The EU Framework Programme for Research and Innovation H2020
Research and Innovation Action

Deliverable D5.2 CENTAURO Terrain Classification

Dissemination Level: Public

Project acronym:	CENTAURO
Project full title:	Robust Mobility and Dexterous Manipulation in Disaster Response by Fullbody Telepresence in a Centaur-like Robot
Grant agreement no.:	644839
Lead beneficiary:	KTH – Kungliga Tekniska Hoegskolan
Authors:	X. Chen, F. Schilling, T. Klamt, and P. Jensfelt
Work package:	WP5 Navigation
Date of preparation:	2016-08-23
Type:	Report
Version number:	1.0

Document History

Version	Date	Author	Description
0.1	2016-07-26	XC and FS	First draft
0.2	2016-08-01	XC and FS	First complete draft
0.3	2016-08-02	PJ	Executive summary, edits and todos
0.4	2016-08-03	XC and FS	More future work, some results, and fixes
0.5	2016-08-03	PJ	Edits
0.6	2016-08-23	PJ, XC and FS	Incorporated feedback from LIU and UBO. Appended papers from UBO.
0.7	2016-08-26	PJ, XC and FS	Incorporated feedback from UBO. Modified geometric cost function.
0.8	2016-08-30	XC and FS	Adjust figures. Add UBO feedback.
0.9	2016-08-31	TK	Add UBO content.
1.0	2016-08-31	Coordinator	Submitted version.

Executive Summary

This deliverable describes our approach to terrain classification, one of the core components in the CENTAURO system.

Our terrain classification method consists of two complementary processing pipelines that assess the traversability of the local terrain based on sensor readings from camera and laser scanner. We split the task of terrain classification into geometry-based and appearance-based navigability assessment. Both processing steps are described in detail and we introduce possible hybrid approaches of combining them into a final cost map that can be used for path planning and navigation.
The first pipeline is based on geometric features of the terrain obtained from a laser scanner and outputs a 2D cost map. The traversability cost is measured from different feature metrics such as terrain roughness, height variability, and slope.
The second pipeline is based on visual features obtained from a RGB camera and outputs a 2D map of discrete terrain classes such as vegetation, asphalt, person, or vehicle. We employ a convolutional neural network approach to distinguish different terrain classes with the goal of complementing the geometric traversability assessment.
This deliverable provides preliminary qualitative results on both processing pipelines and suggests methods of combining geometric and visual features into a final cost map as future work. In addition, we suggest the possibility of using proprioceptive sensory information for self-supervised learning and ideas for human-in-the loop learning to refine our traversability assessment. Moreover we present first results of path planning.

Contents

1 Introduction

This deliverable reports on the design and realization of the terrain classification component in the CENTAURO project, in which a centaur-like robot is developed. It supports two locomotion modalities: driving and walking. Terrain classification is a core component for autonomous mobile robot navigation and path planning in real-world environments (WP5), as well as visual guidance for the human operator (WP3). Our terrain classification system addresses the project-wide objectives of robust mobility (O2) and situation awareness (O5). In this report, terrain classification is framed as the problem of assessing the traversability of discrete patches of space. We employ geometric and appearance based features in the assessment.

The most important factor in traversability assessment is the geometry of the scene, which we obtain from a laser scanner. We rely on features such as local height differences and slope to predict terrain roughness. A secondary input to the traversability assessment is the segmentation of the terrain into pre-defined classes based on visual information. The visual features will give important insights into possible wheel sinkage and slippage on soft ground and provide additional information in cases where the laser scanner cannot fully assess the traversability on its own.

Our main goal is to combine geometric and visual features of the local environment into a final traversability or cost map, which contains sufficient information for the robot to navigate and plan its motions autonomously. This map is populated with cost values, reflecting the cost to travel. Since the creation of the cost value itself is rather arbitrary, we also outline how the human operator and feedback from the robot traversing its local environment can be used to continuously refine the traversability assessment. We believe that adding human-in-the-loop learning and self-supervision can be important mechanisms to accomplish full autonomy.

The remainder of the deliverable is structured as follows. Section 2 provides related work around terrain classification, as well as a taxonomy of the field into proprioceptive, exteroceptive, and hybrid approaches. Section 3 describes our geometric and visual pipeline in detail, and suggests possible ways to combine these two into a final cost map used for path planning. Section 4 contains some implementation details of our system, especially regarding the Pluto robot platform we are currently working with. Finally, Section 5 highlights the most important milestones on our roadmap to fully autonomous path planning and navigation.

2 Related Work

Autonomous robot navigation in complex environments is a considerable challenge due to the difficulty in identifying terrain regions that can be safely traversed. The issues have been explored through different approaches. One way to divide the related work is on the basis of the sensor type. They can be characterized as geometry-based traversability analysis, appearance-based terrain classification, and hybrid approaches with multiple sensors.

2.1 Geometry-based Traversability Analysis

We use terrain traversability to estimate the capability of the robot to traverse over a terrain region. A 3D laser sensor is commonly used to scan the terrain surface. There are two ways to represent the traversability using 3D laser data: statistical processing and frequency processing. Statistical processing captures the terrain traversability by defining a cost function that aggregates the geometric features as well as robot constraints. Langer et al. [19], Andersen et al. [2] and Joho et al. [14] calculate the traversability score using features such as elevation, slope, roughness, obstacle presence, and data point accuracy. Kubota et al. [18] derive the traversability probability by estimating the roll, pitch, and height criteria of the robot. Ishigami et al.[13] formulate a dynamic mobility index which combines stability as well as wheel slippage, time duration, and energy consumption on a given terrain.

Frequency processing is widely used in terrain classification based on proprioceptive sensors, which measure the response when a vehicle traverses a local terrain. Features such as velocity, vibrations, and tactile sensors (bumbers) are commonly used in frequency based terrain classification. Brooks and Iagnemma [5] present a classification based on processing vibration signals recorded by a microphone. Warda and Iagnemmaa [32], Collins and Coyle [15] and DuPont et al. [9] demonstrate the effectiveness of terrain classification based on vibration signatures and explain how the terrain elevation profile and the wheel vibrations are related. We can apply the same approach by simulating laser inputs or elevation maps as proprioceptive sensor signals. Lu et al. [21] identify terrain features using Fast Fourier Transform (FFT) on laser data and Wang et al. [31] define an algorithm that detects terrain types using a power spectrum density (PSD) feature matrix.

Geometry-based approaches assume the ground that provides the support for driving or walking can be observed. However, for soft terrain like snow and grass, the real ground is hidden under a soft cover. In such cases, the terrain surface that sensors such as stereo vision or laser scanners perceive can be very different from the real ground that the robot experiences. We attempt to solve such problems by combining geometry and vision based traversability assessment as described in Section 3.

2.1.1 Appearance-based Terrain Classification

We refer to appearance-based terrain classification as a type of traversability assessment using only RGB image data without additional depth information. Since the literature on terrain classification with state-of-the-art visual models is rather sparse, we adduce related work on the more general semantic segmentation task, where the goal is to classify each pixel in the input image into a predefined class. In the field of semantic segmentation, convolutional neural networks show excellent performance when discriminating between visual object classes. We hypothesize that these types of models are equally suitable for the terrain classification task, and ultimately to assess traversability.

Convolutional neural networks for semantic segmentation are visual models that take inspiration from the architecture of convolutional autoencoders. The encoder part of the network processes the input image by applying repeated convolution and downsampling operations. Downsampling the input creates a so-called bottleneck that projects the high-dimensional representation onto a lower dimensional subspace. The decoder part of the network reverses the operations in order to create a reconstruction of the original input from a compressed representation. For the specific task of semantic segmentation, the model is trained on per-pixel class labels instead of using the original input as targets.

Fully convolutional networks (FCN) [20] generalize convolutional networks for image recognition to handle inputs of arbitrary size by replacing fully connected with convolutional layers. The key insight is that 1×1 convolutions perform the same operation as fully connected layers without putting input size constraints on the network. Moreover, Long et al. [20] show that fully convolutional networks can make dense per-pixel predictions for tasks such as semantic segmentation and achieve state-of-the-art results. The key novelty of their segmentation architecture is the combination of coarse information from higher layers with fine, lower-layer information by summing the predictions at different downsampling stages of the network and upsampling the generated label maps accordingly. However, unlike other modern segmentation architectures, the FCN model is lacking a decoder part.

SegNet [3] is a fully convolutional model architecture that can be created from arbitrary networks used for single-label image classification. By mirroring each convolutional and downsampling layer, the network creates a final prediction map that matches the spatial size of the input. However, rather than reversing the downsampling operations in the encoder by naive bilinear or nearest neighbor upsampling in the decoder, the model reuses the indices of the maximum activations to upsample the learned representation. This has the effect of forcing the spatial structure of the input onto the final segmentation map, while also making robust and accurate class predictions. An extension of this architecture called Bayesian SegNet [16] is able to model the pixel-wise per-class uncertainty of an assigned label in the output map. The uncertainty measure is generated by taking stochastic dropout samples of the predictions and computing their mean (class label) and variance (class uncertainty).

DeconvNet [22] also belongs to the fully convolutional family of architectures and features a slightly different up-sampling approach in the decoder part of the network. The main innovation of the proposed architecture is the use of fractionally strided convolutions rather than regular convolutions. Fractionally strided convolutions are often inadvisedly called deconvolutions due to the work of Zeiler et al. [33] but they are merely the transpose (gradient) of the convolution operation. Where regular convolutions map from a patch of the same size as the filter to an output pixel, deconvolutions project one pixel onto a patch of the size of the filter. The authors suggest using strided convolutions and deconvolutions in their respective parts of the network in order to circumvent the saving of pooling indices.

Lastly, Razavian et al. [23] show that features learned by convolutional neural networks provide representations that are generic enough to enable transfer learning, i.e. a model trained on a suitable dataset provides a good baseline for any visual recognition task. The features of one model can be extracted and fine-tuned with additional training data to perform well on a task of interest. In case of image classification and segmentation, a task is usually defined entirely by its set of class labels.

2.2 Hybrid Approaches

Since individual sensors have their own strengths and limitations, many authors consider different combinations of features, sensors and fusion methods. The mainstream in hybrid methodologies is the combination of geometry-based and appearance-based features. Lu et al. [21] fuse the FFT features extracted from laser data and image features such as contrast, correlation, energy, and homogeneity using a probabilistic neural network. Hadsell et al. [11] provide an online learning framework that uses geometric features as sample labels with image features extracted from a convolutional neural network model.

Kim et al. [17] consider not only cameras and laser scanners, but also feedback from GPS, bumpers, and motor encoders as ground truth labels to combine the sensor data. Halatci et al. [12] use an accelerometer and implement two classifiers, namely maximum likelihood estimation (MLE) and a support vector machine (SVM), to process color, texture and range features from the image. A Bayesian fusion method is then applied to combine the image features with the ones from the accelerator. Brooks and Iagnemma [5] use a microphone instead of an IMU to monitor the vibrations between the robot and the terrain and train an SVM to classify the image data.

3 Method

In this section, we derive terrain characteristics from geometric and visual inputs. The geometry-based analysis takes the terrain elevation point cloud generated from the raw laser scanner data as input and provides straight-forward features of the terrain structure which can be applied directly to assess the terrain traversability. Similarly, the vision-based analysis takes the camera images as input and classifies the terrain type into a set of terrain classes. The terrain traversability score is then inferred from the discrete terrain class. Firstly, we describe the geometric terrain feature extraction in Section 3.1, then we explain the visual terrain segmentation architecture in Section 3.2. Some approaches of fusing these results and transfer the traversability value into an actual cost for a specific robot in navigation task are described in Section 5 as future work.

3.1 Geometry-based Traversability Analysis

As described in the introduction, features from both spatial domain and frequency domain can be extracted from the terrain elevation map. Features from the spatial domain such as slope and roughness summarize the relationship of a point with respect to its surrounding points. Features from the frequency domain such as FFT and PSD can predict the robot's response to entire terrain segments. In our approach, only spatial features are used to represent the terrain traversability since the frequency features require a large window size to generate a reasonable result. Therefore, it is hard to generate a high resolution cost map with frequency-based features. The process of geometry-based traversability analysis includes:

(1) pointcloud pre-processing,
(2) terrain feature extraction and
(3) 2D cost map projection.

The input of the process is the point cloud generated by laser scanner and the output is a 2D cost map that contains a traversability score for each map cell.

The laser scanner used in our approach is the Velodyne VLP-16. It scans the environment using 16 sweeping beams which cover the field of view between ± 15 degrees vertically and 360 degrees horizontally. In order to obtain denser point clouds, we register the scans using a Normal Distributions Transform (NDT) presented in [24]. An example of a full Velodyne scan registration is shown in Figure 1. The focus of this deliverable is terrain classification which can be evaluated using this local scan alignment. In the integrated system, we plan to use our terrain classification methods in conjunction with the mapping framework being developed in Task 5.1. The corresponding results [26, 8] are enclosed at the end of this deliverable.

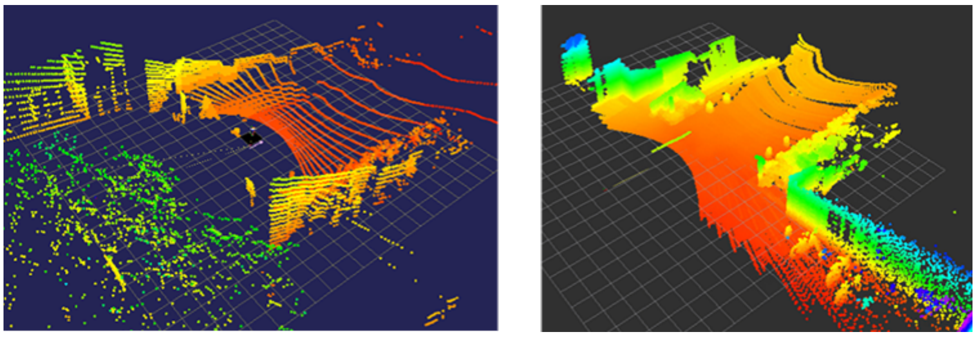


Figure 1: Left: one scan from the Velodyne VLP-16. Right: registered pointcloud from 30 scans.

3.1.1 Pointcloud Pre-processing

From Figure 1 we can see almost one third of the points belong to big obstacles like the wall, which can never be accessed by the robot. In order to speed up the computation we remove the points that belong to obstacles before we calculate the terrain features for each point. Apart from the obstacle points, we define the ground points to be the points where we can place the robot without colliding with any obstacle. For every point in the point cloud, we investigate whether it belongs to the ground or an obstacle. We define a cube surrounding a point to represent the minimum space required by the robot body to perform actions. The CENTAURO robot can lower its body to avoid obstacles above the head and use its legs to step over obstacles with low height. In Figure 2, $h_{\text{head\_min}}$ denotes the minimum height of the robot body and $h_{\text{leg\_max}}$ the maximum height the robot can step on, respectively. The red points between $h_{\text{head\_min}}$ and $h_{\text{head\_min}}$ are the points which cannot be accessed by the robot. A point is classified as a ground point only if there is no points exist between $h_{\text{head\_min}}$ and $h_{\text{leg\_max}}$. An example of the pre-processing result is shown in Figure 3.

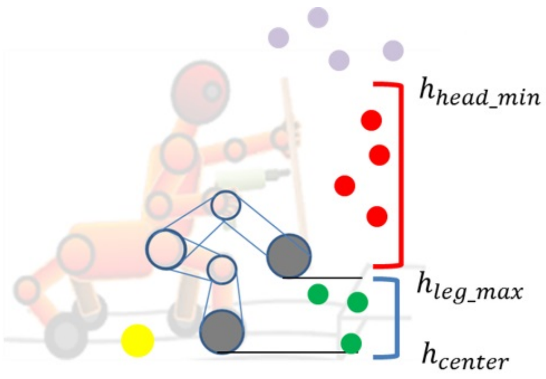


Figure 2: Obstacle points. The robot is placed on the yellow point; the grey points above the robot can be avoided by shrinking down the body; the green points on the bottom can be stepped over using the legs and the red points cannot be accessed by the robot.

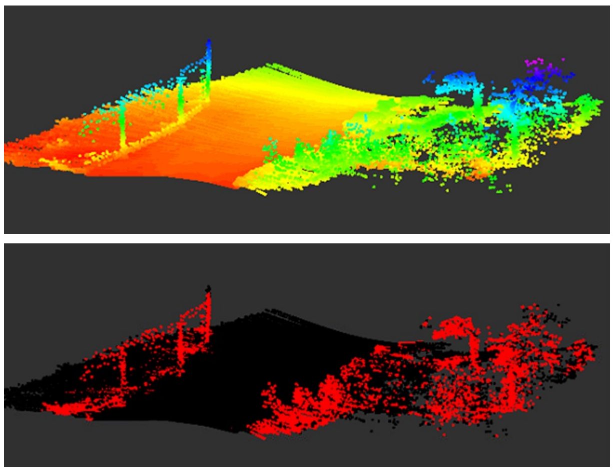


Figure 3: Point cloud pre-processing. The top image is the registered point cloud where the color represent the height. The bottom image is the result of pre-processing, the black points are the ground points and the red points are the obstacles.

3.1.2 Terrain Feature Extraction

Methods for road quality evaluation have been studied extensively. Various standards at national and international level such as ISO 2631 [1], the international roughness index and the riding index [25] are available to represent and evaluate the road quality and riding comfort for the passengers sitting in the vehicle. Human perception of the riding quality are highly dependent on the vibrations of the vehicle. ISO 2631 describes an evaluation method on how vibrations affect health, comfort, perception, and motion sickness. The basic evaluation measured in the standard is the weighted acceleration root-mean-square value in a certain time domain.

We select the standard ISO 2631 as one of our terrain feature and slightly modify the standard to use the vertical acceleration root-mean-square value in a spatial domain instead of time domain. The vertical accelerations are the product of the underlying terrain profile. To simplify the problem, we assume there is no friction and deformation between terrain and wheel. Then the vertical acceleration can be calculated purely based on the gravity slope angle of the terrain under the robot, as shown in Figure 4.

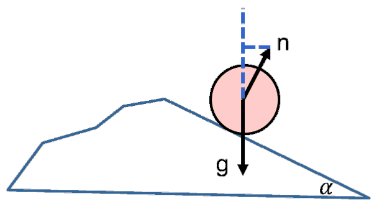


Figure 4: The vertical acceleration is only related to the gravity terrain slope angle a.

For every ground point we define a cube of the same size as the robot's wheel footprint. Four features are computed using all the points inside the cube, which are namely its mean slope, maximum height difference, height variance, and root-mean-square vertical acceleration. These four features describe the terrain characteristics and are independent from the robot model. Both slope and height difference are related to the pose stability when the robot is standing on the point of interest; the height variance can be interpreted as terrain roughness and the root-mean-square of the vertical acceleration captures the robot's expected vibration when driving over a specific point.

