbol{R}_{(1,1)}) \\ \beta &= \quad \text{atan2}(-\boldsymbol{R}_{(3,1)}, \boldsymbol{R}_{(2,1)} \sin\alpha + \boldsymbol{R}_{(1,1)} \cos\alpha) \\ \gamma &= \quad \text{atan2}(\boldsymbol{R}_{(1,3)} \sin\alpha - \boldsymbol{R}_{(2,3)} \cos\alpha, \\ & \quad - \boldsymbol{R}_{(1,2)} \sin\alpha + \boldsymbol{R}_{(2,2)} \cos\alpha) \end{aligned} \tag{41}$$

to extract Euler angles from the elements $\boldsymbol{R}_{(i,j)}$ of a rotation matrix, the angle differences γ_{diff} and β_{diff} can be computed from $\boldsymbol{R}_{\text{diff}}$ which give the measurement vector $\boldsymbol{z}_{\text{Euler},t}$:

$$\boldsymbol{z}_{\text{Euler},t} = \begin{bmatrix} \gamma_{\text{diff}} \\ \beta_{\text{diff}} \end{bmatrix} \tag{42}$$

The measurement matrix $\boldsymbol{H}_{\text{Euler},t}$ which projects the state vector $\boldsymbol{x}_t$ onto the measurement vector $\boldsymbol{z}_{\text{Euler},t}$ is

$$\boldsymbol{H}_{\text{Euler},t} = \begin{bmatrix} \mathbf{0}_{2\times6} & \mathbf{I}_{2\times2} & \mathbf{0}_{2\times7} \end{bmatrix} \tag{43}$$

For the augmented state vector, the measurement matrix has to be augmented with zeros to

$$\boldsymbol{H}^{\text{aug}}_{\text{Euler},t} = \begin{bmatrix} \mathbf{0} & \boldsymbol{H}_{\text{Euler},t} \end{bmatrix}, \tag{44}$$

because the measurement does not depend on any previous states but is absolute. The measurement noise matrix $\boldsymbol{Q}^{m}_{\text{Euler},t}$ contains the variances of the absolute roll and pitch angle measurements and can be found by filter tuning.

Knowing $\boldsymbol{z}_{\text{Euler},t}, \boldsymbol{H}^{\text{aug}}_{\text{Euler},t}, \boldsymbol{Q}^{m}_{\text{Euler},t}$ the information contribution $\boldsymbol{i}^{\text{aug}}_{\text{Euler},t}$ and the information amount $\boldsymbol{I}^{\text{aug}}_{\text{Euler},t}$ are computed using (19)-(20).

Using absolute angles obtained by accelerometer data as measurements for the data fusion filter violates Kalman filter theory which assumes that measurement noise and process noise are uncorrelated. Hence, the filter result is suboptimal. However, the suboptimal filter result is still better than not using absolute roll and pitch angle measurements for limiting the drift of the orientation estimates.

Relative Translation and Rotation Measurements The relative motion measurements have to be fused with the relative rotations and translations computed by the strapdown algorithm within the same time period. Visual odometry as well as leg odometry provide relative position and orientation measurements between two consecutive images or robot poses, respectively. Visual odometry and leg odometry are fused in the same way as relative translation and rotation measurements. Thus, they will both be referred to as "odometry sensor" and not be distinguished in the next paragraphs.

A relative measurement has two timestamps t_{start} and t_{end} at the beginning and the end of the relative measurement. Furthermore, for fusing relative rotations and translations, all values must be represented in the same coordinate system. That means, the relative measurements of all sensors have to be transformed into relative measurements in the IMU coordinate frame in order to be fused with IMU measurements. That also means, that the transformations between the different sensor coordinate frames must be known, either by design or by calibration.

The differences between the relative motion given by the strapdown algorithm in the time interval from t_{start} to t_{end} and the relative motion measured by the odometry sensor give the measurement vector $\boldsymbol{z}_{\text{rel},t}$. In order to compute the difference between two relative rotations $\boldsymbol{R}^{I}_{\text{rel}}$ measured by the IMU and $\boldsymbol{R}^{S}_{\text{rel}}$ measured by an odometry sensor, an absolute rotation matrix has to be computed. To preserve the relative character of the measurements, both relative rotations have to be multiplied with the same absolute rotation matrix $\boldsymbol{R}_{t_{\text{start}}}$ to get pseudo-absolute rotation measurements. This absolute rotation matrix $\boldsymbol{R}_{t_{\text{start}}}$ should be the best estimate of the rotation from the IMU into the world frame at time step t_{start}:

$$\boldsymbol{R}^{I}_{t_{\text{end}}} = \boldsymbol{R}_{t_{\text{start}}}\boldsymbol{R}^{I}_{\text{rel}}, \qquad \boldsymbol{R}^{S}_{t_{\text{end}}} = \boldsymbol{R}_{t_{\text{start}}}\boldsymbol{R}^{S}_{\text{rel}}. \tag{45}$$

Now the rotational difference matrix can be computed as

$$\boldsymbol{R}_{\text{diff}} = \boldsymbol{R}^{I}_{t_{\text{end}}} \cdot (\boldsymbol{R}^{S}_{t_{\text{end}}})^{T}. \tag{46}$$

The measurement vector $\boldsymbol{z}_{\text{rel},t}$ contains the differences $\boldsymbol{p}_{\text{diff}}$ between the two relative translations and the angle differences $\boldsymbol{\varphi}_{\text{diff}}$ computed from $\boldsymbol{R}_{\text{diff}}$ using (41):

$$\boldsymbol{z}_{\text{rel},t} = \left[\boldsymbol{p}_{\text{diff}}, \boldsymbol{\varphi}_{\text{diff}}\right]^{T}. \tag{47}$$

The augmented measurement matrix $\boldsymbol{H}^{\text{aug}}_{\text{rel},t}$ which projects the augmented state vector $\Delta\boldsymbol{x}^{\text{aug}}_{t}$ onto the measurement vector $\boldsymbol{z}_{\text{rel},t}$ is

$$\boldsymbol{H}^{\text{aug}}_{\text{rel},t} = \begin{bmatrix} -\boldsymbol{H}_{\text{rel},t_{\text{Start}}} & \boldsymbol{H}_{\text{rel},t} \end{bmatrix}. \tag{48}$$

$$\boldsymbol{H}_{\text{rel},t} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times6} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times6} \end{bmatrix}. \tag{49}$$

In the measurement matrix $\boldsymbol{H}^{\text{aug}}_{\text{rel},t}$ the relative character of the measurements must be represented. This is achieved by the matrix $\boldsymbol{H}_{\text{rel},t_{\text{Start}}}$ which contains an identity matrix in the columns corresponding to the location of the cloned covariance of time t_{Start} and zeros everywhere else.

The measurement noise matrix $\boldsymbol{Q}^{m}_{\text{rel},t}$ is computed from the standard deviations of the relative position and rotation measurements. For leg odometry, the measurements errors depend on how much the feet of the robot slip on the ground. On homogeneous ground the amount of slippage can be assumed constant and found by filter tuning. Using visual odometry, assuming constant standard deviations for the relative motion measurements is not appropriate for environments with changing light or texture conditions. Thus, the estimated errors of each visual odometry measurement (ref. Sec. 5.3) are transformed into the IMU coordinate system using error propagation and then fed into the measurement noise matrix.