3.1.3 2D Traversability Cost Map Projection

We define a 2D structured map that contains one traversability value for every pixel to be the interface between terrain features and the navigation component. The 2D map is a convenient structure for data processing and visualization, and it is compatible with most of the off-the-shelf navigation algorithms. We first assign a traversability value to each ground point and then project the points to the horizontal map surface. The traversability value can be designed in different ways using the combination of our four geometric features and the actual robot model. We compute the cost C as

$$C(x,y) = k_1 \cdot \triangle H(x,y) + k_2 \cdot HV(x,y) + k_3 * S(x,y) + k_4 * A(x,y) \tag{1}$$

which is the weighted sum of the maximum height difference $\triangle H(x,y)$, the height variance $HV(x,y)$, the slope $S(x,y)$ and the vertical acceleration $A(x,y)$ at (x,y). We assign a larger weight to the root-mean-square acceleration feature $A(x,y)$ since our platform is very sensitive to body vibrations.

Based on the map resolution, several ground points can be projected onto the same pixel and the pixel traversability value is the mean of the point traversability value that projected on the pixel. A special case has to be considered when the robot is standing in front of a table or a bridge where it can both go over and under the table/bridge. If the points of the ground surface and the points of the table/bridge surface are visible at the same time, they are also projected onto the same pixel. In this case, we cannot fuse them together because they belong to different surfaces. We check the maximum open space between points to identify multiple access paths. If the space is big enough for the robot to go though, we separate them into two groups. We define another traversability map to represent the second accessible surface and the traversability value is the mean traversability computed by the points belonging to each group. Figure 5 shows the process of detecting multiple surfaces.

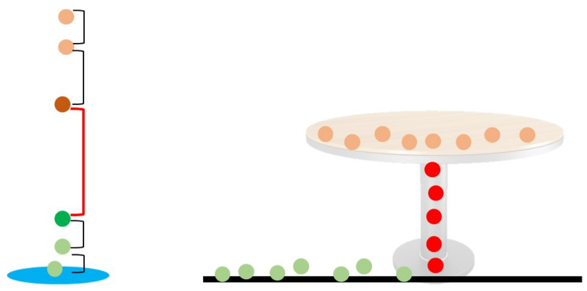


Figure 5: Multiple accessibility detection. The left image shows the case when multiple points are projected to the same pixel. The blue circle represents the pixel, the small circles represent the points. The blocks are the size of open spaces between points and the red block is the maximum open space. The points above and below the maximum open space are sepaated into two group. The right image shows an example of a table, the green and yellow points are two surface groups and the red points are obstacles.

Two examples of the cost generation is shown in Figure 6. After we project the point cloud to the cost map we apply a morphological interpolation algorithm to fill the holes that are not covered by the sensor data. The color of the interpolated cost map is normalized for better display.

The traversability cost extracted from the laser point cloud measures the quality of the terrain, but when driving on soft terrain, the elevation profile measured by the laser scanner may not be the same profile the robot actually encounters. The laser scan can easily be fooled by the properties of the terrain surface, especially for highly deformable terrain types such as long grass. They can thus be marked as obstacles although they are perfectly drivable. The deformation of wheel and terrain, weight of the robot, terrain sinkage rate and slippage rate will all affect the interaction between the wheels and the terrain surface. Bing et al. [4], Smith and Peng [29], and Smith [30] attempt to simulate the interaction between the vehicle and soft terrain but all models mentioned in these papers require additional information that a laser cannot provide. To compensate for this problem, we apply another terrain classification system based on image features to recognize the surface material for each terrain type.

3.2 Vision-based Traversability Analysis

In addition to the information on terrain geometry obtained from the laser scanner, we employ a semantic segmentation architecture to process image data from the RGB camera. The goal of this subsystem is to augment the geometry-based terrain information with semantic labels to

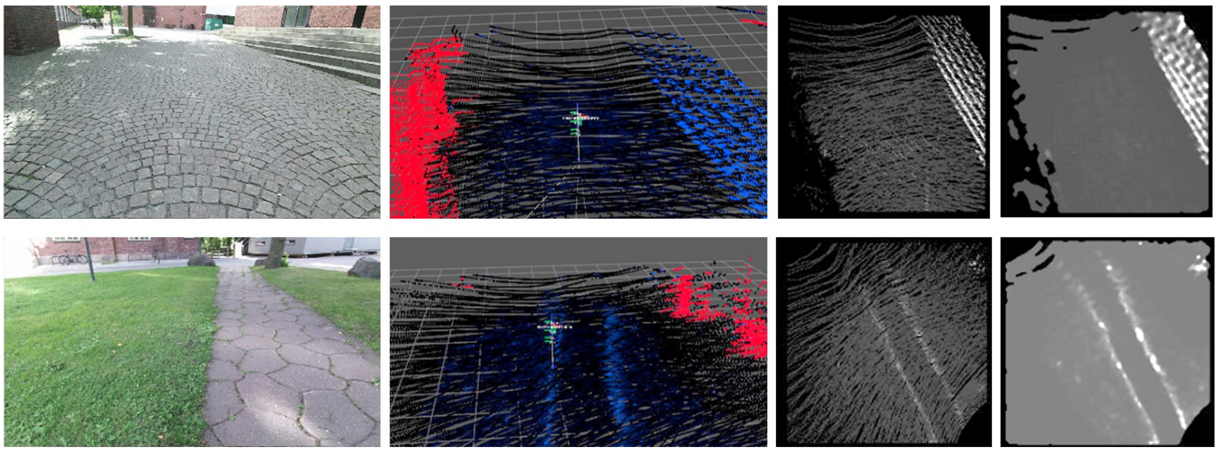


Figure 6: Cost map generation. The first column is the RGB image. The second column is the registered point cloud with cost value. The red points are the obstacles detected during the pre-processing step. The cost value is colored in blue, where darker blue denotes lower cost and lighter blue denotes higher cost. The third column is the cost map projected by the point cloud and the fourth column is the cost map after interpolation.

gain a better understanding of the scene in general, and specifically to assess its navigability. A semantic understanding of the local terrain is vital in cases where the laser scanner is not able to fully assess its traversability. Two possible scenarios can be identified and defined by the following two examples.

In the first example, the robot is facing a large patch of flat ground and based on laser scanner data alone, the system would suggest to drive over it. If the alleged flat ground were water, a system that does not assess the appearance of the terrain and issues the drive command will inadvertently put the robot into immediate danger.

In the second example, the robot is surrounded by patches of large height variability and deems the terrain non-navigable solely based on the traversability cost from the laser scanner. However, the environment is made up of (deformable) knee-high grass and bushes, which the robot could handle by walking or even driving. Again, a system that does not take terrain appearance into account might not find a path with sufficiently low risk and thus get stuck in this type of environment.

To address these types of issues, we define a model architecture that is inspired by state-of-the-art convolutional networks for semantic segmentation. In order to perform fast inference on our model, we relax our requirements from per-pixel to superpixel accuracy, which we believe to be sufficient for navigability assessment and path-planning purposes. The model is defined in Section 3.2.1, along with general system requirements and possible limitations.

Since the creation of a large task-specific segmentation dataset is arguably infeasible, we make use of transfer learning techniques. Section 3.2.2 provides information about the dataset for our baseline model, as well as requirements for task-specific training data for fine-tuning.

3.2.1 Segmentation Architecture

We define a segmentation architecture that can provide both accurate labeling while preserving the spatial structure of the input. Although variable input size is not a strict requirement, we opt for a fully convolutional architecture without the loss of representational power. In order to reduce computational complexity, as well as speeding up training and inference, we employ multiple stages of downsampling between stacks of convolutional layers to reduce the spatial size of the input. The encoder part of our network resembles the popular VGG-16 [28] architecture, whereas the decoder mirrors the encoder with deconvolutions and unpooling layers as shown in Figure 7.

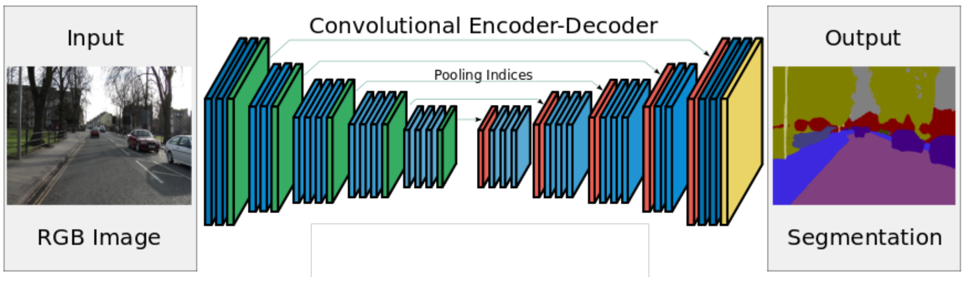


Figure 7: Our networks resembles a convolutional encoder-decoder segmentation architecture. The blue layers denote convolution and deconvolution operations of the encoder and decoder, respectively. The green layers denote spatial max-pooling, whereas the red layers denote spatial unpooling operations according to the saved pooling indices. The final yellow layer denotes a pixel-wise softmax operation. Figure modified from [3].

When downsampling, we save the indices of maximum activations in the encoder part of the network in order to perform upsampling that relates the spatial location of the predictions back to the original input. A visualization of both the unpooling and deconvolution operation can be obtained in Figure 8.

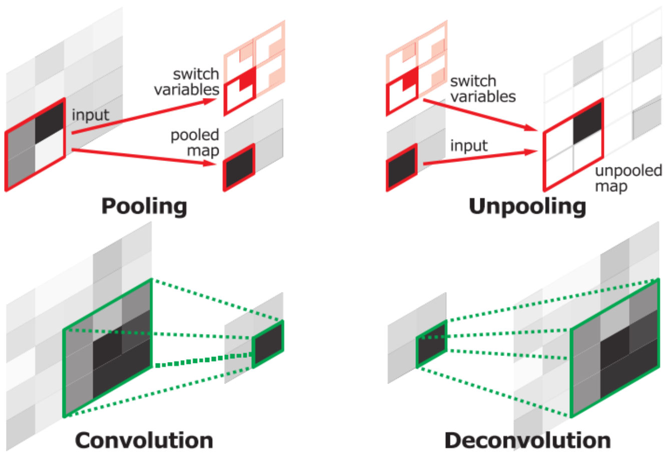


Figure 8: Visualization of unpooling and deconvolution operations. During the pooling operation, the indices of the maximum activations are saved in switch variables. Those switch variables are used to create a sparse activation map during upsampling. The deconvolution – or fractionally strided convolution – operation is the transpose (gradient) of the convolution operation. Figure taken from [22].

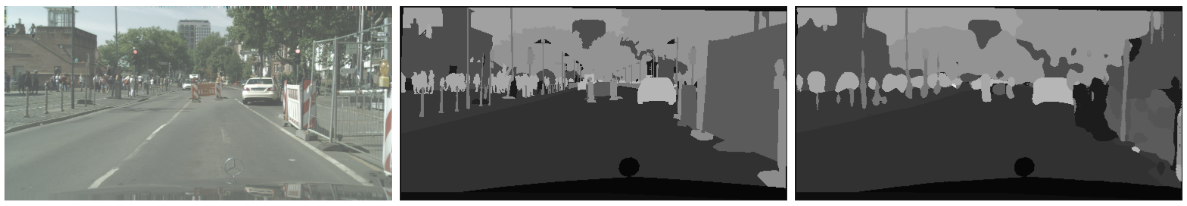
3.2.2 Segmentation Datasets

Training deep learning architecture such as convolutional neural networks require a large amount of labeled training data. Popular benchmark datasets for semantic segmentation are, for instance, PASCAL VOC [10] and MS COCO [6]. The main shortcoming of these datasets is their focus on objects in the foreground of the scene, whereas the interest in terrain classification is to assess the navigability of the entire field of view, including the background.

We aim to train our model on a dataset that resembles the terrain classes of the final task to a large extent, the choice fell on the CityScapes dataset [7]. CityScapes is a semantic segmentation dataset geared towards autonomous driving in urban scenes. Interestingly, the group labels of the CityScapes dataset already provide a good approximation for our classes of interest. The interesting classes of the dataset are namely: hard ground (road and sidewalk), soft ground (grass, soil, and sand), vegetation (trees, bushes, shrubbery, etc.), vehicle (car, truck, motorcycle, etc.), construction (buildings, road signs, etc.), and humans. At this point we deemed it to require too much resources to create our own dataset for training before we have tested the full system and identified the bottlenecks in performance.

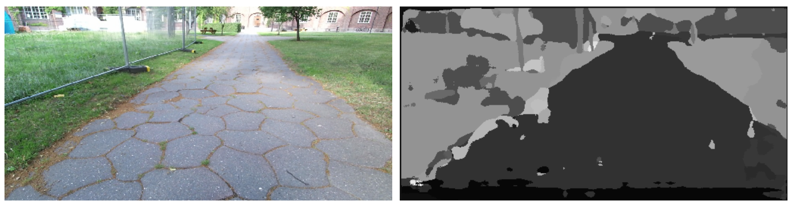
3.2.3 Preliminary Segmentation Results

Preliminary results using the CityScapes test image is shown in Figure 9 and the result using an image acquired on our test platform in our test environment is shown in Figure 10. Further improvements can be done by fine-tuning the model using data taken in our test environment. Note that this fine-tuning requires considerably less data than the dataset used for training the baseline model.



(a) Input image (b) Ground truth (c) Output segmentation

Figure 9: Preliminary segmentation results on the CityScapes dataset.



(a) Input image (b) Output segmentation

Figure 10: Preliminary segmentation results on an image taken with Pluto.

3.3 Robots Shape Specific Cost

The ground contact surface of the CENTAURO robot consist of four individual pairs of wheels. In contrast to many other robots this allows to take obstacles, such as stones or poles, between its legs when driving. In disaster scenarios, many situations occur in which the robot has to manoeuvre in narrow environments such as corridors which also might include smaller obstacles. Thus, a path planning algorithm which considers the robots footprint increases traversability. The footprint of a centaur-like robot is included in the following cost computation which differs between wheel cost and base cost.

As introduced in Sec. 3.1.3, wheel specific traversability cost are computed as weighted sum of terrain features:

$$C_W(x,y) = k_1 \cdot \triangle H_{coarse}(x,y) + k_2 \cdot HV(x,y) + k_3 * S(x,y) + k_4 * A(x,y). \quad (2)$$

All features are inflated by the robots wheel size which allows to consider each wheel as a single point and to discard the wheel orientation. The height difference $\triangle H(x,y)$ is computed from a coarse neighborhood (about 0.5m in diameter). This leads to higher cost in areas around obstacles.

There are situations in which wheel specific cost are not sufficient to describe the traversability of the whole robot. e.g. wheel cost are hardly able to distinguish between a low and a tall pole. The robot would be able to take the former one between its legs while a tall pole could damage the robots base. The same goes for obstacles over the robots head. The additional base cost

$$C_B(x,y,\theta) = k_5 \cdot \max(H_{\max,uB} - H_B, 0) + k_6 \cdot \max(H_H - H_{\min,oB}, 0) + k_7 \cdot \triangle H_{max,W} \quad (3)$$

are therefore introduced. If the maximum terrain height under the robots base $H_{\max,uB}$ is higher than the robots clearance H_B, additional cost are generated. Those describe the action of lifting the robots body. The same goes for the case when the robots head height H_H is higher than the lowest obstacle height over the robots base $H_{\min,oB}$ and the robots base has to be lowered. In addition, a term is added which induces cost when the robots wheels stand on different terrain heights which is an indicator for difficult situations. This is either the case when traversing slopes or when moving over steps or gravel.

Finally, wheel specific cost at the respective positions and base specific cost are combined to a cost value

$$C(x,y,\theta) = 1 + k_8 \cdot max(C_{Wi}) + k_9 \cdot \sum_{i=1}^{4} C_W(x_{w_i}, y_{w_i}) + k_{10} \cdot C_B(x,y,\theta) \quad (4)$$

which describes the entire three-dimensional state (x,y,θ) of the robot. The basic cost is 1 which is increased when terrain difficulties occur. There are different possibilities to combine the four wheel specific cost C_{Wi}. One option is to sum up all cost values. However, this would give a state in which all four wheels stand on slightly difficult terrain the same cost as a state in which one wheel stands on very difficult terrain while the other three wheels see no difficulties. This solution is not satisfying because one would prefer the former situation to the later one. It is also not satisfying to only choose the maximum of the four wheel cost. In this case, four wheels on difficult terrain would be assigned the same cost as one wheel in difficult terrain and three wheels in easy terrain. A good solution is the weighted combination of those two methods because it is able consider all four wheels while paying additional attention to the maximum occurring difficulty. Furthermore, the base cost C_B is added. Whenever one of the cost summands describes that a state is not traversable, this information is directly transferred to the overall cost $C(x,y,\theta)$ which is assigned infinite. The same goes for the case that one of the summands describes that a state is not known. The overall cost for this state are also set to unknown and the state is, hence, not considered during path planning. The cost computation, as presented above, is tested in simulation which is described in Sec. 4.3. Note that these costs are used for the planning of driving paths. Additional maneuvers such as wheel position changes, center of mass shifting and walking will be developed for D5.3 and D5.4.

First work has already been done on identifying feasible step positions. The developed method is wheel specific and does not consider the whole robots body yet. From a starting position (x_s, y_s) which is traversable for a wheel by driving $(C_W(x_s, y_s) < inf)$ but which has a neighbour cell that is not traversable the following algorithm searches for feasible steps. First those stepping directions are identified which require to step over an untraversable cell. Within this range of possible stepping directions and within a given maximum step length all traversable cells are identified as possible stepping targets $(x_{t,i}, y_{t,i})$. For each feasible step, the maximum height difference

$$\triangle\, H_{max,step} = max\Big[|H_{max,step} - H(x_s, y_s)|, |H_{max,step} - H(x_{t,i}, y_{t,i})|\Big] \quad (5)$$

is computed. It describes the maximum height difference between the highest cell over which to step and either the start cell or the target cell. This value is used to compute the stepping cost

$$C_S(x_s, y_s, x_{t_i}, y_{t,i}) = k_{11} \cdot \|(x_s, y_s) - (x_{t_i}, y_{t,i})\|_2 \cdot \triangle H_{max,step} + k_{12} \cdot C_W(x_{t,i}, y_{t,i}). \quad (6)$$

$\triangle H_{max,step}$ is multiplied by the step length $\|(x_s, y_s) - (x_{t_i}, y_{t,i})\|_2$. Consequently, this term represents the necessary leg movement. Longer or higher steps lead to higher cost. Furthermore, the wheel cost of the target cell $C_W(x_{t,i}, y_{t,i})$ is added. By doing this, steps to easy regions are preferred over steps onto difficult terrain. Stepping cost are also evaluated in Sec. 4.3.

4 Implementation

We implement and test the cost function (1) introduced in Section 3.1.3 seperatly. The cost function (1) is tested using an ATRV robot (Figure 11). It has four powered wheels connected to the body and uses skid steering. It features an onboard computer, an IMU, a Kinect 2 RGBD camera looking forward and a Velodyne VLP-16 mounted on top. The Kinect and Velodyne are tilted 15 degrees down with respect to the horizontal plane in order to capture as much of the ground as possible.

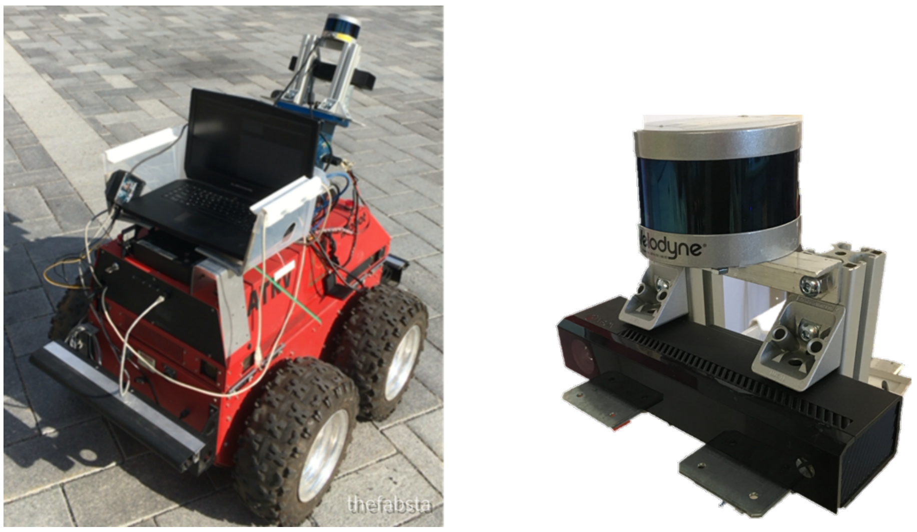


Figure 11: Left: The robot used in the evaluation, a proxy for the CENTAURO robot system under development. Right: The sensor rig used in the experiments.

For the Pluto platform, whenever there is a sharp turn or it is driving on shaky terrain, the performance of the NDT registration along the vertical direction deteriorates significantly. In the worst cases, the error along the vertical direction is off by more than half a meter, such that the reconstructed elevation map and the traversability cost no longer represents the real terrain structure at all. Therefore, instead of computing the traversability cost from the registered point cloud, we compute it directly from the single laser scan before registration. The traversability cost calculated using single scans are reasonable, but it only represents the partial features of the real terrain. Since we only consider the cost value along the direction of laser scans, we cannot capture the roughness features along other directions. In the future we will improve the registration performance to overcome this problem.

4.1 Preliminary Experimental Results

Preliminary evaluation and tests using ATRV with cost function (1) have been performed on different terrain types including asphalt, short grass, different types of stones and soil on the KTH campus. The big stone terrain contains stone patches with flat surface in the middle and the small stone terrain contains small round cabal stones. Terrain types are shown in Figure 12.

Figures 13 and 14 show the traversability cost of the five terrain types mentioned above and one with mixed grass and asphalt. For each terrain type, we scan four-meter-long terrain segments using the middle laser beam of the Velodyne. The resolution along the direction that the laser beams rotate is $0.03m$ per point. The elevation value is therefore the height of the laser points. Figure 13 shows the cost for relatively flat terrain and Figure 13 shows the cost for rough terrain. The cost value is high when there is a height difference or a continuous slope change.

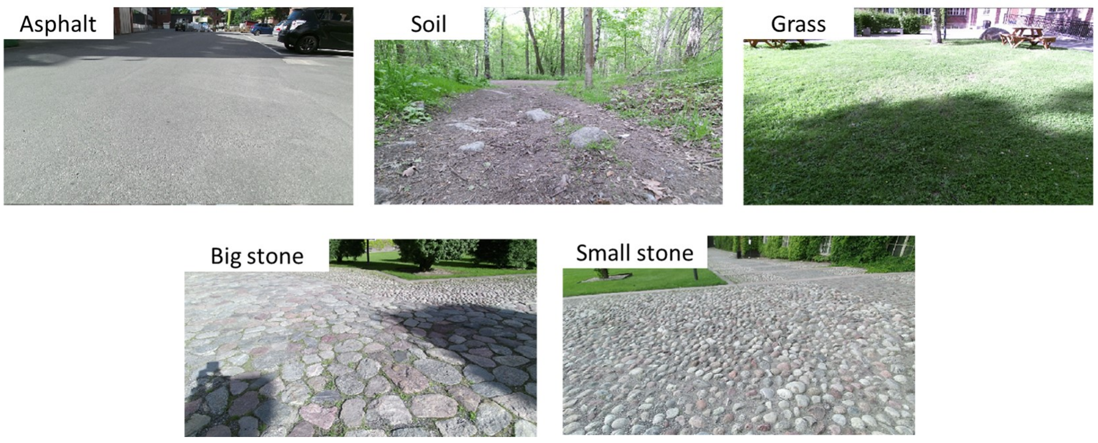


Figure 12: Different terrain types encountered in our preliminary evaluation.

On average, the big stone terrain has a lower cost value than the small stone terrain, since the slope changes between individual patches is smaller.

The cost of grass depends largely on the grass length and density, the real ground under the grass cannot be captured by the laser scanner. One can observe that in Figures 13 and 14 that grass shows a similar cost to the big stone terrain, but in reality it is much easier to drive on grass than on big stones. To account for this, we can incorporate the segmentation from the vision pipeline to decrease the geometric cost for the grass class, for instance.

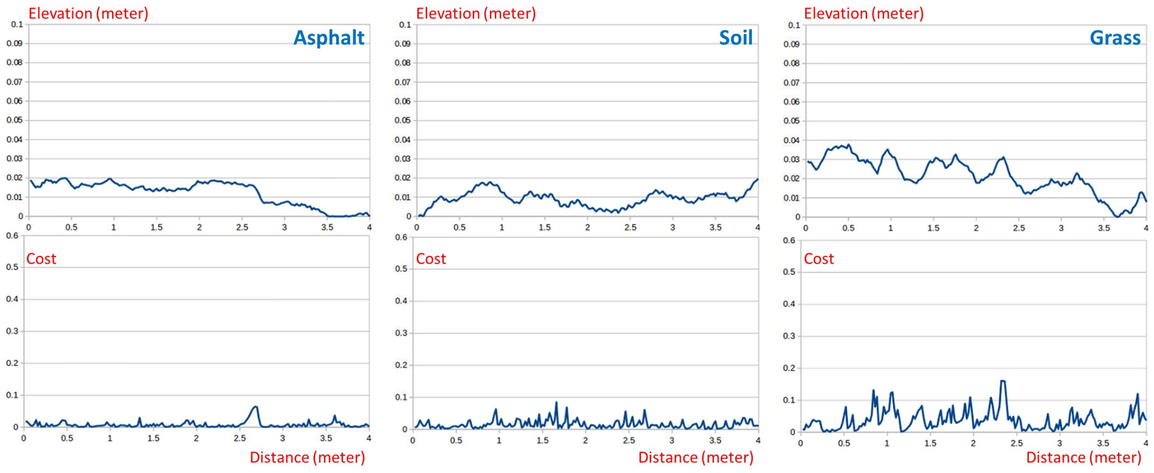


Figure 13: Elevation segment and cost value of asphalt, soil, and grass. The vertical axis of the top image is the elevation of the terrain segments, the vertical axis of the bottom image is the cost value, and the horizontal axis is the distance from the laser scanner.

4.2 Integration of Traversability Map with ROS Navigation Stack

The output geometric traversability cost map is formatted as a ROS `costmap2D` and is linked to the ROS navigation stack with an A^* path planner plugin. An example of a sensing-driving-updating loop is shown in Figure 15, where the robot autonomously navigates from position A to C without any prior knowledge of the environment. From left to right, the first column contains the images taken by the Kinect at each point; the second column contains the local traversability result calculated from laser scan, and the third column contains the updated ROS

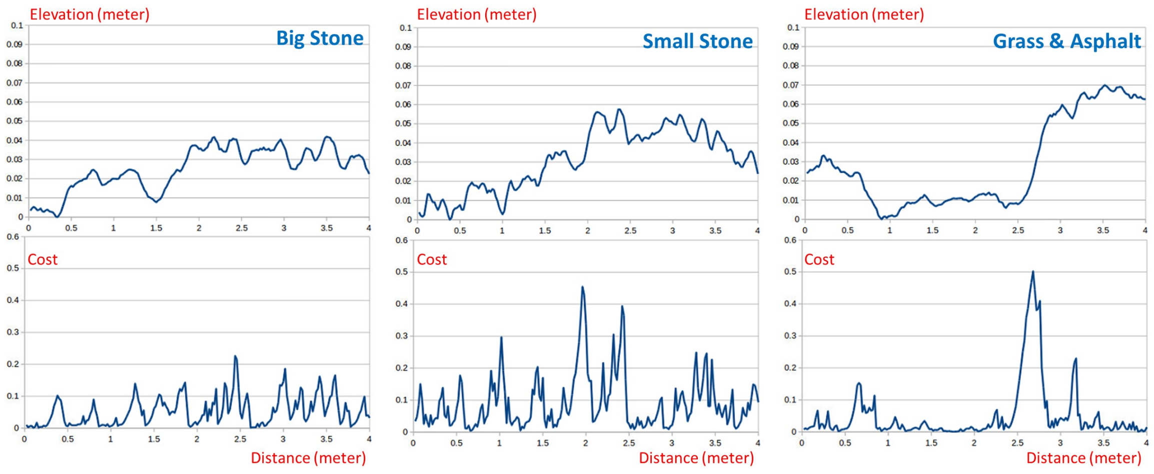


Figure 14: Elevation segment and cost value for big stones, small stones, and a mixture of grass and asphalt. The grass is on the side with higher elevation and the asphalt is in the middle with lower elevation. The vertical axis of the top image is the elevation of the terrain segments, the vertical axis of the bottom image is the cost value, and the horizontal axis is the distance from the laser scanner.

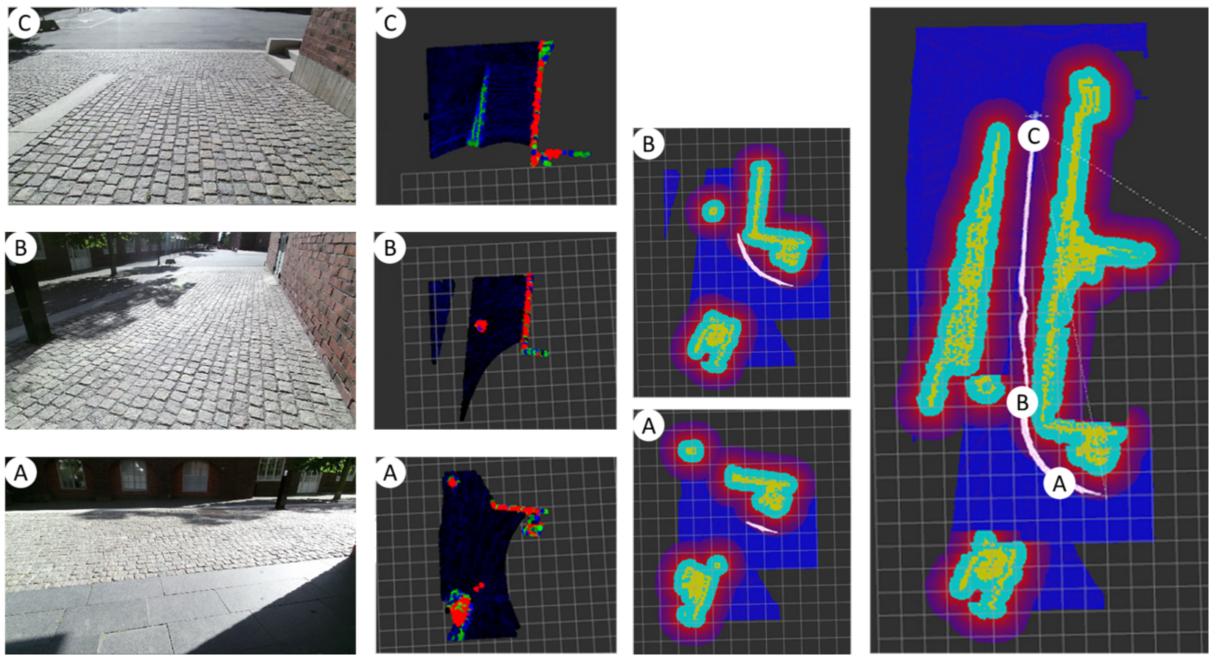


Figure 15: Global cost map and local classification result captured from spot A, B and C. From left to right: the images taken by the Kinect, local laser traversability, and updated ROS cost map. Obstacles are shown in yellow and drivable areas in blue. Light blue and red areas are spatial inflations of the obstacle areas and provide a safety distance.

cost map after we reach point A and B. The image on the right side is the cost map after we reach the destination C. The path is recomputed whenever an update in the cost map occurs.

4.3 Evaluation of Robot Shape Specific Cost

The robot shape specific cost described in Sec. 3.3 cannot be tested with the ATRV robot because its shape is too different to the shape of the CENTAURO robot. Wheel specific cost computation would only bring small advantages. Instead, the Momaro robot [27] as shown in Fig. 16 is used for this evaluation. It has four legs with steerable wheel pairs at the end of each leg. It thus requires footprint specific path planning to exploit the advantages of this robot shape. A model of this robot is used in the Gazebo simulation environment to evaluate the cost function at this state of development. Evaluation will be transferred to the real robot for further evaluation.

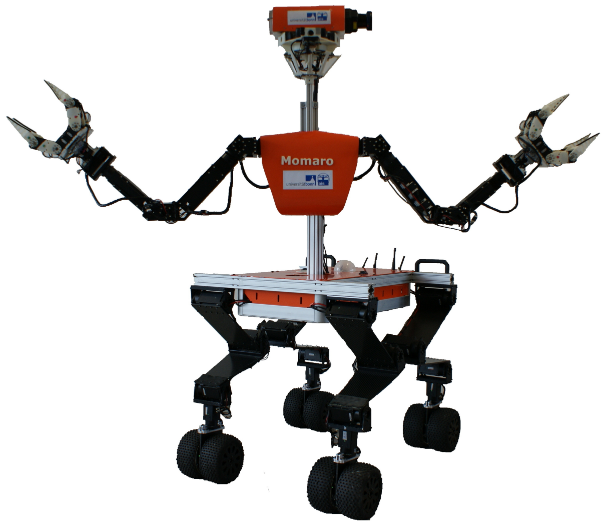


Figure 16: The mobile manipulation robot Momaro.

The cost function is tested in a scenario with two poles of different heights and a ramp as can be seen in Fig. 17. The final pose cost is computed by summing up wheel cost for each individual wheel and base cost. It can be seen that wheel cost induce an area of higher cost around obstacles which can be seen as a safety distance area which is only entered when necessary. Constant slope on the ramp leads to constant wheel cost which are higher than on flat terrain. In the shown scenario, the left pole can be taken between the robots legs while the right pole is too high and would require base lifting. Consequently, base cost occur for the right pole while the left pole stays base cost free. Base cost also identify the slope on the ramp through a height difference between the different wheels. These cost also identify that the edge of the ramp is quite dangerous and should be avoided. Pose cost show the cost for the robot to traverse a pose which consists of position and orientation. The robots footprint can be easy recognized in this visualization.

An A* graph search algorithm is utilized to find cost-optimal paths on these cost representations. The considered state space (x, y, θ) is three-dimensional, i.e. it contains the robot heading orientation θ. As expected, it is possible to drive over the short pole by taking it between the robots legs, as shown in Fig. 18. In more complicated manoeuvres, the exact avoidance of obstacles can be observed.

In addition to the path planning under consideration of the robots shape, the identification of feasible steps and computation of respective cost is evaluated. This is done on a wheel cost map because step start positions are those map cells which have non-traversable neighbours, regarding wheel cost. Fig. 19 shows an exemplary start position and respective feasible steps. It can be seen that both step length and target cell wheel cost influence the step cost.

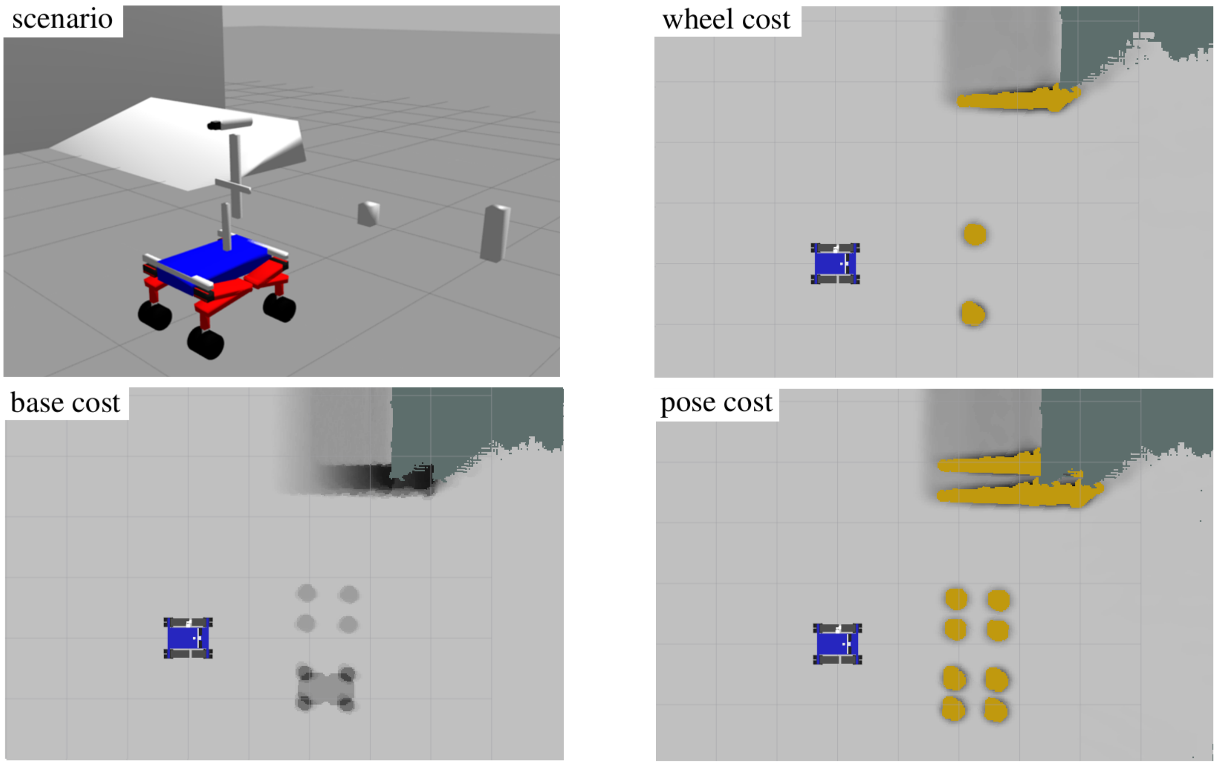


Figure 17: Simulation of robot shape specific cost computation. The darker a map cell, the higher its cost. Untraversable map cells are marked yellow. Unknown map cells are marked dark green. Top left: In the test scenario the robot sees a ramp and two poles of different heights. Top right: wheel cost of the scenario. Bottom left: base cost. Note that base cost are 3-dimensional (x, y, θ). The shown cost are computed for the shown robot orientation. Bottom right: wheel and base cost are combined to pose cost. They describe the traversability of the terrain for the center of the robots base. Wheel cost are computed for each wheel at its respective position. As described before the image shows cost for all states with the shown robot orientation.