Knowing $\boldsymbol{z}_{\text{rel},t}, \boldsymbol{H}^{\text{aug}}_{\text{rel},t}, \boldsymbol{Q}^{m}_{\text{rel},t}$ the information contribution $\boldsymbol{i}^{\text{aug}}_{\text{rel},t}$ and the information amount $\boldsymbol{I}^{\text{aug}}_{\text{rel},t}$ are computed using (19)-(20).

Update At every time step, $\boldsymbol{i}^{\text{aug}}_{k,t}$ and $\boldsymbol{I}^{\text{aug}}_{k,t}$ of each available sensor measurement are computed. In the final step of the multisensor information filter, these values are summed and used to update the predicted information vector and information matrix using (19)-(20). Finally, the resulting information vector $\Delta\boldsymbol{y}^{\text{aug}}_{t}$ is transformed into an error state vector $\Delta\boldsymbol{x}^{\text{aug}}_{t}$. From $\Delta\boldsymbol{x}^{\text{aug}}_{t}$, $\Delta\boldsymbol{x}_{t}$ is extracted, which contains the estimated errors of the single robot states. By inverting the resulting information matrix $\boldsymbol{Y}^{\text{aug}}_{t}$, the covariance matrix $\boldsymbol{P}^{\text{aug}}_{t}$ can computed if required and $\boldsymbol{P}_{t}$ can be extracted.

6.4.3 Error State Feedback

To correct the position, velocity and bias values of the predicted state vector $\boldsymbol{x}^{-}_{t}$, the corresponding error estimates from the error state vector $\Delta\boldsymbol{x}_{t}$ are subtracted. For feeding back the estimated rotation angle error $\Delta\boldsymbol{\varphi}_{t}$, a rotation matrix $\boldsymbol{R}_{\text{corr}}$ has to be computed from $\Delta\boldsymbol{\varphi}_{t}$ using

$$\boldsymbol{R}_{\text{corr}} = \mathbf{I} + \lfloor \Delta\boldsymbol{\varphi}_{t} \times \rfloor \tag{50}$$

and correction is performed as

$$\boldsymbol{R}_t = \boldsymbol{R}_{\text{corr}}^T \cdot \boldsymbol{R}_t^- . \tag{51}$$

From $\boldsymbol{R}_t$ the corrected Euler angles can be extracted via (41).

6.5 Filter Initialization

In the beginning of the data fusion process the robot is motionless in its starting position. This phase can be used for filter initialization.

From the very first IMU measurement, the starting orientation $\boldsymbol{R}_{t_0}$ with respect to the gravity vector is estimated from the acceleration measurements $\boldsymbol{a}_{t_0}$ as shown in (37)-(38). Furthermore, the bias estimates $\boldsymbol{b}_{a,t_0}$ and $\boldsymbol{b}_{g,t_0}$ are initialized using the starting orientation, the known gravity vector $\boldsymbol{g}$ and the gyroscope measurements $\boldsymbol{\omega}_{t_0}$. Good starting values for the sensor biases are

$$\boldsymbol{b}_{a,t_0} = \boldsymbol{a}_{t_0} + \boldsymbol{R}_{t_0}^T \cdot \boldsymbol{g} \tag{52}$$

$$\boldsymbol{b}_{g,t_0} = \boldsymbol{\omega}_{t_0}. \tag{53}$$

From the following IMU measurements, the estimates of the bias values and the starting orientation can be refined exploiting the fact that the robot does not move. Hence, position, velocity and orientation measurements with the value of zero and small noise matrices are fed into the information filter. As a result the bias value estimation stabilizes. Furthermore, the absolute roll and pitch angle measurements from the accelerations are fused with the orientation measurements from the gyroscopes as described in section 6.4.2. The initialization phase is finished when the change in the bias estimates drops below a threshold. This process usually takes a few seconds. Once the information filter is initialized, the robot can start moving and visual odometry and leg odometry measurements are used.

6.6 Filter Results

Results of using the multisensor data fusion filter for position estimation compared to using only visual or leg odometry are shown in Fig. 6. In this experiment, the Crawler was steered manually along a rectangular path in a testbed filled with gravel. An external infrared tracking system provided ground truth trajectories. Fig. 6(a) shows the test setup with the robot in its starting pose and the approximate steered path. Fig. 6(b) shows the ground truth trajectory measured by the tracking system, the fusion result and the different odometry trajectories which were obtained by summing the relative measurements of the respective sensors. The trajectory computed using only the IMU measurements is not shown here because its enormous drift leads to an error of more than 100 m after 60 s runtime. The visual odometry trajectory is quite accurate apart from a small drift of the yaw angle. The leg odometry trajectory shows that yaw angles are overestimated because of slip in the gravel (ref. Fig. 6(d)). The fusion trajectory is very close to the ground truth path. Fig. 6(c) shows plots of the z-coordinates. While visual odometry and leg odometry drift due to roll and pitch angle errors, the estimated z-coordinate of the fusion result remains close to the ground truth curve because of absolute roll and pitch angle measurements from the accelerometers. Fig. 6(e) shows the standard deviations of the position estimates computed from the estimation covariance matrix. As can be seen, the standard deviations grow with time since no absolute position measurements are available. The detailed plot in this figure illustrates the influence of the relative measurements on the covariance: Visual odometry measurements usually have lower uncertainty than leg odometry measurements and, thus, reduce the estimation covariance more than the leg odometry measurements. However, during turning in the corners of the testbed, the errors of the visual odometry measurements are higher. The reason for that is the texture of the testbed walls which is worse than the texture of the gravel. Hence, the covariances increase stronger during these periods. A more detailed overview of the filter performance including experimental results under poor visual conditions are given in [7].

7 Mapping

A digital terrain model (DTM), which is incrementally built from the depth images computed by SGM, 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.

Mapping is performed in two steps. First, a local DTM is created from a single depth image. After that, the local DTM is attached to the global DTM.

The traditional method for creating a local DTM from a range measurement such as a depth image is the reconstruction of the 3D coordinates $\boldsymbol{P}^c = (P_x^c, P_y^c, P_z^c)$ in the camera coordinate frame c from the image points $p_{lx}^i, p_{ly}^i, p_d^i$ in the image coordinate frame i using

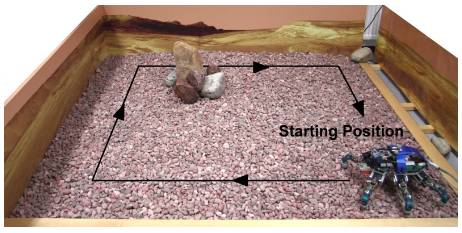
$$P_x^c = \frac{t \cdot p_{lx}^i}{p_d^i} = \frac{t \cdot p_{lx}^i}{p_{lx}^i - p_{rx}^i} \tag{54}$$

$$P_y^c = \frac{t \cdot p_{ly}^i}{p_d^i} = \frac{t \cdot (p_{ly}^i + p_{ry}^i)}{2 \cdot (p_{lx}^i - p_{rx}^i)} \tag{55}$$

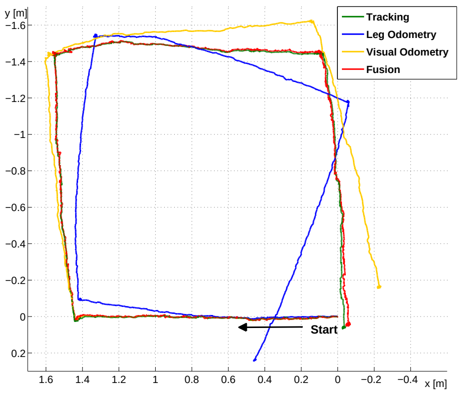
$$P_z^c = \frac{t \cdot f}{p_d^i} = \frac{t \cdot f}{p_{lx}^i - p_{rx}^i} \tag{56}$$

with f as focal length in pixels and t as stereo baseline. The resulting point cloud is then projected onto the grid. This approach was presented in [6]. Although the stereo disparity image is dense, this approach can result in sparse maps because the optical axis of the camera is usually not perpendicular to the surface but the surface is viewed at a certain angle.