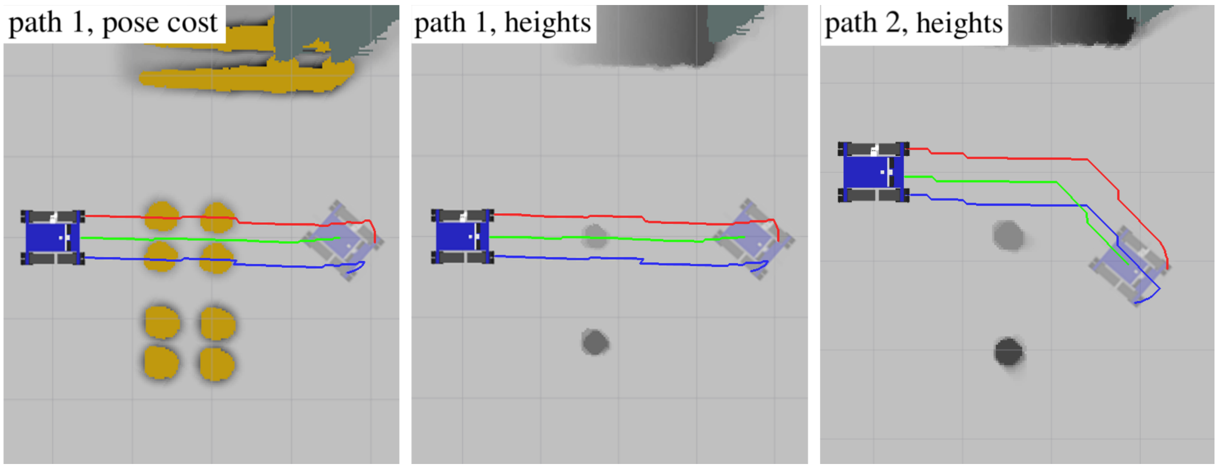


Figure 18: Path planning on pose cost. The robot center path is green, the path of the front left wheel is red and the path of the front right wheel is blue. Left: Path are planned on pose cost. Those allow driving over the smaller pole. Center: The same path is shown on the respective height map. Right: A different scenario in which the robot manoeuvres around the pole.

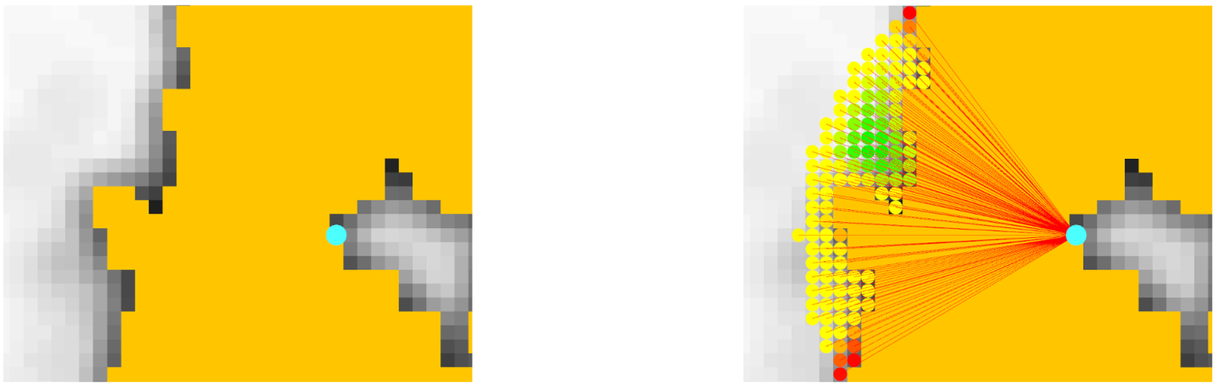


Figure 19: Step evaluation. Map cells represent wheel cost. Yellow cells are not traversable by driving. Left: Steps start at positions which have a nontraversable neighbour. Right: feasible steps with respective cost. Steps which end in green circles are cheap, steps ending in red circles are expensive.

5 Future Work

5.1 Combination of Appearance and Geometry-based Features

As mentioned before, for highly deformable terrain like tall grass and snow, the actual ground is hidden while the visible ground can be of arbitrary shape and consistency. One example of a misclassification of geometric traversability is shown in Figure 20. While it was possible to safely traverse the bush on the left side of the image, the geometry-based classification predicted ”very difficult to drive, need to climb” (green points) and ”obstacle” (red points), because the surface of the vegetation was very cluttered according to the laser scan. The laser points only see the apparent geometry structure of the terrain but the terrain type classification using vision can provide information from another perspective. We can generate a more reliable result by fusing the geometric and visual cost maps together.

Initially, we plan to use a simple weight-based algorithm to combine the results. For every terrain class, we define a default traversability cost. The final traversability cost is the weighted sum of the geometric cost and the terrain class default cost. For terrain classes such as hard ground, construction, sky and other background object we give more weight to the geometric terrain assessment; for terrain classes such as soft ground, vegetation, vehicle and human we give more weight to the visual cost. In the future, we aim to replace the simple weight-based algorithm to a self-supervised fusion model that automatically adjusts the weights in order to adapt to new types of environments.

5.2 Self-supervised Learning based on Proprioceptive Feedback

In traditional supervised learning frameworks, a human provides labeled training examples for each class. In a self-supervised learning framework, another classification algorithm is responsible for providing labels to each training sample. There are two key elements of the self-supervised learning algorithm: a mechanism to autonomously collect and label training data as the robot interacts with its environment, and an online method to learn the concept of traversability.

We plan to divide the traversability value to certain levels based on the quality the robot experiences when driving over specific terrain types. The quality level will be used as the label attached to each terrain patch. We will develop a self-supervised labeling algorithm that

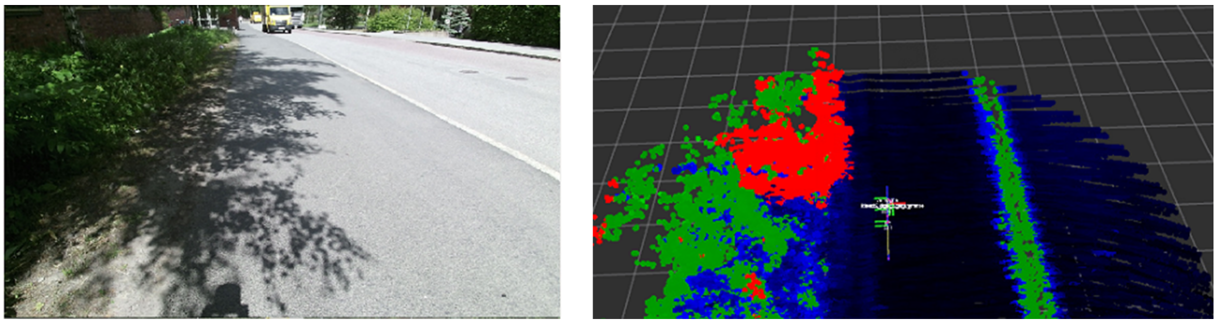


Figure 20: Example of a misclassification from the geometric traversability assessment. Left: Kinect image of a scene contains half vegetation and half road. Right: geometry traversability result.

associates the proprioceptive sensor feedback (e.g. from the IMU) with the quality level of each terrain patch. Data samples of various terrain types with label and features extracted from above classifiers can be automatically collected. Then, using the classifier described above as feature extractor, we will design a neural network that learns the relationship between features and quality level in an online fashion.

5.3 Human-in-the-loop Learning

In situations where the terrain segmentation or feedback labeling fail, for example due to extreme lighting conditions, similar adverse factors or sensor errors, we envision a human-in-the-loop learning system. Specifically, we aim to incorporate the knowledge of the operator to refine misclassified regions of the segmentation output and the mislabeled feedback when training the fusion model. The aspect of human supervision enables us to feed the corrected segmentations back into the network and thus providing new ground truth labels.

We envision an interactive system that adapts to new situations by incorporating the operator's knowledge into the learning algorithm. For the semantic segmentation task, the output is a pixel-wise map of class labels. When the segmentation proposal is different from the opinion of the operator, the operator can access the image, relabel the misclassified region and add it to the database to retrain the classifier. For the feedback labeling, the output is displayed on the local map the robot has just traversed. The operator can access the map data, adjust the result of each terrain patch and fine-tune the model.

References

[1] Mechanical vibration and shock – evaluation of human exposure to whole-body vibration. 1997.

[2] Jens Christian Andersen, Morten Rufus Blas, Ole Avn, Nils A Andersen, and Mogens Blanke. Traversable terrain classification for outdoor autonomous robots using single 2D laser scans. *Integrated Computer-Aided Engineering*, 13:223–232, 2006.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[4] Zhou Bing, Bi Tianle, and Li Weiping. Ride comfort research of-off road vehicle based on soft terrain. 2010.

[5] Christopher A. Brooks and Karl Iagnemma. Self-supervised terrain classification for planetary surface exploration rovers. *Journal of Field Robotics*, 29, 2012.

[6] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] David Droeschel, Max Schwarz, and Sven Behnke. Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner. *Submitted to Robotics and Autonomous Systems*, 2016. under review.

[9] Edmond M. DuPont, Jr. Emmanuel G. Collins, Eric J. Coyle, and Rodney G. Roberts. Terrain classification using vibration sensors: theory and methods. *In New Research on Mobile Robotics*, 2008.

[10] M. Everingham, Van Gool L., C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/, 2012.

[11] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[12] Ibrahim Halatci, Christopher A Brooks, and Karl Iagnemma. Terrain classification and classifier fusion for planetary exploration rovers. In *2007 IEEE Aerospace Conference*, pages 1–11. IEEE, 2007.

[13] Genya Ishigami, Keiji Nagatani, and Kazuya Yoshida. Path planning and evaluation for planetary rovers based on dynamic mobility index. In *IEEE International Conference on Intelligent Robots and Systems*, pages 601–606, 2011.

[14] Dominik Joho, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. Autonomous Exploration for 3D Map Learning. *Autonome Mobile Systeme 2007*, pages 22–28, 2007.

[15] Emmanuel G. Collins Jr. and Eric J. Coyle. Vibration-based terrain classification using surface profile input frequency responses. 2008.

[16] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian SegNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv:1511.02680v1 [cs.CV]*, 2015.

[17] Dongshin Kim, Jie Sun, Sang Min Oh, James M Rehg, and Aaron F Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 518–525. IEEE, 2006.

[18] T. Kubota, Y. Kuroda, Y. Kunii, and T. Yoshimitsu. Path planning for newly developed microrover. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 4, pages 3710–3715, 2001.

[19] D. Langer, J. Rosenblatt, and M. Hebert. A Behavior-Based System for Off-Road Navigation. *IEEE Transactions on Robotics and Automation*, 10(6):776–783, 1994.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, November 2015.

[21] Liang Lu, Camilo Ordonez, Emmanuel G. Collins, Jr., and Edmond M. DuPont. Terrain surface classification for autonomous ground vehicles using a 2d laser stripe-based structured light sensor. 2009.

[22] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[23] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[24] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J Lilienthal. Normal distributions transform monte-carlo localization (ndt-mcl). In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 382–389. IEEE, 2013.

[25] MW Sayers. The little book of profiling. 1998.

[26] Max Schwarz, Marius Beul, David Droeschel, Sebastian Schüller, Arul Selvam Periyasamy, Christian Lenz, Michael Schreiber, and Sven Behnke. Supervised autonomy for exploration and mobile manipulation in rough terrain with a centaur-like robot. *Frontiers in Robotics and AI*, 3, 2016.

[27] Max Schwarz, Tobias Rodehutskors, Michael Schreiber, and Sven Behnke. Hybrid driving-stepping locomotion with the wheeled-legged robot Momaro. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5589–5595, 2016.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[29] William Smith and Huei Peng. Modeling of wheel-soil interaction over rough terrain using the discrete element method. *Journal of Terramechanics*, 2013.

[30] William Clarke Smith. Modeling of wheel-soil interaction for small ground vehicles operating on granular soil. 2014.

[31] Shifeng Wang, Sarath Kodagoda, Zhan Wang, and Gamini Dissanayake. Multiple-sensor based approach for road terrain classification. 2013.

[32] Chris C. Warda and Karl Iagnemmaa. Speed-independent vibration-based terrain classification for passenger vehicles. *International Journal of Vehicle Mechanics and Mobility*, 47, 2009.

[33] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535, June 2010.

A Appendix

The following articles are enclosed:

- David Droeschel, Max Schwarz, and Sven Behnke. Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laserscanner. Submitted to Robotics and Autonomous Systems, 2016. under review. [8]
- Max Schwarz, Marius Beul, David Droeschel, Sebastian Schüller, Arul Selvam Periyasamy, Christian Lenz, Michael Schreiber, and Sven Behnke. Supervised autonomy for exploration and mobile manipulation in rough terrain with a centaur-like robot. Frontiers in Robotics and AI, 3, 2016. [26]

Continuous Mapping and Localization for Autonomous Navigation in Rough Terrain using a 3D Laser Scanner

David Droeschel, Max Schwarz, Sven Behnke

Autonomous Intelligent Systems Group, Computer Science Institute VI
University of Bonn, Friedrich-Ebert-Allee 144, 53113 Bonn, Germany

Abstract

For autonomous navigation in difficult terrain, such as degraded environments in disaster response scenarios, robots are required to create a map of an unknown environment and to localize within this map. In this paper, we describe our approach to simultaneous localization and mapping that is based on the measurements of a 3D laser-range finder. We aggregate laser-range measurements by registering sparse 3D scans with a local multiresolution surfel map that has high resolution in the vicinity of the robot and coarser resolutions with increasing distance, which corresponds well to measurement density and accuracy of our sensor. By modeling measurements by surface elements, our approach allows for efficient and accurate registration and leverages online mapping and localization. The incrementally built local dense 3D maps of nearby key poses are registered against each other. Graph optimization yields a globally consistent dense 3D map of the environment. Continuous registration of local maps with the global map allows for tracking the 6D robot pose in real time. We assess the drivability of the terrain by analyzing height differences in an allocentric height map and plan cost-optimal paths. The system has been successfully demonstrated during the DARPA Robotics Challenge and the DLR SpaceBot Camp. In experiments, we evaluate accuracy and efficiency of our approach.

Keywords:
Mapping, Localization, Rough Terrain

Email addresses: `droeschel@ais.uni-bonn.de` (David Droeschel), `max.schwarz@uni-bonn.de` (Max Schwarz), `behnke@cs.uni-bonn.de` (Sven Behnke)

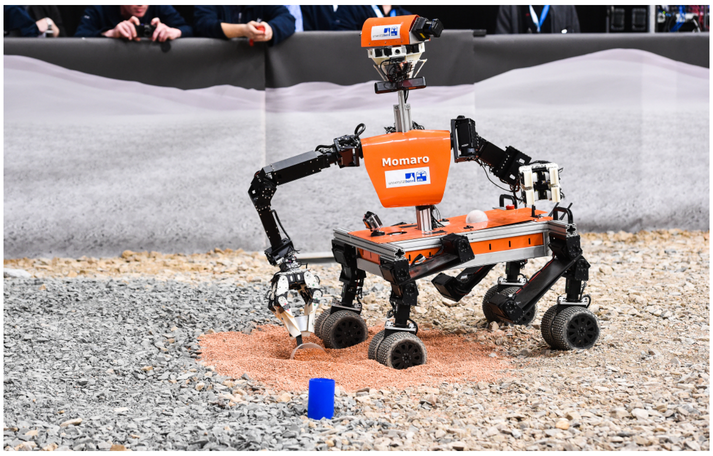


Figure 1: The mobile manipulation robot Momaro taking a soil sample during the DLR SpaceBot Camp. Without intervention of an operator, the robot learned a map of the previously unknown environment, localized within this map, and autonomously navigated to the goal pose that has been specified in a coarse environment map beforehand.

1. Introduction

In order to enable robot systems to enter areas inaccessible to humans, e.g., in disaster scenarios or for planetary exploration, autonomous navigation is key. It necessitates the capability to simultaneously build maps of unknown environments and to localize within. These environments can be cluttered or degraded and pose a challenge for perception algorithms. To enable autonomous navigation, the perceived map of the environment has to be accurate enough to allow for analyzing whether a particular region is drivable or not. Besides that, the efficiency of the perception system is important since the operation in these environments often requires online mapping and localization in real time with limited onboard computers.

In this paper we describe our system for mapping and localization on our mobile manipulation robot Momaro. The robot has been developed according to the requirements of the DARPA Robotics Challenge[1] (DRC). The goal of the DRC was to foster research for robots that are able assist humans in responding to catastrophic situations, such as the nuclear disaster at Fukushima in 2011. Being teleoperated over a limited network connection,

[1]`http://www.theroboticschallenge.org/`

the robots had to solve eight tasks relevant to disaster response. While the DRC showed the potential of robots for tasks found in disaster response scenarios, it also showed that fully autonomous navigation and manipulation in unstructured environments—also due to the lack of applicable perception methods—is still beyond the state of the art.

In contrast to the DRC, where robots could be teleoperated for navigation, the DLR SpaceBot Camp 2015 focused on autonomy. Based on a coarse map of the environment, the robot had to explore a previously unknown planetary-like environment and to perform a set of mobile manipulation tasks. Figure 1 shows our robot Momaro taking a soil sample. By means of a 3D continuously rotating laser scanner, Momaro acquires range measurements in all spatial directions. The 3D scans of the environment are aggregated in a robot-centric local multiresolution map. The 6D sensor motion is estimated by registering the 3D scan to the map using our efficient surfel-based registration method [1]. In order to obtain an allocentric map of the environment—and to localize in it—individual local maps are aligned to each other using the same surfel-based registration method. A pose graph that connects the maps of neighboring key poses is constructed and optimized globally. By localizing the robot with respect to the optimized pose graph, we gain an accurate estimate also in larger environments with big loops, where filter-based approaches would obtain an inaccurate estimate. The graph-based formulation allows to globally minimize accumulated errors, resulting in an accurate map of the environment and localization pose.