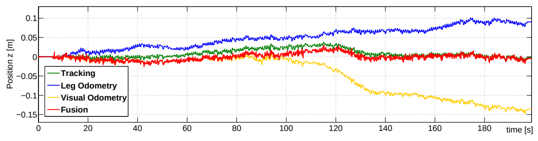
The locus method [26] is an efficient way to create dense height maps with arbitrary resolution from range images. It finds the elevation z at a point (u_x, u_y) of a reference plane by computing the intersection of the terrain with a vertical line at (u_x, u_y). The projection of the vertical line on the image is called locus. The intersection point with the terrain is computed in image space rather than Cartesian space. For each cell in the DTM a vector representing a



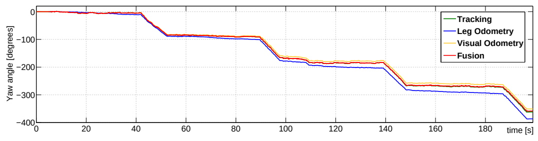
(a) Test setup and steered trajectory



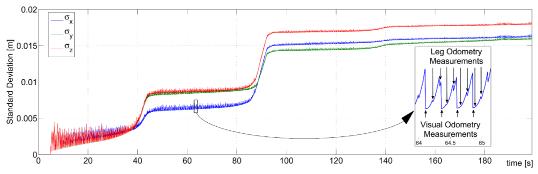
(b) Recorded trajectories



(c) z-coordinates



(d) Yaw angles



(e) Standard deviations computed from the estimation covariance matrix

Figure 6: Test run for filter performance evaluation

vertical line $\boldsymbol{l}$ is computed as

$$\boldsymbol{l} = (\boldsymbol{u}, \boldsymbol{v}) = \left([u_x, u_y, u_z]^T, [v_x, v_y, v_z]^T \right), \tag{57}$$

where $\boldsymbol{u}$ is a point and $\boldsymbol{v}$ is a unit directional vector. Each point $\boldsymbol{r}$ on the line is then represented by

$$\boldsymbol{r} = \boldsymbol{u} + \lambda \boldsymbol{v}. \tag{58}$$

For a vertical line in world coordinates the values of $\boldsymbol{u}$ and $\boldsymbol{v}$ become

$$\boldsymbol{u} = [u_x, u_y, 0]^T \tag{59}$$

$$\boldsymbol{v} = [0, 0, 1]^T. \tag{60}$$

Using the rotation matrix $\boldsymbol{R}_c^w$ which turns a point from the camera coordinate frame c to the world coordinate frame w and the translation $\boldsymbol{t}^w$ from the world coordinate frame to the camera coordinate frame, this line can be transformed into camera coordinates:

$$\boldsymbol{l}^c = (\boldsymbol{u}, \boldsymbol{v}) = \left((\boldsymbol{R}_c^w)^T(\boldsymbol{u} - \boldsymbol{t}^w), (\boldsymbol{R}_c^w)^T\boldsymbol{v}\right). \tag{61}$$

By projecting $\boldsymbol{l}^c$ onto the disparity image, a generalized locus is obtained. The intersection of the locus with the disparity profile gives the elevation of the grid cell at (u_x, u_y). The projections $\boldsymbol{r}^i = (r_{lx}^i, r_{ly}^i, r_d^i)^T$ of all points $\boldsymbol{r}^c$ of line $\boldsymbol{l}^c$ onto the image also form a line $\boldsymbol{l}^i$ with

$$r_d^i = \frac{f \cdot t}{r_z^c} \tag{62}$$

$$r_{lx}^i = \frac{r_d^i \cdot r_x^c}{t} = \frac{f \cdot r_x^c}{r_z^c} \tag{63}$$

$$r_{ly}^i = \frac{r_d^i \cdot r_y^c}{t} = \frac{f \cdot r_y^c}{r_z^c}. \tag{64}$$

The intersection of $\boldsymbol{l}^i$ with the disparity profile of the image has to be computed. Due to the orientation of the camera with respect to the world coordinate system, it is beneficial to parameterize the locus line by the image row coordinate r_{ly}^i, because a vertical line in world coordinates will most likely run through all rows in the image, but will only appear in few columns. Hence, line equations for the column $r_{lx}^i = f(r_{ly}^i) = a_x r_{ly}^i + n_x$ and disparity $r_d^i = f(r_{ly}^i) = a_d r_{ly}^i + n_d$ can be created by transforming two distant points $\boldsymbol{r}_1^c$ and $\boldsymbol{r}_2^c$ on $\boldsymbol{l}^c$ into image coordinates $(r_{lx1}^i, r_{ly1}^i, r_{d1}^i)$ and $(r_{lx2}^i, r_{ly2}^i, r_{d2}^i)$ and determining the line equation parameters slope a and offset n for disparity (index d) and column (index x).

$$a_x = \frac{r_{lx1}^i - r_{lx2}^i}{r_{ly1}^i - r_{ly2}^i} \qquad n_x = r_{lx1}^i - a_x \cdot r_{ly1}^i \tag{65}$$

$$a_d = \frac{r_{d1}^i - r_{d2}^i}{r_{ly1}^i - r_{ly2}^i} \qquad n_d = r_{d1}^i - a_d \cdot r_{ly1}^i \tag{66}$$

The first step for finding the intersection is to search two sample points $(r_{lx1}^i, r_{ly1}^i, r_{d1}^i)$ and $(r_{lx2}^i, r_{ly2}^i, r_{d2}^i)$ on the line which fulfil the condition

$$r_{d1}^i < p_d^i(r_{lx1}^i, r_{ly1}^i) \tag{67}$$

$$r_{d2}^i > p_d^i(r_{lx2}^i, r_{ly2}^i). \tag{68}$$

For doing so, the first sample point is searched for by starting at the bottom row of the image; searching for the second sample point starts at the top row.

For each row, the disparity and column values can be computed using the line equations above.