The remainder of the paper describes our laser perception system that was used during the DRC Finals and the DLR SpaceBot Camp. During the DRC, only the local mapping components where used to build a egocentric map of the robot's direct vicinity. This map was used by the manipulation operator when planning motions and to correct odometry drift of the robot, when aligning to a previously acquired local map. This part of the system is described in Section 4 and Section 5 and is based on our previous work in [1]. Apart from the local mapping, our allocentric mapping component [2] was used to allow for fully autonomous navigation during DLR SpaceBot Camp and is described in Section 6.

In this article, we present a complete system for continuous mapping and localization, fully integrated in our navigation system and extensively tested. Building a fully integrated system with the given requirements led to the following advances over our previous work:

1. We extended our local multiresolution map to address for dynamics in the environment. By efficiently maintaining occupancy information we increase the quality of the maps and the robustness of the registration.
2. We extended our allocentric mapping system to allow for fully continuous mapping and localization during mission, without the necessity to map the environment beforehand or to stop for acquiring new 3D scans and to process them.
3. In the evaluation section, we show data acquired during the DARPA Robotics Challenge Finals and the DLR SpaceBot Camp 2015.

Our mapping pipeline is published open-source[2], making it available to other researchers in order to facilitate developing robotic applications, contributing to the system, and for comparing and reproducing results.

2. Related Work

For mobile ground robots that operate in cluttered and degraded environments, 3D laser scanners are the preferred sensor for mapping and localization. They provide accurate distance measurements, are almost independent on lighting conditions, and have a large field-of-view.

Mapping with 3D laser scanners has been investigated by many groups [3, 4, 5, 6]. A common research topic in laser-based simultaneous localization and mapping (SLAM) is efficiency and scalability, i.e. maintaining high run-time performance and low memory consumption. To gain both memory and runtime efficiency, we build local multiresolution surfel grid maps with a high resolution close to the sensor and a coarser resolution farther away. Local multiresolution corresponds well to the sensor measurement characteristics. Measurements are aggregated in grid cells and summarized in surface elements (surfels) that are used for registration. Our registration method matches 3D scans on all resolutions concurrently, utilizing the finest common resolution available between both maps, which also makes registration efficient. In previous own work [7, 8], we used this concept within an octree voxel representation.

For aligning newly acquired 3D scans with the so far aggregated map, we use our surfel-based registration method [1]. In contrast to many methods for point set registration—mostly based on the Iterative Closest Point (ICP)

[2]`https://github.com/AIS-Bonn/mrs_laser_map`

algorithm [9]—our method recovers the transformation between two points sets through probabilistic assignments of surfels. Probabilistic methods for point set registration are becoming more and more popular recently and show promising results [10, 11, 12].

Hornung et al. [13] implement a multiresolution map based on octrees (OctoMap). Ryde et al. [14] use voxel lists for efficient neighbor queries. Both of these approaches consider mapping in 3D with a voxel being the smallest map element. The 3D-NDT [15] discretizes point clouds in 3D grids and aligns Gaussian statistics within grid cells to perform scan registration.

Belter et al. [16] also propose to use local grid maps with different resolutions. In contrast to our approach, different map resolutions are used for different sensors, resulting in an uniform grid map for each sensor. Herbert et al. propose elevation maps [17], extending 2D grid maps by adding a height for every grid cell. While elevation maps only model a single surface, multi-level surface maps [18] store multiple heights in each grid cell, allowing to model environments with more than on surface, such as bridges for example. Pfaff et al. [19] propose a method for detecting loop closures in elevation maps. Frankhauser et al. [20] use local elevation maps and handle drift by propagating uncertainties of the robot pose through the map.

Our mapping system has been successfully applied on micro aerial vehicles (MAV) to allow for fully autonomous navigation [21]. In contrast to this work, we do not rely on accurate visual odometry anymore, but on imprecise wheel odometry in combination with measurements from an inertial measurement unit (IMU). Compared to other mapping approaches, we efficiently build robot-centric maps that are locally consistent—with constant computation and memory requirements. We construct an allocentric graph of local maps from different view poses, resulting in a sparse pose graph that can be optimized efficiently. Compared to the mapping system used in our previous work [22], the system presented here is more efficient and maps the environment in a continuous manner—without the requirement to stop for acquisition and processing of new 3D scans. Besides that, the robustness of the localization improved since the current system aligns dense local maps to the allocentric map, in contrast to single 2D scans. While many methods assume the robot to stand still during 3D scan acquisition, some approaches also integrate scan lines of a continuously rotating laser scanner into 3D maps while the robot is moving [23, 24, 25, 26, 27].

Path planning for navigating in 3D indoor environments with flat floors is well-studied [28, 29]. For navigation on non-flat terrain, several approaches

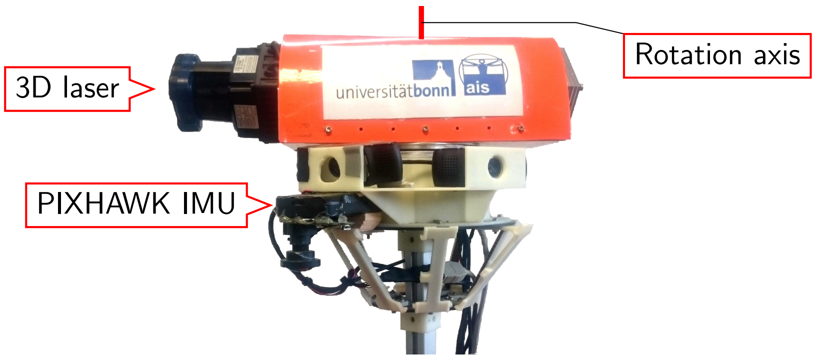


Figure 2: Momaro's sensor head. The Hokuyo laser scanner is rotated by an actuator around the red axis to allow for an omnidirectional field-of-view. The IMU is used to compensate for motion during scan acquisition and for estimating the attitude.

generate 2D cost maps from sensor readings and plan paths in these [30, 31, 32, 33]. Rusu et al. [34] model 3D maps by a set of convex polygons and adapt existing 2D planners to operate in 3D terrain. Chhaniyara et al. [35] and Papadakis [36] compiled surveys on judging traversability of terrain and avoiding obstacles with robots. In many environments, color or texture do not provide sufficient traversability information, so 3D geometry is needed. We present an integrated system for efficient laser-based 3D SLAM, traversability analysis, and cost-optimal path planning.

3. System Overview

Momaro is equipped with four articulated compliant legs that end in pairs of directly driven, steerable wheels. The combination of legs and steerable wheels allows for omnidirectional driving and stepping locomotion. To perform a wide range of manipulation tasks [37], Momaro is equipped with an anthropomorphic upper body with two 7 degrees of freedom manipulators that end in dexterous grippers.

Momaro's main sensor for environmental perception is a continuously rotating laser scanner on its sensor head (see Figure 2). It consists of a Hokuyo UTM-30LX-EW 2D laser scanner which is rotated around the vertical axis by a Robotis Dynamixel MX-64 servo actuator to gain a 3D FoV. Hence, the sensor can measure in all directions, except for a cylindrical blind spot around the vertical axis centered on the robot. The 2D LRF is electrically connected by a slip ring, allowing for continuous rotation of the sensor.

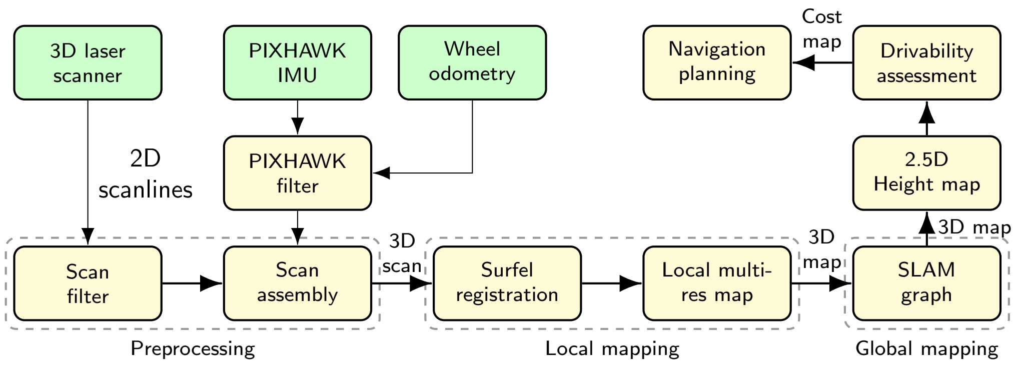


Figure 3: Overview of our mapping, localization and navigation system. The measurements are processed in preprocessing steps described in Section 5.1. The resulting 3D point cloud is used to estimate the transformation between the current scan and the map (Section 5). Registered scans are stored in a local multiresolution map (Section 4). Keyframe views of local maps are registered against each other in a SLAM graph (Section 6). A 2.5D height map is used to assess drivability. A standard 2D grid-based approach is used for planning (Section 7).

The Hokuyo 2D laser scanner has an apex angle of 270° and an angular resolution of 0.25°, resulting in 1080 distance measurements per 2D scan, called a *scan line*. The Dynamixel actuator rotates the 2D laser scanner at 0.2 rotations per second, resulting in 200 scan lines per full rotation. Slower rotation is possible if a higher angular resolution is desired. For our current setup, we acquire one full 3D scan with up to 216,000 points per rotation every 5 seconds (shown in Figure 4a).

A PIXHAWK IMU is mounted close to the laser scanner, which is used for motion compensation during scan aggregation and attitude estimation.

An overview of our software system is shown in Figure 3. It consists of preprocessing steps to assemble 3D scans (Section 5.1), local mapping (Sections 4 and 5), global mapping (Section 6), and navigation planning (Section 7).

4. Local Multiresolution Map

Distance measurements from the laser-range sensor are accumulated in a 3D multiresolution map with increasing cell sizes from the robot center. The representation consists of multiple robot-centered 3D grid-maps with different resolutions. On the finest resolution, we use a cell length of 0.25 m. Each grid-map is embedded in the next level with coarser resolution and doubled cell length. The stored points and grid structure is shown in Figure 4.

We use a hybrid representation, storing 3D point measurements along with occupancy information in each cell. Point measurements of consecutive 3D scans are stored in fixed-sized circular buffers, allowing for point-based data processing and facilitating efficient nearest-neighbor queries. Figure 5 shows the point-based representation of the local multiresolution map during the SpaceBot Camp. It even shows relatively small objects—like a battery pack that the robot shall grasp.

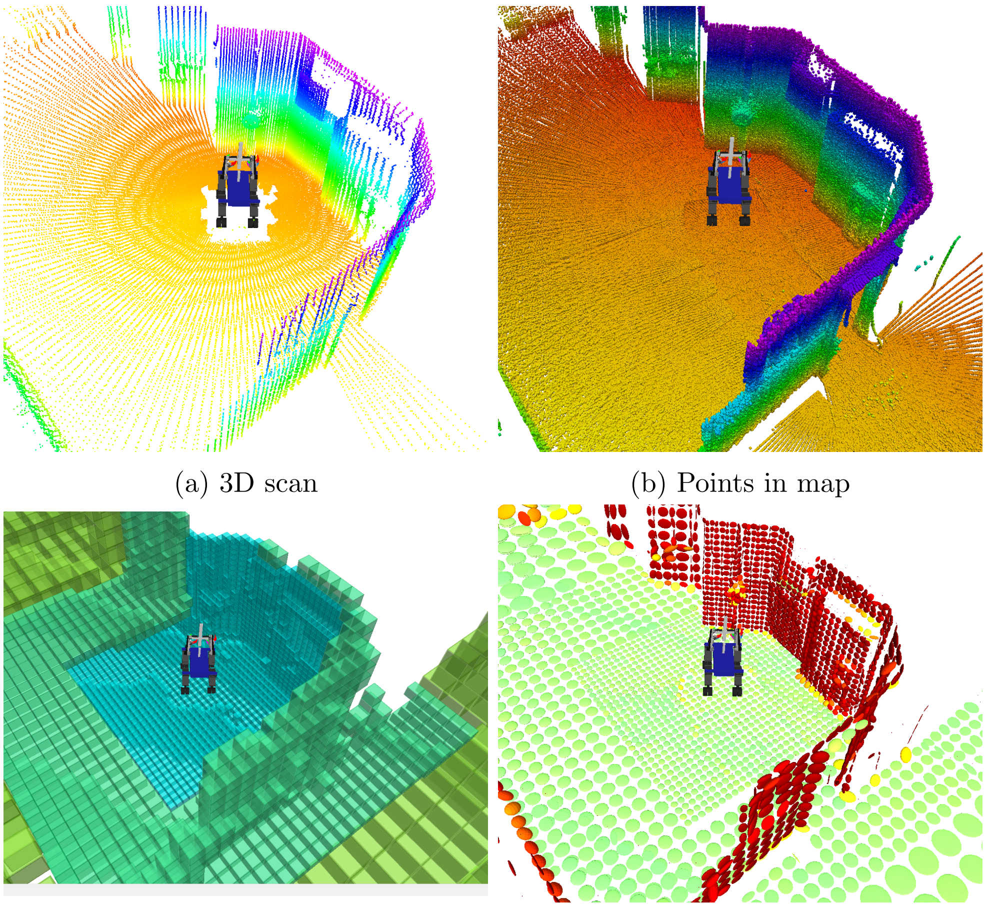
Figure 6 shows a 1D schematic illustration of the map organization. We aim for efficient map management for translation and rotation. Individual grid cells are stored in a circular buffer to allow for shifting elements in constant time. We interlace multiple circular buffers to obtain a map with three dimensions. The length of the circular buffers depends on the resolution and the size of the map. In case of a translation of the robot, the circular buffers are shifted whenever necessary to maintain the egocentric property of the map. In case of a translation equal or larger than the cell size, the circular buffers for respective dimensions are shifted. For sub-cell-length translations, the translational parts are accumulated and shifted if they exceed the length of a cell.

Since we store 3D points for every cell for point-based processing, single points are transformed in the local coordinate frame of a cell when adding, and back to the map coordinate frame when accessing. Every cell in the map stores a list of 3D points from the current and previous 3D scans. This list is also implemented by a fixed-sized circular buffer. If the capacity of the circular buffer is exceeded, old measurements are discarded and replaced by new measurements.

Rotating the map would necessitate to shuffle all cells. Consequently, our map is oriented independent to the robot orientation. We maintain the orientation between the map and the robot and use it to rotate measurements when accessing the map.

5. Scan Registration

We register consecutive 3D laser scans with our local multiresolution grid map to estimate the motion of the robot. Since the scans are taken while the robot is driving, the motion of the robot needs to be compensated for


(a) 3D scan

(b) Points in map

(c) Grid cells in map

(d) Surfels in map

Figure 4: The local multiresolution grid map during the first DRC competition run. (a): The 3D scan acquired with our continuously rotating laser scanner. (b): 3D points stored in the local multiresolution map. Color encodes height from ground. (c): The multiresolution grid structure of the map. Cell size (indicated by color) increases with the distance from the robot. (d): For every grid cell a surfel es summarizes the 3D points in the cell. Color encodes the orientation of the surfel.

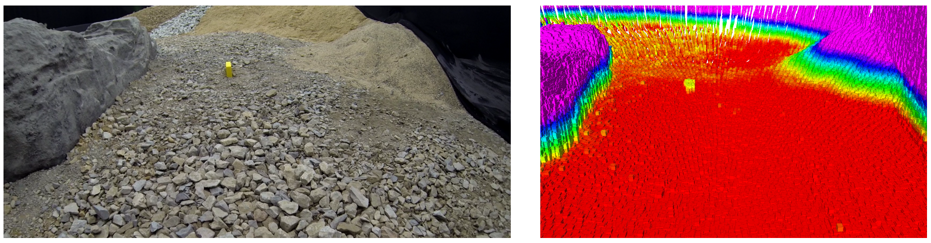


Figure 5: Photo and the corresponding local map of the battery pack—one of the objects to manipulate during the SpaceBot Camp. Color encodes distance from ground.

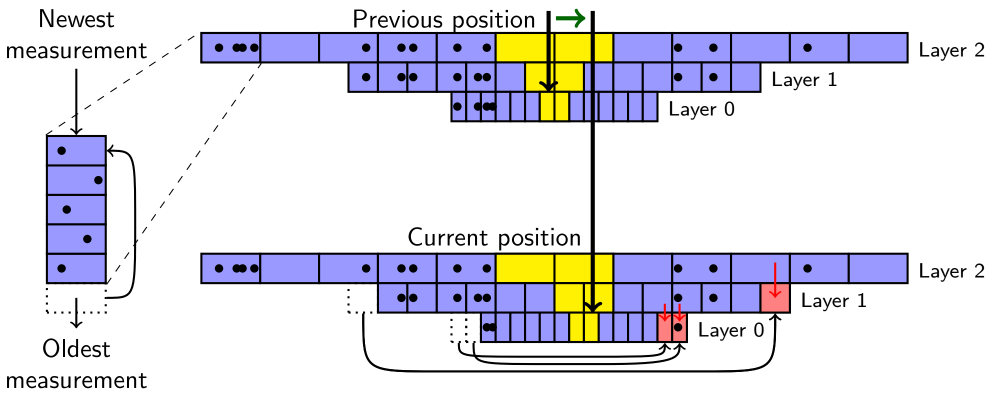


Figure 6: One-dimensional illustration of the hybrid local multiresolution map. Along with the occupancy information, every grid-cell (blue) maintains a circular buffer with its associated measurement points (green). The map is centered around the robot and in case of a robot motion, ring buffers are shifted according to the translational parts of the movement, maintaining the egocentric property of the map. Cells at coarser levels are used to retain points from vanishing cells at finer levels and to initialize newly added cells (red arrows).

when assembling the scan measurements into 3D scans. We register 3D scans with the so far accumulated map of the environment and update it with the registered 3D scan.

5.1. Preprocessing and 3D Scan Assembly

The raw measurements from the laser scanner are subject to spurious measurements at foreground-background transitions between two objects. These so-called *jump edges* are filtered by comparing the angle of neighboring measurements. After filtering for jump edges, we assemble a 3D scan from the 2D scans of a complete rotation of the scanner. Since the sensor is moving during acquisition, we undistort the individual 2D scans in two steps.

First, measurements of individual 2D scans are undistorted with regards to the rotation of the 2D laser scanner around the sensor rotation axis. Using spherical linear interpolation, the rotation between the acquisition of two scan lines is distributed over the measurements.

Second, the motion of the robot during acquisition of a full 3D scan is compensated. Due to Momaro's flexible legs, it is not sufficient to simply use wheel odometry to compensate for the robot motion. Instead, we estimate the full 6D state with the PIXHAWK IMU attached to Momaro's sensor head. Here we calculate a 3D attitude estimate from accelerometers and gyroscopes to compensate for rotational motions of the robot. Afterwards, we filter the wheel odometry with measured linear acceleration to compensate for linear motions. The resulting 6D state estimate includes otherwise unobservable motions due to external forces like rough terrain, contacts with the environment, wind, etc. It is used to assemble the individual 2D scans of each rotation to a 3D scan.

5.2. Scan To Map Registration

We register the points $\mathcal{P} = \{p_1, \ldots, p_P\}$ in a 3D scan with the points $\mathcal{Q} = \{q_1, \ldots, q_Q\}$ in the local grid map of the environment [1]. Similarly, the registration of two local maps is treated as the registration of their point sets. We formulate the registration of the 3D scan with the local environment map as optimizing the joint data-likelihood

$$p(\mathcal{P} \mid \theta, \mathcal{Q}) = \prod_{k=1}^{P} p(p_k \mid \theta, \mathcal{Q}). \tag{1}$$

Instead of considering each point individually, we map the 3D scan into a local multiresolution grid and match surfels, i.e.,

$$p(\mathcal{P} \mid \theta, \mathcal{Q}) \approx \prod_{i=1}^{N} p(x_i \mid \theta, Y)^{P_{x,i}}. \tag{2}$$

By this, several orders of magnitudes less map elements are used for registration. Figure 4d shows the surfels of an exemplary multiresolution map. We denote the set of surfels in the scene (the 3D scan) by $X = \{x_1, \ldots, x_N\}$ and write $Y = \{y_1, \ldots, y_M\}$ for the set of model surfels in the environment map. E.g., a surfel x_i summarizes its attributed $P_{x,i}$ points by their sample mean $\mu_{x,i}$ and covariance $\Sigma_{x,i}$. We assume that scene and model can be aligned by a rigid 6 degree-of-freedom (DoF) transformation $T(\theta)$ from scene to model. Our aim is to recover the relative pose θ of the scene towards the model.

5.3. Gaussian Mixture Observation Model

We explain each transformed scene surfel as an observation from a mixture model, similar as in the coherent point drift (CPD) method [10]. A surfel x_i is observed under the mixture defined by the model surfels and an additional uniform component that explains outliers, i.e.,

$$p(x_i \mid \theta, Y) = \sum_{j=1}^{M+1} p(c_{i,j})\ p(x_i \mid c_{i,j}, \theta, Y), \tag{3}$$

where $c_{i,j}$ is a shorthand for the 1-of-(M+1) encoding binary variable $c_i \in \mathbb{B}^{M+1}$ with j-th entry set to 1. Naturally, c_i indicates the association of x_i to exactly one of the mixture components. The model is a mixture on Gaussian components for the M model surfels through

$$p(x_i \mid c_{i,j}, \theta, Y) := \mathcal{N}\left[T(\theta)\mu_{x,i}; \mu_{y,j}, \Sigma_{y,j} + R(\theta)\Sigma_{x,i}R(\theta)^T + \sigma_j^2 I\right], \tag{4}$$

where $\sigma_j = \frac{1}{2}\rho_{y,j}^{-1}$ is a standard deviation that we adapt to the resolution $\rho_{y,j}$ of the model surfel. We set the likelihood of the uniform mixture component to $p(c_{i,M+1}) = w$. For this uniform component, the data likelihood of a surfel x_i is

$$p(x_i \mid c_{i,M+1}, \theta) = \frac{P_{x,i}}{P}\ \mathcal{N}(0; 0, R(\theta)\Sigma_{x,i}R(\theta)^T + \sigma_j^2 I). \tag{5}$$

For the prior association likelihood, we assume the likelihood of x_i to be associated to one of the points in the model map to be equal. Hence, for each Gaussian mixture component $j \in \{1, \ldots, M\}$ we have $p(c_{i,j}) = (1 - w)\frac{Q_{y,j}}{Q}$. By modeling the scene surfels as samples from a mixture on the model surfels, we do not make a hard association decision between the surfels sets, but a scene surfel is associated to many model surfels.

5.4. Registration through Expectation-Maximization

The alignment pose θ is estimated through maximization of the logarithm of the joint data-likelihood

$$\ln p(\mathcal{P} \mid \theta, \mathcal{Q}) \approx \sum_{i=1}^{N} P_{x,i} \ln \sum_{j=1}^{M+1} p(c_{i,j})\ p(x_i \mid c_{i,j}, \theta, Y). \tag{6}$$

We optimize this objective function through expectation-maximization (EM) [38]. The component associations $c = \{c_1, \ldots, c_N\}$ are treated as latent variables to yield the EM objective

$$L(q, \theta) := \sum_{i=1}^{N} P_{x,i} \sum_{j=1}^{M+1} q(c_{i,j}) \ln \frac{p(c_{i,j})\ p(x_i \mid c_{i,j}, \theta, Y)}{q(c_{i,j})}, \tag{7}$$

for which we exploit $q(c) = \prod_{i=1}^{N} \prod_{j=1}^{M+1} q(c_{i,j})$. In the M-step, the latest estimate $\overline{q}$ for the distribution over component associations is held fixed to optimize for the pose θ

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmax}}\ L(\overline{q}, \theta) \tag{8}$$

with

$$L(\overline{q}, \theta) := const. + \sum_{i=1}^{N} P_{x,i} \sum_{j=1}^{M+1} \overline{q}(c_{i,j}) \ln p(x_i \mid c_{i,j}, \theta, Y). \tag{9}$$

This optimization is efficiently performed using the Levenberg-Marquardt method as in [7]. The E-step obtains a new optimum $\widehat{q}$ for the distribution q by the conditional likelihood of the cluster associations given the latest pose estimate $\overline{\theta}$

$$\widehat{q}(c_{i,j}) = \frac{p(c_{i,j})\ p(x_i \mid c_{i,j}, \overline{\theta}, Y)}{\sum_{j'=1}^{M+1} p(c_{i,j'})\ p(x_i \mid c_{i,j'}, \overline{\theta}, Y)}. \tag{10}$$

In order to evaluate these soft assignments, we perform a local search in the local multiresolution surfel grid of the model. We first look-up the grid cell with a surfel available on the finest resolution in the model map at the transformed mean position of the scene surfel. We consider the surfels in this cell and its direct neighbors for soft association.

5.5. Filtering Dynamic Objects

Dynamics in the environment—caused e.g., by moving doors or debris—results in spurious measurements during mapping. Also, registration failures or fast motion of the laser during acquisition of a 3D scan, that could not be compensated by the IMU, result in abandoned measurements in the map. These spurious measurements can affect registration or distract the operator when using the map to plan manipulation tasks. We account for these by maintaining an occupancy probability—using log-odds notation to avoid multiplication when updating—for each cell in our multiresolution map. Similar to [13] we use a beam-based inverse sensor model and ray-casting to update the occupancy of a cell. For every measurement in the 3D scan, we update the occupancy information of cells on the ray between the sensor origin and the endpoint. Since this ray-casting operation is computationally expensive, we use an approximation to take advantage of the multiresolution structure of our map.

Before updating the occupancy information of the cells in question, we determine the endpoints of each beam—in our case the 3D point in the local coordinate system of the map—and the corresponding cell in every level. These cells are marked as occupied and are excluded from further occupancy updates of the 3D scan they belong to. We do this to prevent from artifacts caused by shallow angles between the line-of-sight of the sensor and the surface, as suggested by [13].

To update the occupancy information efficiently, we start with the coarsest level in our map and perform ray-casting with an approximated 3D Bresenham algorithm [39]. Information from the coarser level is used when updating the finer levels to quickly traverse empty spaces. In detail, we omit ray-casting points on the finer levels if the traversed cells on the coarsest level are observed free. An example is shown in Figure 7. One can observe that the opened door is quickly removed from the local map.

6. Allocentric Mapping and Localization

To estimate the motion of the robot, we incorporate IMU measurements, wheel odometry measurements and the the local registration results. While

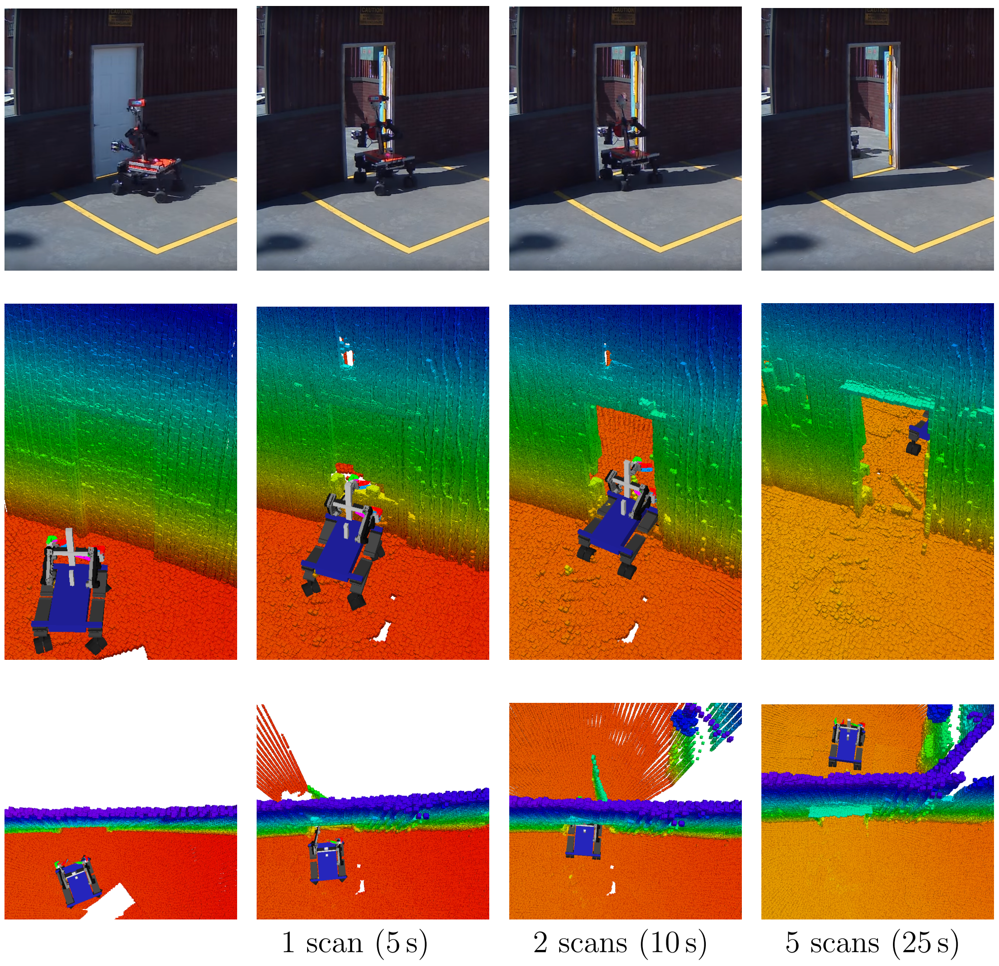


Figure 7: Filtering dynamic objects such as the door during the DRC Finals. After opening the door, abandoned measurements are filtered from the local multiresolution map. Camera image and the point-based representation of the map at 4 different time steps. The columns shows the map before opening the door and after adding 1, 2, and 5 scans. Color encodes height from the ground.

these estimates allow us to control the robot and to track its pose over a short period of time, they are prone to drift. Furthermore, they do not provide a fixed allocentric frame for the definition of mission-relevant poses. To overcome drift and to localize the robot with respect to a fixed frame, we build an allocentric map from local multiresolution maps acquired at different view poses [2].

Therefore, a pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed. Every node in the graph corresponds to a view pose and its local multiresolution map. Nearby nodes are connected by edges, modeling spatial constraints between two nodes. Each spatial constraint is a normally distributed estimate with mean and covariance. An edge $e_{ij} \in \mathcal{E}$ describes the relative position x_i^j between two nodes v_i and v_j, which arises from aligning two local multiresolution maps with each other. Similar to the alignment of a newly acquired 3D scan, two local multiresolution maps are aligned by our surfel-based registration method described in the previous section. Each edge models the uncertainty of the relative position by its information matrix, which is established by the covariance from registration.

During operation, the current local map is registered towards the closest node in the graph, the reference node v_{ref}. This allows us to track the current pose in the allocentric frame. A new node is generated for the current view pose, if the robot moved sufficiently far. In addition to edges between the previous node and the current node, we add spatial constraints between close-by nodes in the graph that are not in temporal sequence. Thus, we check for one new constraint between the current reference v_{ref} and other nodes v_{cmp}. We determine a probability

$$p_{\text{chk}}(v_{\text{cmp}}) = \mathcal{N}\left(d(x_{\text{ref}}, x_{\text{cmp}}); 0, \sigma_d^2\right) \tag{11}$$

that depends on the linear distance $d(x_{\text{ref}}, x_{\text{cmp}})$ between the view poses x_{ref} and x_{cmp}. According to $p_{\text{chk}}(v)$, we choose a node v from the graph and determine a spatial constraint between the nodes.

By adding edges between close-by nodes in the graph, we detect loop closures. Loop closure allows us to minimize drift from accumulated registration errors. For example, if the robot traverses unknown terrain and reenters a known part of the environment.

From the graph of spatial constraints, we infer the probability of the trajectory estimate given all relative pose observations

$$p(\mathcal{V} \mid \mathcal{E}) \propto \prod_{e_{ij} \in \mathcal{E}} p(x_i^j \mid x_i, x_j). \tag{12}$$

This pose graph optimization is efficiently solved using the g^{2}o framework [40], yielding maximum likelihood estimates of the view poses x_i. Optimization is performed when a loop closure has been detected, allowing for on-line operation.

6.1. Localization

While traversing the environment, the pose graph is extended whenever the robot explores previously unseen terrain and optimized when a loop closure has been detected. We localize towards this pose graph during mission to get the pose of the robot in an allocentric frame.

Since the laser scanner acquires complete 3D scans with a relatively low frame rate, we incorporate the egomotion estimate from the wheel odometry and measurements from the IMU. The egomotion estimate is used to track the pose of the robot w.r.t. the last localization result between two consecutive 3D scans. In detail, we track the pose hypothesis by alternating the prediction of the robot movement given the filter result and alignment of the current local multiresolution map towards the allocentric map of the environment.

The allocentric localization is triggered after acquiring a 3D scan and adding it to the local multiresolution map. Therefore, the updated local map is registered towards the closest node in the graph. By aligning the dense local map to the pose graph—instead of the relative sparse 3D scan—we gain robustness, since information from previous 3D scans is incorporated. We update the allocentric robot pose with the resulting registration transform. To achieve real-time performance of the localization module, we track only one pose hypothesis.

During the SpaceBot Camp, we assumed that the initial pose of the robot was known, either by starting from a predefined pose or by means of manually aligning our allocentric coordinate frame with a coarse height map of the environment. Thus, we could navigate to goal poses in the coarse height map by localizing towards our pose graph.

7. Navigation

One important application of the allocentric 3D maps and the localization approach is autonomous navigation. In order to demonstrate the suitability of our approach in this domain, we briefly discuss our navigation pipeline, even though it does not contain novel ideas. Figure 8 gives an overview of

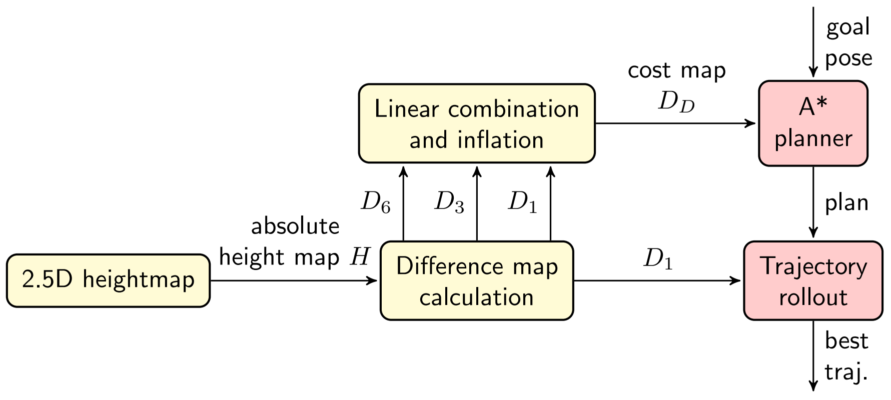


Figure 8: Data flow of our navigation method. Data filtering/processing modules are colored yellow and navigation components red.

our the pipeline. Our approach is based on the RGB-D-based local navigation approach of [22], which is now used on the 3D laser measurements. Furthermore, we make no distinction between local and global navigation. For details on the approach, we refer to [22].

For most terrains, a 2.5D height map contains all information necessary for navigation. This reduction greatly reduces the amount of data to be processed and allows planning in real time. The allocentric 2.5D height map is represented as a 2D grid with a resolution of 5×5 cm. For each map cell $H(x, y)$, we calculate the median height of the points whose projections onto the horizontal plane lie in the map cell.

An absolute height map is not meaningful for planning local paths or for avoiding obstacles. To assess drivability, the allocentric height map is transformed into a height difference map. We calculate local height differences at multiple scales l. Let $D_l(x, y)$ be the maximum difference to the center pixel (x, y) in a local l-window:

$$D_l(x, y) := \max_{\substack{|u-x|<l;u\neq x\\ |v-y|<l;v\neq y}} |H(x, y) - H(u, v)|\,.$$

Missing $H(u, v)$ values indicated by NaN are ignored. If the center pixel $H(x, y)$ itself is not defined, or there are no other defined l-neighbors, we assign $D_l(x, y) :=$ NaN.

Small, but sharp obstacles show up on the D_l maps with lower l scales. Larger inclines, which might be better to avoid, can be seen on the maps with a higher l value.

The height difference maps are transformed into cost maps as in [22]. In particular, the cost map for path planning is inflated by the robot radius (for an example, see Figure 13). We conduct a standard A* search on the graph defined by the 8-neighborhood in the inflated cost map.

To determine forward driving speed and rotational speed that follow the planned trajectory and avoid obstacles, we use the ROS trajectory rollout planner (*dwa_planner*). Replanning is done at least once every second to account for robot movement and novel terrain percepts.

8. Evaluation

This section describes the evaluation of our system during two public events, the DARPA Robotics Challenge Finals and the DLR SpaceBot Camp. The datasets used for the experiments are from our runs during the competition. Since quantitative measures are hard to generate—especially on rough terrain or disaster scenarios due to the absence of a reference measure—we focus on qualitative evaluation. We make the used data sets available on our website[3]. Parts of our system have been evaluated independently in our previous work. For example, our surfel-based registration method has been compared to state-of-the-art registration methods on data from a motion capture system [1], showing that it is more accurate and computationally more efficient. The data sets shown in the experiments are made publicly available[4].