After finding those two sample points, binary search can be applied between these points to find the intersection $r^i_{ilx}, r^i_{ily}, r^i_{id}$ with $r^i_{id} = p^i_d(r^i_{ilx}, r^i_{ily})$. It should be noted that there are image pixels which do not have a valid disparity value. If the search reaches such a pixel, the search terminates and a linear search is performed between the current interval boundaries. If the search interval cannot be reduced any further and the disparity difference between the boundary values $p^i_d(r^i_{lx1}, r^i_{ly1})$ and $p^i_d(r^i_{lx2}, r^i_{ly2})$ is within a certain threshold (range shadows), the intersection is found and calculated as

$$r^i_{ilx} = \frac{r^i_{lx1} + r^i_{lx2}}{2} \tag{69}$$

$$r^i_{ily} = \frac{r^i_{ly1} + r^i_{ly2}}{2} \tag{70}$$

$$r^i_{id} = \frac{r^i_{d1} + r^i_{d2}}{2} \tag{71}$$

If multiple intersection points are found by the linear search, only the intersection with the highest elevation is considered. From the intersection point, the elevation value can be calculated using the reconstruction formulas in (54)-(56).

Fig. 7 shows the top views on local maps created by the traditional approach and by the locus method with colors indicating the elevations. As can be seen, the locus method gives denser terrain maps with less artifacts.

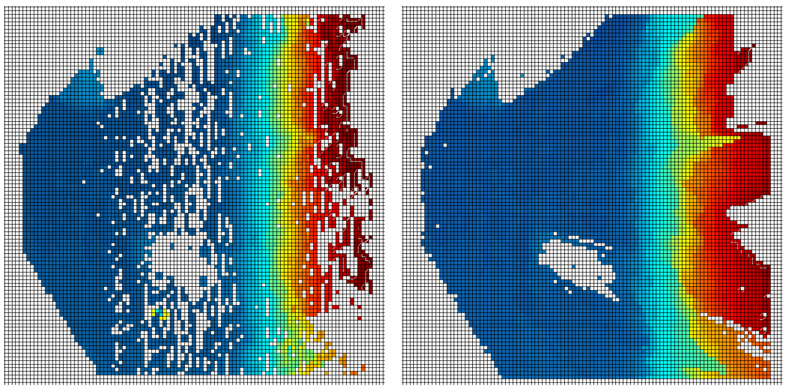

(a) Traditional approach (b) Locus method

Figure 7: Comparison of a local map created by the traditional and the locus method

Since in stereo vision the error of reconstructing the distance of an object point grows quadratically with the distance of that point from the camera, the size of the local map was limited to 1 m. That prevents erroneous range measurements from being inserted into the terrain map.

Each local DTM from a depth image is attached to the global DTM using the estimated robot pose at the time of image acquisition. Existing height values are overwritten by newer values. This approach is prone to errors from pose estimation, 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.

8 Traversability Estimation

Based on the global terrain model, the traversability of the cells is assessed as presented in [6]. Each cell is assigned a danger value d ($d \in \{[0,1], \infty\}$) describing the terrain difficulty. A cell is traversable if the robot is not exposed to critical terrain hazards irrespective of its orientation given its center is located in that cell. Thus, the robot can be treated as a point in further computations. A danger value of $d = 0$ stands for completely flat, smooth terrain, which can be traversed by the robot most easily. Higher danger values are assigned to areas which are harder to pass. A value of $d = 1$ describes terrain which is just barely traversable for the robot. Untraversable regions are assigned $d = \infty$. Unknown areas are assumed to be traversable but are assigned a high danger value of $d = 1$.

There are three potential hazards: steep slopes, high terrain roughness and high steps. Each of these criteria must be estimated from the DTM. If one of the hazard criteria exceeds the corresponding critical value, the cell is marked as untraversable. The critical values s_{crit}, r_{crit} and h_{crit} are the maximum slope, roughness and step height which the robot can traverse without tipping over or getting stuck. For traversable cells the danger value is computed from the three types of hazards as

$$d = \alpha_1 \frac{s}{s_{\text{crit}}} + \alpha_2 \frac{r}{r_{\text{crit}}} + \alpha_3 \frac{h}{h_{\text{crit}}}, \tag{72}$$

where α_1, α_2 and α_3 are weight parameters which sum up to 1.

Similar to the traversability estimation in GESTALT [12], the slope s of a cell is calculated by fitting a plane in a circular region around the cell with a diameter corresponding to the maximum diameter of the robot. The angle between the plane normal and the z-axis of the global coordinate frame gives the slope inclination s. The terrain roughness r is calculated as the standard deviation of the terrain height values from the computed plane in the circular region around the cell.

The step height h is computed in two steps. First, local height differences within a square window of several (e.g. 11×11) grid cells are computed for all cells in the circular region. If the maximum height difference between any cell in that window and the center cell of the window is greater than the critical step height h_{crit} and the slope between the corresponding two terrain points is higher than the critical slope s_{crit}, the maximum height difference is stored as the temporary step height of the central cell of the window. Second, the step height of the central cell of the circular region is computed as

$$h = \min(h_{\max}, h_{\max} \cdot \frac{n_{st}}{n_{\text{crit}}}), \tag{73}$$

where $h_{\max}$ is the maximum temporary step height in the circular region, n_{st} is the number of cells in the circular region whose temporary step heights are higher than the critical step height and n_{crit} is the valid number of cells (e.g. 50)

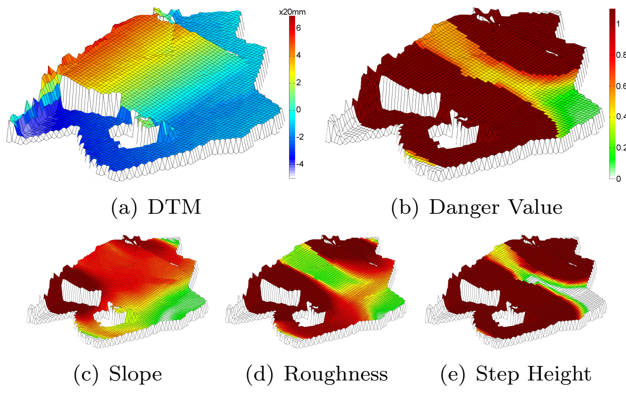


Figure 8: Danger value computation from the criteria slope, roughness and step height ($s_{\text{crit}} = 20°$, $r_{\text{crit}} = 30$ mm, $h_{\text{crit}} = 50$ mm, $\alpha_1 = 0.5$, $\alpha_2 = 0.25$, $\alpha_3 = 0.25$)

with a temporary step height higher than the critical step height. This method for calculating the step height also detects small steep slopes as steps and is robust to missing terrain information.

Fig. 8 illustrates the computation of the danger value from the three criteria. It shows that the step height is well suited for detecting whether a cell is traversable or not, but it provides little information about the difficulty of the traversable cells. By contrast, the slope and roughness criteria can fail to detect untraversable cells but are better suited for estimating the difficulty of traversable cells.

The terrain traversability is reestimated every time a new local map is added to the global map. It is not necessary to recompute danger values for the complete global map. Only the cells within the area of the new local map surrounded by a border of the size of the robot have to be reestimated.

As mentioned above, artifacts can be present in the DTM, which must not be detected as terrain hazards (ref. Fig. 9). For this reason, only height values which were detected within a certain range Δf of frame numbers are considered in the traversability estimation process. When reestimating the danger of a cell that was detected impassable in the previous estimation step, height values of a greater frame range $\Delta f' > \Delta f$ are considered in comparison to reestimating a previously traversable cell. Practical tests showed that taking only height values within Δf into account often causes untraversable cells to be detected as traversable cells because nearby hazards are not covered by the images. Thus, also considering older height values within $\Delta f'$ for reestimating the traversability of previously untraversable cells gives more realistic results.

The traversability of a cell is only computed if a sufficient number of height values is present in the circular rover-sized region around the cell. In addition to the danger value, a certainty value is calculated as the percentage of available height values in the circular region. This value is later required to decide whether the active exploration of an area is necessary.

The described traversability estimation approach depends on many parameters such as the size of the robot, the critical step height, roughness and slope values and the weight of the three criteria for calculating a danger value. These

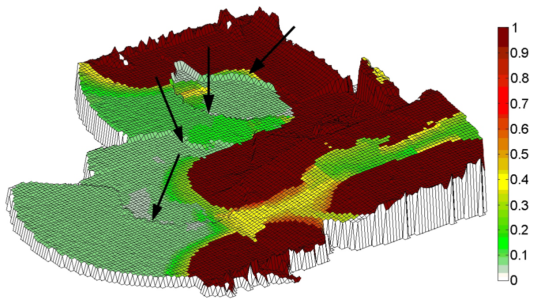


Figure 9: Traversability estimation of a DTM with artifacts caused by errors in position estimation: By considering only height values with frame numbers within $\Delta f = 20$ for passable and $\Delta f' = 200$ for impassable cells artifacts are not detected as hazards.

parameters allow to adapt the traversability estimation process to different types of robots. In addition to the DLR Crawler, we successfully use this method for generating traversability maps for different wheel-driven robots and for finding flat footholds for a biped walking robot. However, while some of the parameters are known from the design of the robot, others have to be found empirically in practical tests. Here, in future, a learning approach could be implemented so that each robot can learn suitable parameter values by itself.

9 Path Planning

Based on the traversability map, the robot can plan a path to the goal point. Since the robot does not have an a priori map of the environment, its knowledge about the terrain changes over time. The path planner must be able to adapt the path to changes in the map in an efficient way. Thus, a path planner of the D* family was chosen. D* developed by Stentz [36] is the dynamic version of the A* graph based path planning algorithm. These search algorithms find the minimum cost path to a goal vertex in a graph by first searching those vertexes which most likely appear to lead to the goal. In contrast to A* planners, D* is able to modify previous search results locally and, thus, is more efficient when dynamic replanning is required. For the present navigation system a D* Lite [24] path planner was used, which is simpler and more efficient than the classic D* algorithms.

To apply the D* Lite planner to the grid map, the map has to be considered as a graph. The grid cells are the vertexes of the graph and edges connect vertexes which correspond to adjacent cells in the grid map. As for the A* algorithm, a cost function and a heuristic distance function must be implemented for the D* Lite path planner. However, D* Lite plans a path in opposite direction from the goal vertex G to the start vertex S. The cost function $c(N, N')$ describes the cost for moving from vertex N to its neighbor N'. The heuristic distance function $h(N, S)$ is an estimate of the costs remaining to reach the start vertex from the current vertex N. The formulation of the cost function defines the optimality of a path. Often, a path is optimal if it is the shortest path to the goal. In the present work, not only the path length but also the traversability

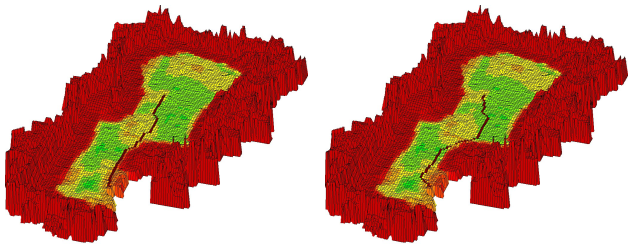

(a) $\beta = 3$ (b) $\beta = 10$

Figure 10: Paths planned with different values of β.

of the path cells should be taken into account. Thus, the cost function for going from vertex N to its neighbor N' is

$$c(N, N') = \sqrt{(N_x - N'_x)^2 + (N_y - N'_y)^2} + \beta \cdot d(N'). \tag{74}$$

The first term describes the distance between the vertexes and the second term denotes the danger value of the destination vertex weighted by $\beta > 1$. The bigger the value of β is chosen the longer paths are accepted if they go through safer cells (ref. Fig. 10). The costs of going to an untraversable cell are ∞.

The heuristic distance function is important for the efficiency of the planner. For the planner to be optimal it must fulfill the monotony condition

$$h(N, S) \leq c(N, N') + h(N', S). \tag{75}$$

That implies that $h(N, S)$ must not overestimate the true costs $h^*(N, S)$ of going from N to S along the shortest path. Since the minimum danger value of a cell is 0, the heuristic distance function must be

$$h(N, S) = \sqrt{(N_x - S_x)^2 + (N_y - S_y)^2}, \tag{76}$$

which is the direct distance from N to S. Due to the triangle inequality, this function also fulfills the monotony condition. This use of D* Lite for visual navigation has been presented in [6].

In the beginning, the robot does not have any information about its environment. It plans an initial path, which is the direct path to the goal. As the robot follows this path, it collects information about the environment. If assumptions about the traversability of the path cells are proven wrong by new data, the path is replanned from the current cell of the robot.

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. However, if the robot is carrying a heavy load or if its hardware is damaged, the path planner must adapt the path to the changed motion abilities of the robot. To avoid reassessing the traversability of the whole terrain map, a danger value threshold $0 \leq d_{\max} \leq 1$ can be set in the path planner. If the danger value of a cell is higher than $d_{\max}$, the costs of moving to that cell are set to ∞. Thus, the safety of the planned path is improved. Currently, the danger value threshold is set by the operator, but in future, the robot should be able to set the threshold according to its estimated motion capabilities. For this, the robot will have to learn which maximum danger value corresponds to its current motion capabilities. Furthermore, the value for β should also be learned by the robot instead of being set to a fixed value by the operator. The use of the danger value threshold was shown in [13].

10 Motion Control

Path following is achieved by a simple proportional controller which sends the motion commands "move forward", "turn left" and "turn right" as well as the maximum danger value of the upcoming path cells to the walking layer. 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 elevator reflexes for overcoming higher obstacles.

Depending on the horizontal opening angle of the stereo camera, it might be possible that the robot is not able to perceive enough information about the upcoming terrain to calculate the traversability with high certainty. Furthermore, the path could lead into a region that is currently outside the view of the robot but could be perceived if the robot would turn. In these cases, actively exploring the environment of the path is necessary. For this, the motion controller can command exploration turns as presented in [6] to the robot. An exploration turn is necessary, if the certainty value of a path cell which is in range of the cameras is lower than 1. During an exploration turn the robot turns over an angular range of 2ϵ so that the cameras cover the rover-sized circular region around the path cell being explored. Since a certainty value of 1 is hard to reach in practice, a set of rules about when exploration turns are permitted has been established:

Between two exploration turns

- the distance between the path cell to be explored and the previously explored path cell must be at least l

and one of the following conditions must hold:

- The robot must have passed a distance of at least l.
- The path cells to be explored in two subsequent exploration turns must be at an angle of at least ϵ given the current robot position is the pivot point.
- The path must have been replanned.