8.1. DARPA Robotics Challenge

Since the robots could be teleoperated during the competition, we did not use our allocentric mapping and localization at the DRC. The local mapping components where used to build an egocentric map of the robot's direct vicinity. This map was used by the manipulation operator when planning motions. Besides that, the navigation operator used the resulting local maps and height images build from it to assess driveability. Also, the result of the

[3]Data sets captured during the DRC competition run and the DLR SpaceBot Camp demonstration `http://www.ais.uni-bonn.de/data/3D-Laser.html`

[4]`http://www.ais.uni-bonn.de/laser_mapping`

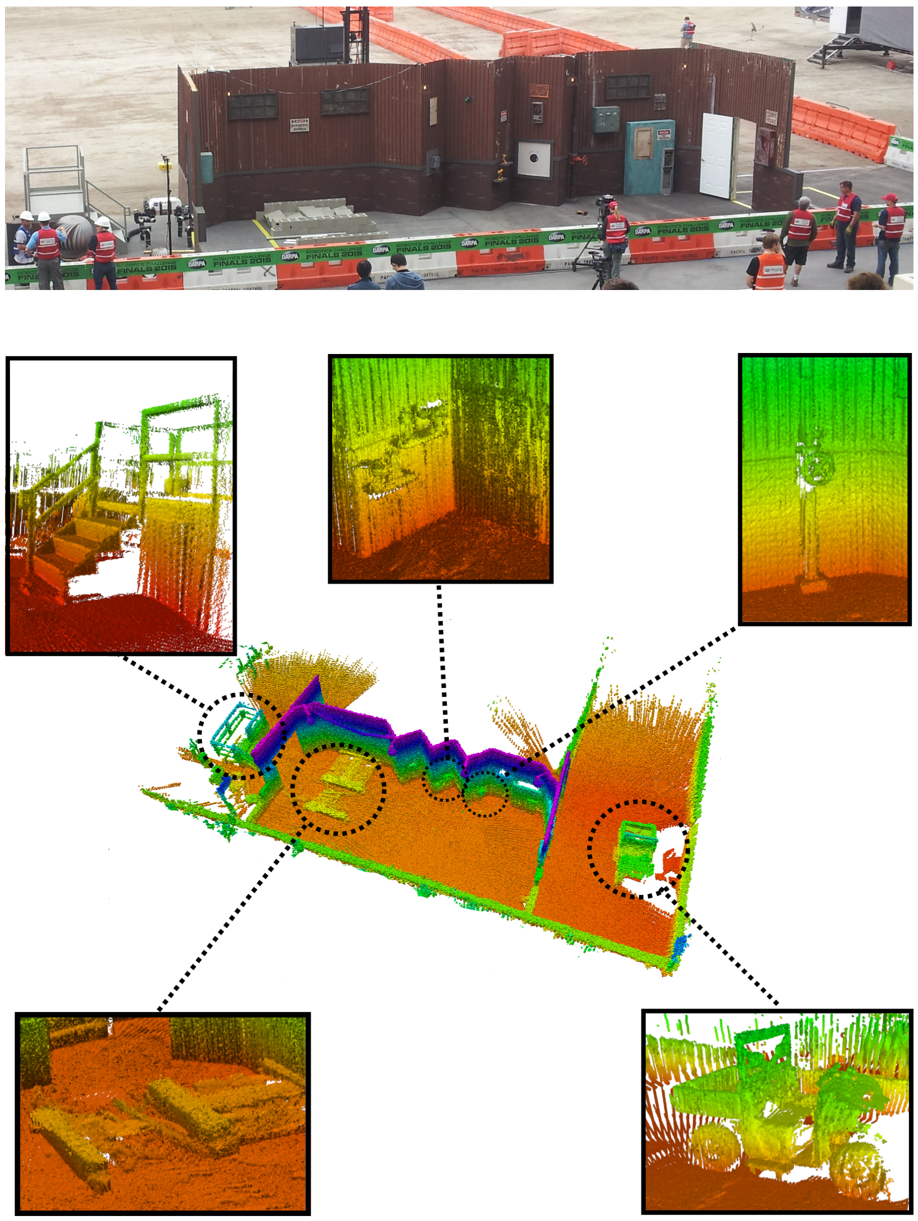


Figure 9: Top: The mock-up disaster scenario of the DRC. Bottom: The resulting allocentric map generated from the data of our first competition run. Color encodes the height from ground.

registration corrects odometry drift of the robot when aligning to a previously acquired local map. Figure 9 shows the resulting allocentric map generated from the dataset of our first-day competition. Besides the allocentric map, selected local multiresolution maps of the pose graph are shown. Although reference data is not available, one can see that the resulting allocentric map is globally consistent and accurate, as indicated by the straight walls and plain floor. Also the local maps look clear and accurate.

Our team was able to solve seven out of eight tasks in the shortest time of all teams who solved seven tasks, which yielded a fourth place in the final ranking as the best European team. While we attribute part of our success to our flexible teleoperation solutions [37], the quality of the 3D environment perception and thus the situational awareness of the operator crew played a large part and was a necessary precondition for developing said teleoperation interfaces. Further information on our DRC competition entry is available of our website[5], including a video or our first day competition run[6].

8.2. SpaceBot Camp 2015

At the DLR SpaceBot Camp, robots had to conduct an exploration mission in a (simulated) extraterrestrial planetary environment. The mission was—based on a rough height image of the environment—to explore and map the environment and to manipulate objects in it. In contrast to the DRC, the robots did not have a permanent network connection that allowed for teleoperation. Consequently—in addition to local mapping—we used our allocentric mapping component and the described planning approach to allow for fully autonomous navigation.

The planetary-like environment was specially challenging due to the rough surface of the terrain, consisting of different types of stones and soil that caused slip in odometry and high-frequency motion of robot and sensor. Due to the relative small wheels of our robot, an accurate terrain map was necessary to assess driveability. The environment and the resulting allocentric map are shown in Figure 10. It was continuously built during autonomous navigation guided by waypoints specified on the rough height map. One can see, that although the robot was autonomously navigating in rough terrain the resulting allocentric map is accurate and precisely models the environment.

[5]Website of our DRC entry `http://www.nimbro.net/Rescue`

[6]Video of first day DRC competition run `http://youtu.be/NJHSFelPsGc`

Figure 11 shows the allocentric map at different time steps. The figure shows how the map is extended during a mission. New nodes (i.e., local multiresolution maps) are added to the pose graph and new nodes are connected to existing nodes by edges. During a mission, the map is used for localization as shown in Figure 12 and to assess traversability for navigation as shown in Figure 13. Our system was able to solve all tasks with few interventions by the operator crew over the degraded communication link, such as stopping navigation before a scheduled communication blackout or re-triggering a failed manipulation task. Further information on our SpaceBot Camp entry is available of our website[7], including a video or our demonstration run[8].

[7]Website of our DLR SpaceBot Camp entry `http://www.nimbro.net/Explorer`

[8]Video of SpaceBot Camp demonstration run `http://youtu.be/q_p5ZO-BKWM`

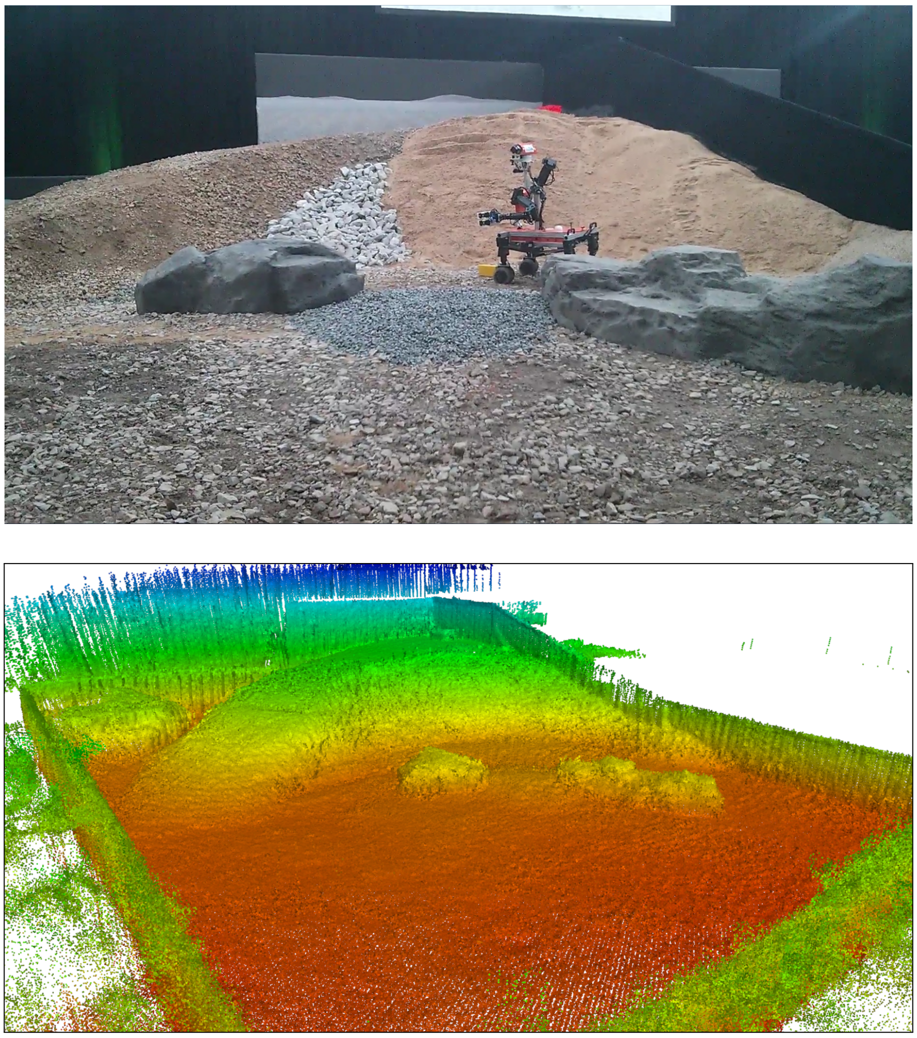


Figure 10: Top: Photo of the planetary-like environment at the DLR SpaceBot Camp consisting of different types of stones and soil. Bottom: The resulting 3D map built by our mapping component from data that has been collected during our run. Color encodes distance from ground.

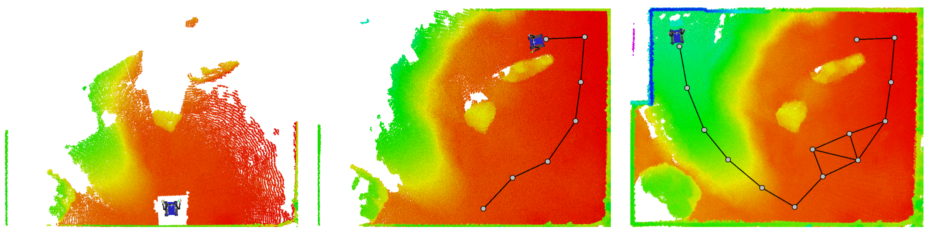


Figure 11: The allocentric map from a top view at different time steps, consisting of 1 (left), 7 (middle) and 14 (right) key frames. Color encodes height. The nodes in the pose graph (grey circles) are connected by spatial constraints (black lines). The robot model shows the current position of the robot.

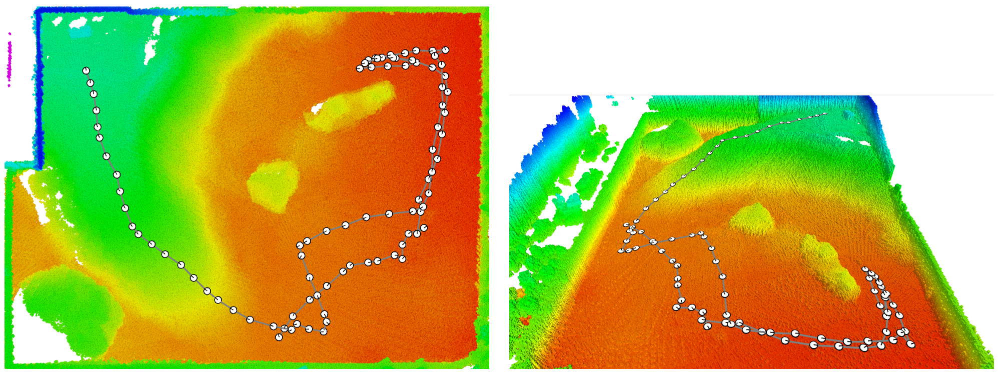


Figure 12: The resulting allocentric map from two different perspectives with the localization poses (black circle) from our run. Color encodes height from ground.

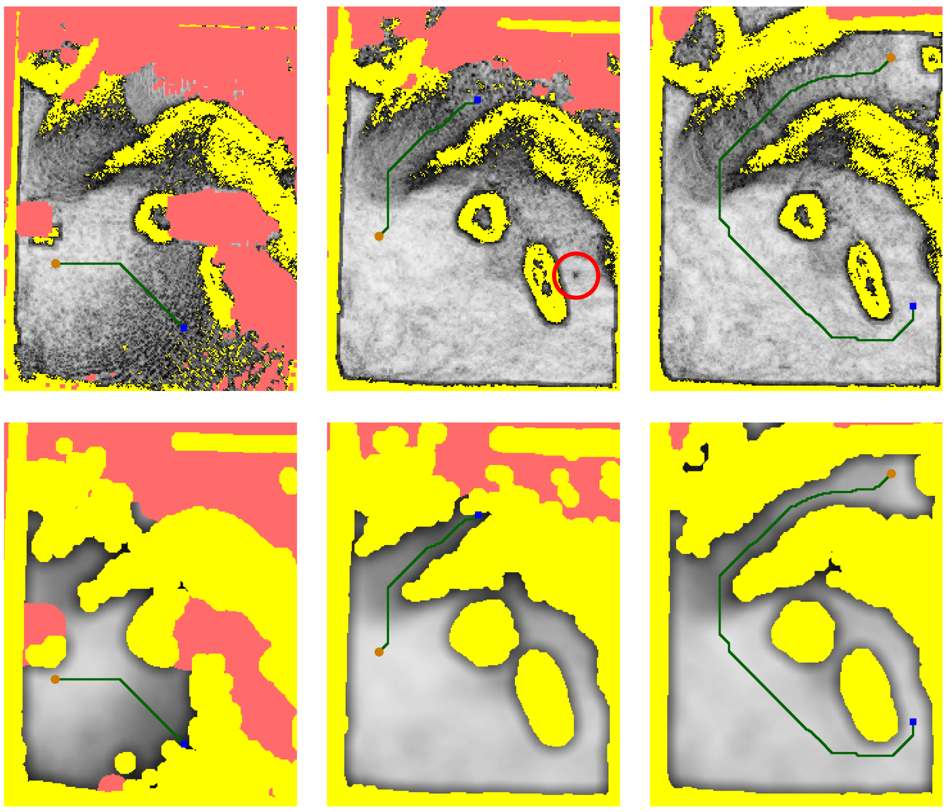


Figure 13: Navigation planning during (left, middle) and after exploration (right) of the SpaceBot Camp arena. The top row shows the calculated traversability costs for each cell. The bottom row shows inflated costs used for A* path planning. The orange dot represents the current robot position, the blue square the target position. The planned path is shown in green. Red areas indicate insufficient measurements for traversability analysis. Yellow areas correspond to absolute obstacles, which the robot may not traverse. In the middle situation, a small battery pack (20 cm × 10 cm × 4 cm) can be seen in the uninflated costs (marked with red circle, also shown in Figure 5).

9. Conclusions

We presented our local and allocentric mapping systems that uses an efficient 3D multiresolution map to aggregate measurements from a continuously rotating laser scanner and align acquired scans with it. By using local multiresolution, we gain computational efficiency by having a high resolution in the near vicinity of the robot and a lower resolution with increasing distance from the robot, which correlates with the sensor characteristics in relative distance accuracy and measurement density.

Scan registration is used to estimate the motion of the robot by aligning consecutive 3D scans to the map. We do not match individual scan points, but represent 3D scans also in local multiresolution grids and condense the points into surface elements for each grid cell. These surface elements are aligned efficiently and at high accuracy in a registration framework which overcomes the discretization in a grid through probabilistic assignments.

Modeling measurement distributions within voxels by surface elements allows for efficient and accurate registration of 3D scans with the local map. The incrementally built local dense 3D maps of nearby key poses are registered globally by graph optimization. This yields a globally consistent dense 3D map of the environment. Continuous registration of local maps with the global map allows for tracking the 6D robot pose in real time. We demonstrate accuracy and efficiency of our approach by showing consistent allocentric 3D maps in difficult environments with rough terrain.

The high-quality 3D environment representations were a key success factor for our team in the competitions. During the DRC Finals, our team NimbRo Rescue solved seven of the eight tasks in only 34 minutes, coming in 4th overall. Our team NimbRo Explorer was the only team to solve all tasks of the DLR SpaceBot Camp.

[1] D. Droeschel, J. Stückler, S. Behnke, Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner, in: Robotics and Automation (ICRA), IEEE International Conference on, 2014.

[2] D. Droeschel, J. Stückler, S. Behnke, Local multi-resolution surfel grids for mav motion estimation and 3d mapping, in: Proc. of the Int. Conf. on Intelligent Autonomous Systems, 2014.

[3] A. Nuechter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM with approximate data association, in: Int. Conf. on Advanced Robotics, 2005.

[4] M. Magnusson, T. Duckett, A. J. Lilienthal, Scan registration for autonomous mining vehicles using 3D-NDT, Journal of Field Robotics 24 (10) (2007) 803–827.

[5] S. Kohlbrecher, O. von Stryk, J. Meyer, U. Klingauf, A flexible and scalable slam system with full 3d motion estimation, in: Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, 2011, pp. 155–160.

[6] D. Cole, P. Newman, Using laser range data for 3d slam in outdoor environments, in: Robotics and Automation (ICRA), IEEE International Conference on, 2006, pp. 1556–1563.

[7] J. Stückler, S. Behnke, Multi-resolution surfel maps for efficient dense 3D modeling and tracking, Journal of Visual Communication and Image Representation 25 (1) (2014) 137–147.

[8] M. Schadler, J. Stückler, S. Behnke, Multi-resolution surfel mapping and real-time pose tracking using a continuously rotating 2D laser scanner, in: Proc. of 11th IEEE Int. Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013.

[9] P. J. Besl, N. D. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14 (2) (1992) 239–256.

[10] A. Myronenko, X. Song, Point set registration: Coherent point drift, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (12) (2010) 2262–2275.

[11] R. Horaud, F. Forbes, M. Yguel, G. Dewaele, J. Zhang, Rigid and articulated point registration with expectation conditional maximization, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (3) (2011) 587–602.

[12] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, E. Z. Psarakis, A generative model for the joint registration of multiple point sets, in: European Conference on Computer Vision, Springer, 2014, pp. 109–122.

[13] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: an efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots 34 (2013) 189–206.

[14] J. Ryde, H. Hu, 3D mapping with multi-resolution occupied voxel lists, Autonomous Robots 28 (2010) 169 – 185.

[15] T. Stoyanov, M. Magnusson, H. Andreasson, A. J. Lilienthal, Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations, The International Journal of Robotics Research 31 (12) (2012) 1377–1393.

[16] D. Belter, P. Łabcki, P. Skrzypczyński, Estimating terrain elevation maps from sparse and uncertain multi-sensor data, in: Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on, IEEE, 2012, pp. 715–722.

[17] M. Hebert , C. Caillas, E. Krotkov, I. S. Kweon, T. Kanade, Terrain mapping for a roving planetary explorer, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '89), Vol. 2, 1989, pp. 997–1002.