These rules are necessary to avoid that the robot repeats exploration turns because the certainty value does not reach 1. When the camera is mounted on a pan unit, the exploration turns can be performed by turning only the camera instead of turning the whole robot. For the hardware used in the practical tests the values were chosen to be $\epsilon = \pi/8$ and $l = 0.2\,\text{m}$.

The navigation algorithm terminates successfully when the robot reaches the specified goal point according to its estimated position. Since the robot cannot reach the goal point exactly, a tolerance area must be defined. The size of the tolerance area depends on the map resolution and the distance the robot travels within one position estimation step. In the present work, the tolerance area is a circle with a radius of 20 mm. As soon as the estimated position of the robot is inside that tolerance circle around the goal point, a stop command is sent to the walking layer and the robot stops.

11 On-line Implementation

Except for depth image computation for which a GPU implementation is used, the methods presented above have been implemented in C/C++ on a Linux 2.6 GHz computer. The navigation algorithm consists of several threads. Each sensor runs in a separate thread writing timestamped data into a buffer. IMU data is buffered at 120 Hz, leg odometry data at 10 Hz and visual odometry measurements at about 6 Hz. The pose estimator thread collects data from all sensors and fuses them according to their timestamps. The resulting pose estimate is stored in another buffer. The mapping thread receives a timestamped depth image from the stereo camera and retrieves the matching pose from the pose estimator thread. From these, the traversability map is computed. The mapping thread runs at a rate of about 1 Hz. The path planner and motion controller also run in separate threads. They retrieve the current pose estimate and calculate paths and motion commands, respectively. The motion controller sends motion commands to the walking layer at a rate of about 10 Hz.

12 Experimental Results

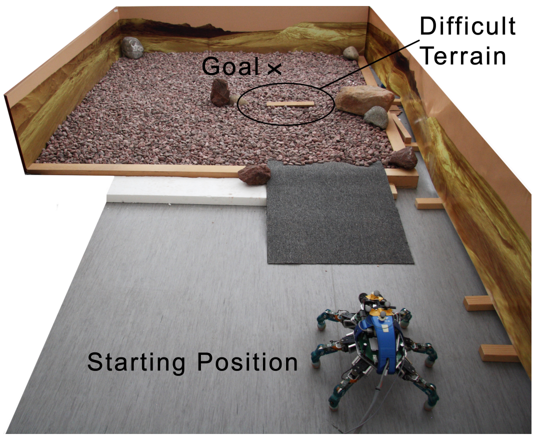


Figure 11: Test setup

For evaluating the performance of the navigation algorithm, an indoor test environment was created as shown in Fig. 11. A gentle slope led into a testbed filled with gravel. Large stones were used as untraversable obstacles. Most of the gravel area is easy to pass for the robot. Fig. 12 shows sample images of

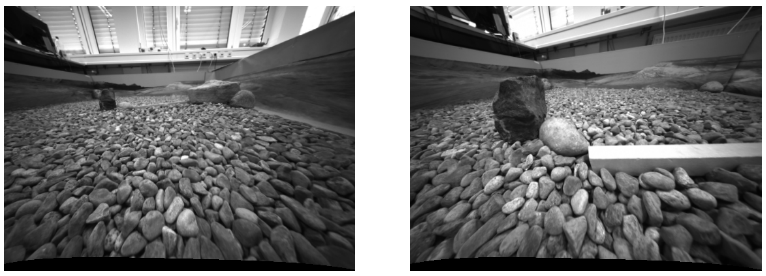


Figure 12: Sample images as viewed by the wide angle camera of the Crawler

the testbed as viewed by the stereo camera of the Crawler (ref. section 5). In a smaller region, the terrain difficulty was increased so that this area could only be passed using the biologically inspired gait with the stretch and elevator reflexes. An external tracking system was used for tracking a target body mounted on the Crawler, which provided ground truth pose measurements in comparison to the pose estimated by the robot. The goal coordinates ($x = 2.8\,\mathrm{m}$, $y = 0.3\,\mathrm{m}$) were given relative to the starting position of the robot.

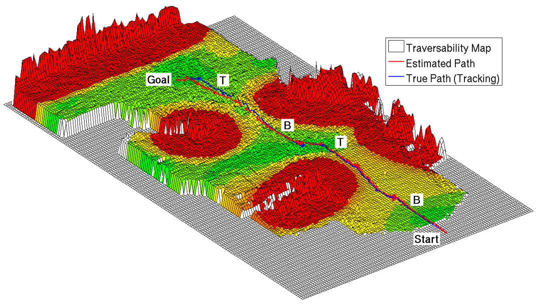
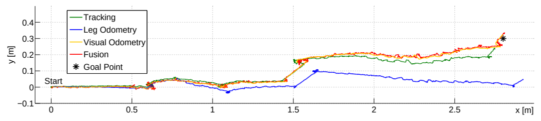
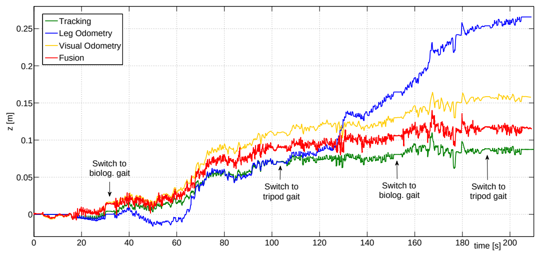


Figure 13: Traversability map and trajectories. **B**: switch to biologically inspired gait. **T**: switch to tripod gait

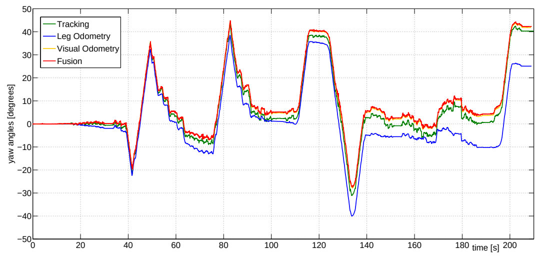
In a first experiment, the robot should reach the goal point autonomously without any external disturbances or limitations in its motion capabilities. The resulting traversability map, the trajectory obtained by the tracking system and the trajectory estimated by the robot are shown in Fig. 13. The map has a resolution of 20 mm per grid cell. The colors indicate the traversability of the cells. Red cells are untraversable, green cells are easily traversable and from green to orange the difficulty of traversing a cell increases. As can be seen, the big stones and the testbed walls were detected to be untraversable obstacles. The method for estimating the terrain traversability labels a cell as traversable only if the robot is safe when its center is located on that cell. Thus, a region of half of the robot diameter around each obstacle is also marked as untraversable. This allows the path planner to neglect the size of the robot but to only plan a path for the center of the robot. Furthermore, the traversability of the slope and the rough terrain region was estimated to be more difficult than the flat gravel areas. Hence, the robot switched to the biologically inspired gait for crossing these areas and switched back to the tripod gait as soon as it entered easier terrain.



(a) Recorded trajectories



(b) z-coordinates



(c) Yaw angles

Figure 14: Test run: No limitations in motion abilities

Fig. 14 compares the trajectories measured by visual odometry and leg odometry with the true trajectory given by the tracking system and the estimated trajectory obtained by fusing all motion measurements. The trajectory estimated by sensor data fusion is close to the visual odometry trajectory because the error of visual odometry was estimated to be very low. However, both visual odometry and leg odometry suffer from a drift in the z-coordinate as well as yaw angle errors. Due to absolute roll and pitch angle measurements, the error in the z-coordinate of the fusion trajectory is small. The yaw angle error cannot be corrected sufficiently since no absolute yaw angle measurements are available. The yaw angle plot also shows the explorations turns that were performed to gather more information about the upcoming terrain. When the robot stopped, its estimated position was $x = 2.80\,\mathrm{m}$, $y = 0.33\,\mathrm{m}$ and $z = 0.12\,\mathrm{m}$. This position is outside the given tolerance region of 20 mm because of several reasons. First, there is a small time delay between the acquisition of the position measurements and the computation of the robot pose. Second, using the tripod gait, the robot cannot stop immediately but has to finish the current step first. The true position of the robot at the goal point was $x = 2.75\,\mathrm{m}$, $y = 0.24\,\mathrm{m}$ and $z = 0.09\,\mathrm{m}$. This gives an endpoint error of 11.2 cm or 3.7% in relation to a path length of about 3 m. This error is mainly caused by the deviation of the yaw angle.

The second test run was used to demonstrate the robustness of the navigation

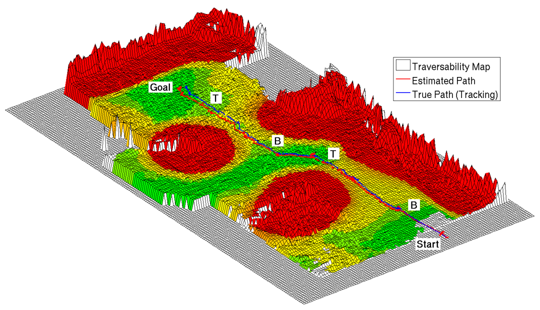


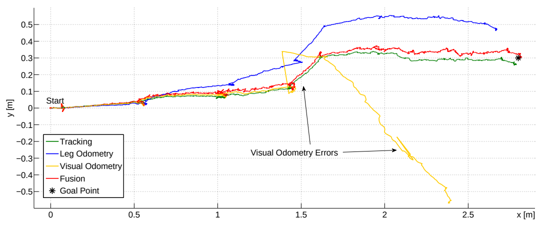
Figure 15: Traversability map after inducing visual odometry errors

algorithm against visual disturbances. While the robot was walking through the test environment, its cameras were covered several times using a sheet of paper. All other test conditions remained equal to the previous run. Fig. 16 shows the resulting trajectories. The visual odometry trajectory (ref. Fig. 16(a)) has large errors caused by covering the cameras. The leg odometry trajectory also deviates from the true path due to slip. The path estimated by fusing all motion measurements is very accurate. The visual disturbances did not affect the fusion result since visual odometry errors were estimated to be very high during these time steps and, thus, these erroneous measurements were given a very low weight in the data fusion process. This can also be seen in Fig. 16(d), because the estimated standard deviation of the position estimate increases strongly during these periods. The resulting traversability map does not show any artifacts or obvious errors (ref. Fig. 15). At the goal point, the robot estimated to be at $x = 2.81\,\mathrm{m}$, $y = 0.30\,\mathrm{m}$ and $z = 0.08\,\mathrm{m}$, while its true position was $x = 2.78\,\mathrm{m}$, $y = 0.26\,\mathrm{m}$ and $z = 0.09\,\mathrm{m}$. The overall endpoint error is 4.8 cm.

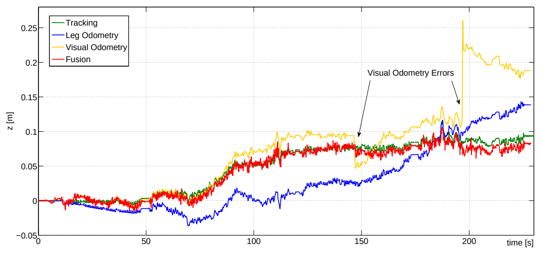
In another test run, it was simulated that the Crawler picked up a heavy load at one point of its path and had to continue to the goal point with limited motion capabilities. Fig. 17(a) shows the resulting traversability map. The robot starts moving towards the goal point as in all previous test runs. After passing the slope, the danger value threshold $d_{\max}$ was set to a low value of 0.15 to simulate that the robot is carrying a heavy load. As a result, the Crawler avoided the difficult area of the testbed and chose the longer but safer path to the goal point. The endpoint error of the estimated position at the goal point was 7.7 cm.

13 Conclusion

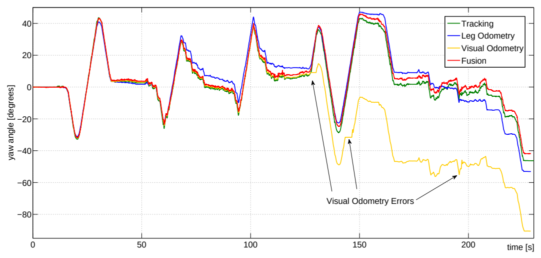
In this paper, a stereo vision based navigation algorithm for a six-legged walking robot in unknown rough terrain has been presented. From stereo images depth images are computed using the robust and accurate SGM method. A visual odometry was implemented and visual odometry errors are estimated along with the relative motion measurements. For achieving accurate and robust pose estimates of the robot, IMU data is fused with visual odometry and leg odometry using an indirect information filter. In parallel, a dense 2.5D terrain



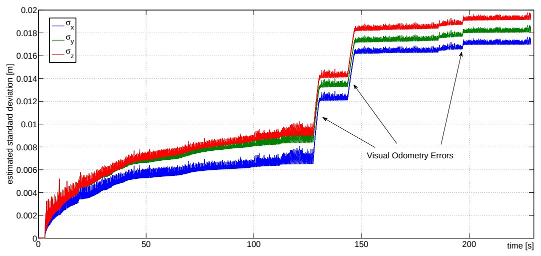
(a) Recorded trajectories



(b) z-coordinates



(c) Yaw angles

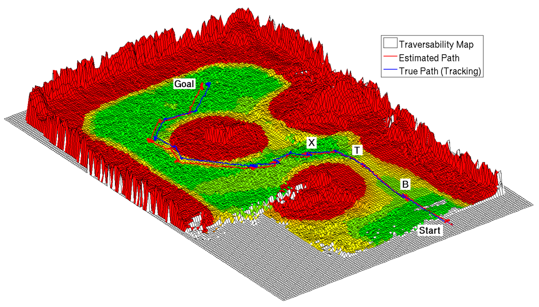


(d) Standard deviations computed from the estimation covariance matrix

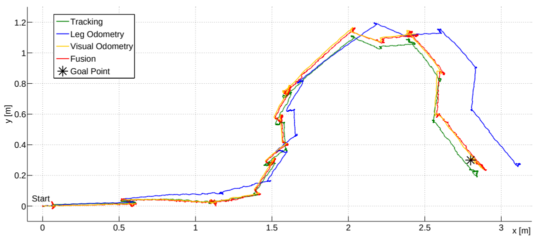
Figure 16: Test run: Visual odometry errors induced

map is created from the disparity images and the pose estimates. A plane fitting approach is used to estimate the traversability of the terrain in order to plan safe and short paths. A D* Lite path planner was implemented with a cost function that takes the terrain traversability into account and is able to adapt paths to changed motion capabilities of the robot. Motion commands are generated and sent to the walking layer along with the estimated traversability of the upcoming path cells.

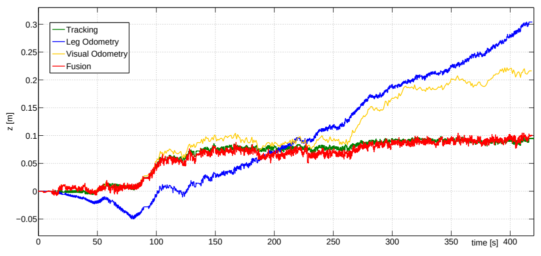
As the experimental results show, the navigation algorithm works well in small scale environments. Fusing inertial, visual odometry and leg odometry



(a) Traversability map. **X**: limitation of motion capabilities



(b) Recorded trajectories



(c) z-coordinates

Figure 17: Test run: Limitations in motion abilities

data and using the computed visual odometry errors results in robust pose estimates that allow navigation under bad visual conditions. Using SGM as stereo method allows to create dense accurate depth images in weakly textured areas. As a result, no artifacts are present in the height maps and the traversability of the terrain can be estimated accurately. The robot is able to exploit the traversability information by choosing the appropriate gait as a step towards vision induced terrain anticipation. Thus, we presented first steps towards robust, multisensor based pose estimation and terrain perception which is essential for a rough terrain specialist in a heterogeneous robotic team.

In future we want to achieve a much closer coupling of visual and tactile information. For example, in case of very bad lighting conditions the robot should switch to a tactile exploration mode and feel its way in order to proceed. For this purpose the map should allow assigning confidence values to the cell heights according to the accuracy of the information source, being tactile or visual. These confidence values should also be considered in the traversability estimation process. In addition to the geometric ground properties, tactile and texture information should be used to classify the substrate which is a valuable information for other team members and could influence the traversability estimation. Furthermore, the robot should be able to assess the traversability of distant regions by learning the visual appearance of nearby terrain with known geometrical properties. Visual terrain anticipation has to be further improved such that step height, step length and leg compliance of the robot can be adapted according to the upcoming region.

14 Acknowledgments

We would like to thank Martin Stelzer for supporting us in the on-line implementation of the navigation algorithm. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] K. S. Arun, T. S. Hunag, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, September 1987.

[2] T. Bailey, E.M. Nebot, J.K. Rosenblatt, and H.F. Durrant-Whyte. Data association for mobile robot navigation: a graph theoretic approach. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2512–2517, 2000.

[3] J. Bortz. A New Mathematical Formulation for Strapdown Inertial Navigation. *IEEE Transactions on Aerospace Electronic Systems*, 7:61–66, 1971.

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

[5] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz. Global Path Planning on Board the Mars Exploration Rovers. *Aerospace Conference, 2007 IEEE*, pages 1–11, March 2007.

[6] A. Chilian and H. Hirschmüller. Stereo Camera Based Navigation of Mobile Robots on Rough Terrain. In *IROS, International Conference on Intelligent Robots and Systems*, pages 4571–4576, 2009.

[7] A. Chilian, H. Hirschmüller, and M. Görner. Multisensor Data Fusion for Robust Pose Estimation of a Six-Legged Walking Robot. In *Proceedings of the IEEE International Conference on Intelligent Robotic Systems (IROS) 2011*, pages 2497–2504, 2011.

[8] M. Daily, J. Harris, D. Keirsey, D. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. *International Conference on Robotics and Automation, 1988. Proceedings., 1988 IEEE*, 2:718–726, April 1988.

[9] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Transactions on Robotics and Automation*, 17(5):731–747, 2002.

[10] 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, pages 228–239, Las Vegas, NV, USA, December 2008.

[11] 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, pages 134–143, Liege, Belgium, October 2009.

[12] S. B. Goldberg, M. W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. *Aerospace Conference Proceedings, 2002. IEEE*, 5:2025–2036, March 2002.

[13] M. Görner, A. Chilian, and H. Hirschmüller. Towards an Autonomous Walking Robot for Planetary Surfaces. In *Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, September 2010.

[14] 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*, pages 1525 – 1531, September 2008.

[15] 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):344–351, 2009.

[16] R. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1446, November-December 1989.

[17] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[18] H. Hirschmüller. *Stereo Vision Based Mapping and Immediate Virtual Walkthroughs*. PhD thesis, School of Computing, De Montfort University, Leicester, UK, June 2003.

[19] H. Hirschmüller. Stereo processing by semi-global matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, February 2008.

[20] 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):229–246, April-June 2002.

[21] 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*, pages 1099–1104, December 2002.

[22] 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):1582–1599, September 2009.

[23] A. Kelly. *An Intelligent Predictive Control Approach to the High-Speed Cross-Country Autonomous Navigation Problem.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 1995.

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

[25] K. Konolige, M. Agrawal, R. Bolles, C. Cowan, M. Fischler, and B. Gerkey. Outdoor mapping and navigation using stereo vision. In *Proc. of Intl. Symp. on Experimental Robotics (ISER)*, 2007.

[26] IS Kweon and T. Kanade. High-resolution terrain map from multiple sensor data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 278–292, 1992.

[27] S. L. Laubach, J. Burdick, and L. Matthies. An autonomous path planner implemented on the Rocky7 prototype microrover. In *IEEE Conference on Robotics and Automation*, pages 292–297, 1998.

[28] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[29] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, June 1987.

[30] D. P. Miller, R. S. Desai, E. Gat, R. Ivlev, and J. Loch. Reactive navigation through rough terrain: experimental results. In *Proceedings of the 1992 National Conference on Artificial Intelligence*, pages 823–828, 1992.

[31] A. H. Mishkin, J. C. Morrison, T. T. Nguyen, H. W. Stone, B. K. Cooper, and B. H. Wilcox. Experiences with operations and autonomy of the Mars Pathfinder Microrover. *Aerospace Conference Proceedings, 1998. IEEE*, 2:337–351, March 1998.

[32] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on*, 17(6):890–897, dec 2001.

[33] S.I. Roumeliotis and J.W. Burdick. Stochastic cloning: A generalized framework for processing relative state measurements. In *Proceedings of the IEEE International Conference on Robotics and Automation, 2002*, volume 2, pages 1788–1795, 2002.

[34] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, volume 2, pages 1656–1663, 1999.

[35] R. Simmons, L. Henriksen, L. Chrisman, and G. Whelan. Obstacle avoidance and safeguarding for a lunar rover. In *Proc. AlAA Forum on Advanced Developments in Space Robotics, Madison WI*, 1998.

[36] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[37] J.F. Vasconcelos, P. Oliveira, and C. Silvestre. Inertial Navigation System Aided by GPS and Selective Frequency Contents of Vector Measurements. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference (GNC2005)*, San Francisco, USA, August 2005.

[38] 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*, May 2010.

[39] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondance. In *Proceedings of the European Conference of Computer Vision*, pages 151–158, Stockholm, Sweden, May 1994.