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

[19] P. Pfaff, R. Triebel, W. Burgard, An efficient extension to elevation maps for outdoor terrain mapping and loop closing, The International Journal of Robotics Research 26 (2) (2007) 217–230.

[20] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, R. Siegwart, Robot-centric elevation mapping with uncertainty estimates, in: International Conference on Climbing and Walking Robots (CLAWAR), Poznań, Poland, 2014, pp. 433–440.

[21] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stückler, S. Behnke, Multilayered mapping and navigation for autonomous micro aerial vehicles, Journal of Field Robotics (JFR).

[22] J. Stückler, M. Schwarz, M. Schadler, A. Topalidou-Kyniazopoulou, S. Behnke, NimbRo Explorer: Semiautonomous exploration and mobile manipulation in rough terrain, Journal of Field Robotics (JFR).

[23] M. Bosse, R. Zlot, Continuous 3D scan-matching with a spinning 2D laser, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2009.

[24] J. Elseberg, D. Borrmann, A. Nuechter, 6DOF semi-rigid SLAM for mobile scanning, in: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, 2012.

[25] T. Stoyanov, A. Lilienthal, Maximum likelihood point cloud acquisition from a mobile platform, in: Proc. of the Int. Conf. on Advanced Robotics (ICAR), 2009.

[26] W. Maddern, A. Harrison, P. Newman, Lost in translation (and rotation): Fast extrinsic calibration for 2D and 3D LIDARs, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2012.

[27] S. Anderson, T. D. Barfoot, Towards relative continuous-time SLAM, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2013.

[28] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, S. Chitta, Navigation in three-dimensional cluttered environments for mobile manipulation, in: Robotics and Automation (ICRA), IEEE International Conference on, 2012.

[29] J. Klaess, J. Stueckler, S. Behnke, Efficient mobile robot navigation using 3D surfel grid maps, in: Proc. German Conf. on Robotics (ROBOTIK), 2012.

[30] J. Lee, C. Pippin, T. Balch, Cost based planning with rrt in outdoor environments, in: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, 2008.

[31] B. P. Gerkey, K. Konolige, Planning and control in unstructured terrain, in: Proc. of the ICRA Workshop on Path Planning on Costmaps, 2008.

[32] D. Ferguson, M. Likhachev, Efficiently using cost maps for planning complex maneuvers, in: Proc. of the ICRA Workshop on Planning with Cost Maps, 2008.

[33] T. Stoyanov, M. Magnusson, H. Andreasson, A. J. Lilienthal, Path planning in 3D environments using the normal distributions transform, in: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, 2010.

[34] R. Bogdan Rusu, A. Sundaresan, B. Morisset, K. Hauser, M. Agrawal, J.-C. Latombe, M. Beetz, Leaving flatland: Efficient real-time three-dimensional perception and motion planning, Journal of Field Robotics 26 (10) (2009) 841–862.

[35] S. Chhaniyara, C. Brunskill, B. Yeomans, M. Matthews, C. Saaj, S. Ransom, L. Richter, Terrain trafficability analysis and soil mechanical property identification for planetary rovers: A survey, J. Terramechanics 49 (2) (2012) 115–128.

[36] P. Papadakis, Terrain traversability analysis methods for unmanned ground vehicles: A survey, Engineering Applications of Artificial Intelligence 26 (4) (2013) 1373–1385.

[37] T. Rodehutskors, M. Schwarz, S. Behnke, Intuitive bimanual telemanipulation under communication restrictions by immersive 3d visualization and motion tracking, in: Proc. of the IEEE-RAS Int. Conference on Humanoid Robots (Humanoids), 2015.

[38] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[39] J. Amanatides, A. Woo, A fast voxel traversal algorithm for ray tracing, in: In Eurographics 87, 1987, pp. 3–10.

[40] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, G2o: A general framework for graph optimization, in: Robotics and Automation (ICRA), IEEE International Conference on, 2011.

Supervised Autonomy for Exploration and Mobile Manipulation in Rough Terrain with a Centaur-like Robot

Max Schwarz*, Marius Beul, David Droeschel, Sebastian Schüller, Arul Selvam Periyasamy, Christian Lenz, Michael Schreiber and Sven Behnke

Institute for Computer Science VI, Autonomous Intelligent Systems, University of Bonn, Bonn, Germany

Correspondence*:
Max Schwarz
Institute for Computer Science VI, Autonomous Intelligent Systems, University of Bonn, Friedrich-Ebert-Allee 144, Bonn, 53113, Germany,
max.schwarz@uni-bonn.de

ABSTRACT

Planetary exploration scenarios illustrate the need for autonomous robots that are capable to operate in unknown environments without direct human interaction. At the DARPA Robotics Challenge, we demonstrated that our Centaur-like mobile manipulation robot Momaro can solve complex tasks when teleoperated. Motivated by the DLR SpaceBot Cup 2015, where robots should explore a Mars-like environment, find and transport objects, take a soil sample, and perform assembly tasks, we developed autonomous capabilities for Momaro. Our robot perceives and maps previously unknown, uneven terrain using a 3D laser scanner. Based on the generated height map, we assess drivability, plan navigation paths, and execute them using the omnidirectional drive. Using its four legs, the robot adapts to the slope of the terrain. Momaro perceives objects with cameras, estimates their pose, and manipulates them with its two arms autonomously. For specifying missions, monitoring mission progress, on-the-fly reconfiguration, and teleoperation, we developed a ground station with suitable operator interfaces. To handle network communication interruptions and latencies between robot and ground station, we implemented a robust network layer for the ROS middleware. With the developed system, our team NimbRo Explorer solved all tasks of the DLR SpaceBot Camp 2015. We also discuss the lessons learned from this demonstration.

Keywords: Mapping, Mobile Manipulation, Navigation, Perception for Grasping and Manipulation, Space Robotics and Automation

1 INTRODUCTION

In planetary exploration scenarios, robots are needed that are capable to autonomously operate in unknown environments and highly unstructured and unpredictable situations. Since human workers cannot be deployed due to economic or safety constraints, autonomous robots have to robustly solve complex tasks without human intervention. To address this need, the German Aerospace Center (DLR) held the DLR

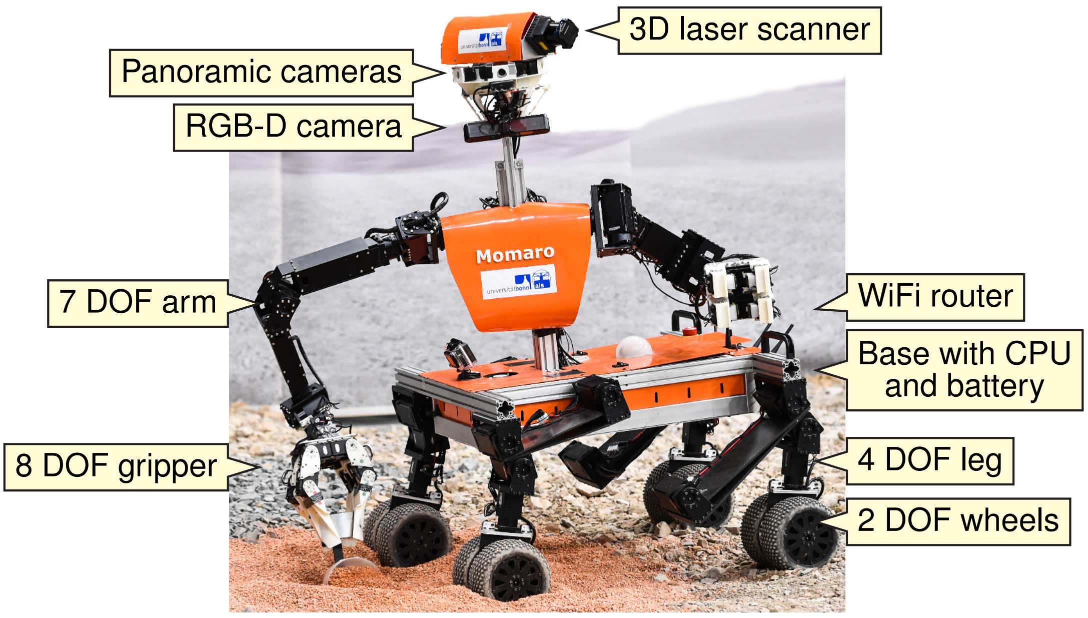


Figure 1. The mobile manipulation robot Momaro taking a soil sample.

SpaceBot Camp 2015[1]. Ten German research groups were supported to foster the development of robots, capable of autonomously solving complex tasks which are required in a typical planetary exploration scenario. During the SpaceBot Camp, the robots needed to tackle these tasks:

- Find and identify three previously known objects in a planetary-like environment (cup, battery, and base station).
- Take a soil sample of a previously known spot (optional).
- Pick up and deliver the cup and the battery to the base station.
- Assemble all objects.

All tasks had to be completed as autonomously as possible, including perception, manipulation and navigation in difficult terrain with slopes up to 15° that needed to be traversed and larger untraversable slopes. The overall weight of the deployed robotic system was limited to 100 kg and the total time for solving all tasks was 60 min. A rough height map with 50 cm resolution of the environment was known prior to the run. The use of any global navigation satellite system (GNSS) was prohibited. No line-of-sight between the robot and the crew was allowed and communication between the robot and the operators was severely restricted. Data transmission was bidirectionally delayed by 2 s, resulting in a round trip time of 4 s – too large for direct remote control. Furthermore, the uplink connection was blocked entirely after 20 min and 40 min for 4 min each. More details on the SpaceBot Camp itself and our performance are provided in Section 11.

To address the tasks, we used the mobile manipulation robot Momaro (see Fig. 1), which is configured and monitored from a ground station. Momaro is equipped with four articulated compliant legs that end in pairs of directly driven, steerable wheels. To perform a wide range of manipulation tasks, Momaro has an anthropomorphic upper body with two 7 degrees of freedom (DOF) manipulators that end in dexterous grippers. This allows for the single-handed manipulation of smaller objects, as well as for two-armed

[1] `http://www.dlr.de/rd/desktopdefault.aspx/tabid-8101/`

manipulation of larger objects and the use of tools. Through adjustable base-height and attitude and a yaw joint in the spine, Momaro has a work space equal to the one of an adult person.

The SpaceBot Camp constitute a challenge for autonomous robots. Since the complex navigation and manipulation tasks require good situational awareness, Momaro is equipped with a 3D laser scanner, multiple color cameras, and an RGB-D camera. For real-time perception and planning, Momaro is equipped with a powerful onboard computer. The robot communicates to a relay at the landing site via WiFi and is equipped with a rechargeable LiPo battery (details provided in Section 3).

The developed system was tested at the SpaceBot Camp 2015. Momaro solved all tasks autonomously in only 20:25 out of 60 minutes including the optional soil sample. No official ranking was conducted at the SpaceBot Camp, but since we were the only team solving all these tasks, we were very satisfied with the performance. We report in detail on how the tasks were solved. Our developments led to multiple contributions, which are summarized in this article, including the robust perception and state estimation system, navigation and motion planning modules and autonomous manipulation and control methods. We also discuss lessons learned from the challenging robot operations.

2 RELATED WORK

The need for mobile manipulation has been addressed in the past with the development of a variety of mobile manipulation systems, consisting of robotic arms installed on mobile bases with the mobility provided by wheels, tracks, or leg mechanisms. Several research projects exist which use purely wheeled locomotion for their robots (Mehling et al., 2007; Borst et al., 2009). In previous work, we developed NimbRo Explorer (Stückler et al., 2015), a six-wheeled robot equipped with a 7 DOF arm designed for mobile manipulation in rough terrain encountered in planetary exploration scenarios.

Wheeled rovers provide optimal solutions for well-structured, and relatively flat environments, however, outside of these types of terrains, their mobility quickly reaches its limits. Often they can only overcome obstacles smaller than the size of their wheels. Compared to wheeled robots, legged robots are more complex to design, build, and control (Raibert et al., 2008; Roennau et al., 2010; Semini et al., 2011; Johnson et al., 2015) but they have obvious mobility advantages when operating in unstructured terrains and environments. Some research groups have started investigating mobile robot designs which combine the advantages of both legged and wheeled locomotion, using different coupling mechanisms between the wheels and legs (Adachi et al., 1999; Endo and Hirose, 2000; Halme et al., 2003). In the context of the DARPA Robotics Challenge, multiple teams (beside ours) used hybrid locomotion designs (Stentz et al., 2015; Hebert et al., 2015). In particular, the winning team KAIST (Cho et al., 2011; Kim and Oh, 2010) used wheels on the knees of their humanoid robot to move quickly and safely between different tasks on flat terrain.

In 2013, DLR held a very similar SpaceBot competition which encouraged several robotic developments (Kaupisch et al., 2015). Heppner et al. (2015) describe one of the participating systems, the six-legged walking robot LAURON V. LAURON is able to overcome challenging terrain, although its six legs limit the locomotion speed in comparison to wheeled robots. As with our system, the software architecture is based on the Robot Operating System (ROS, Quigley et al., 2009).

Sünderhauf et al. (2014) developed a cooperative team of two wheeled robots, named Phobos and Deimos. The straightforward, rugged design with skid steering performed well, compared to more complicated locomotion approaches. We made the same observation in our participation at the SpaceBot Competition 2013, and opted to include wheels (opposed to a purely legged concept) in the Momaro robot. In the 2013 competition, Phobos and Deimos mainly had communication issues such that the ground station crew could neither stop Phobos from colliding with the environment, nor start Deimos to resume the mission. These problems highlight why we spent considerable effort on our communication subsystem (see Section 9) to ensure that the operator crew has proper situational awareness and is able to continuously supervise the robotic operation.

Schwendner et al. (2014) and Joyeux et al. (2014) discuss the six-wheeled Artemis rover. Artemis is able to cope with considerable terrain slopes (up to $45°$) through careful mechanical design. In contrast, Momaro has to employ active balancing strategies (see Section 6) to prevent tipping over due to its high center of mass. The authors emphasize the model-driven design of both hard- and software. The latter is partly ROS-based, but also has modules based on the Rock framework. Artemis demonstrated its navigation capabilities in the 2013 competition, but eventually its navigation planners became stuck in front of a trench, again highlighting the need to design systems with enough remote access so that problems can be diagnosed and fixed remotely.

A few articles on the SpaceBot Camp 2015 are already available. Kaupisch and Fleischmann (2015) describe the event and report briefly on the performances of all teams. Wedler et al. (2015) present the general design of their Lightweight Rover Unit (LRU), which competed in the SpaceBot Camp 2015, successfully solving all tasks except the optional soil sample task. The LRU is a four-wheeled rover with steerable wheels, similar to Momaro's drive. Comparable to our flexible legs, the suspension uses both active and passive mechanisms. However, the LRU wheels are rigidly coupled in pairs and the base height cannot be adapted. Overall, the LRU seems geared towards building a robust and hardened rover for real missions, while Momaro's components are not suitable for space. On the other hand, Momaro can solve tasks requiring stepping motions and is capable of dexterous bimanual manipulation.

In our previous work, we describe the Explorer system used in the 2013 competition (Stückler et al., 2015) and its local navigation system (Schwarz and Behnke, 2014). Compared to the 2013 system, we improve on the

- capabilities of the mechanical design (e.g., execution of stepping motions or bimanual manipulation),
- grade of autonomy (execution of full missions, including assembly tasks at the base station),
- situational awareness of the operator crew,
- robustness of network communication.

The local navigation approach has moved from a hybrid laser-scanner-and-RGB-D system on three levels to a laser scanner-only system on two levels—allowing operation in regions where current RGB-D sensors fail to measure distance (e.g., in direct sunlight).

In contrast to many other systems, Momaro is capable of driving omnidirectionally, which simplifies navigation in restricted spaces and allows us to make small lateral positional corrections faster. Furthermore, our robot is equipped with six limbs, two of which are exclusively used for manipulation. The use of four legs for locomotion provides a large and flexible support polygon when the robot is performing mobile manipulation tasks. The Momaro system demonstrated multiple complex tasks under teleoperation in the DARPA Robotics Challenge (see Schwarz et al., 2016).

Supervised autonomy has been proposed as a development paradigm by Cheng and Zelinsky (2001), who shift basic autonomous functions like collision avoidance from the supervisor back to the robot, while offering high-level interfaces to configure the functions remotely. In contrast to human-in-the-loop control, supervised autonomy is more suited towards the large latencies involved in space communications. Gillett et al. (2001) use supervised autonomy in the context of an unmanned satellite servicing system that must perform satellite capture autonomously. The survey conducted by Pedersen et al. (2003) highlights the (slow) trend in space robotics towards more autonomous functions, but also points out that space exploration will always have a human component, if only as consumers of the data produced by the robotic system. In this manner, supervised autonomy is also the limit case of sensible autonomy in space exploration.

3 MOBILE MANIPULATION ROBOT MOMARO

3.1 Mechanical Design

Our mobile manipulation robot Momaro (see Fig. 1) was constructed with several design goals in mind:

- universality,
- modularity,
- simplicity, and
- low weight.

In the following, we detail how we address these goals.

3.1.1 Universality

Momaro features a unique locomotion design with four legs ending in steerable wheels. This design allows to omnidirectionally drive and to step over obstacles or even climb. Since it is possible to adjust the total length of the legs, Momaro can manipulate obstacles on the ground, as well as reach on heights of up to 2 m. Momaro can adapt to the slope of the terrain through leg length changes.

On its base, Momaro has an anthropomorphic upper body with two adult-sized 7 DOF arms, enabling it to solve complex manipulation tasks. Attached to the arms are two 8 DOF dexterous hands consisting of four fingers with two segments each. The distal segments are 3D printed and can be changed without tools for easy adaption to a specific task. For the SpaceBot Camp, we designed distal finger segments that maximize the contact surface to the SpaceBot objects: The finger tips are shaped to clamp around the circumference of the cylindrical cup object (see Fig. 3). The box-shaped battery object is first grasped using the proximal finger segments, and then locked in place with the distal finger segments as soon as it is lifted from the ground.

The upper body can be rotated around the spine with an additional joint, thus increasing the workspace. Equipped with these various DOF, Momaro can solve most diverse tasks. If necessary, Momaro is even able to use tools. We showed this ability by taking a soil sample with a scoop at the SpaceBot Camp (see Fig. 2).

3.1.2 Modularity

All joints of Momaro are driven by Robotis Dynamixel actuators, which offer a good torque-to-weight ratio. While the finger actuators and the rotating laser scanner actuator are of the MX variant, all others are Dynamixel Pro actuators. Fig. 3 gives an overview of the DOF of Momaro. For detailed information on Momaro's actuators, we refer to (Schwarz et al., 2016).

Using similar actuators for every DOF simplifies maintenance and repairs. E.g. at the SpaceBot Camp one of the shoulder actuators failed shortly before our run. A possibility could have been to repair the vital

(a)

(b)

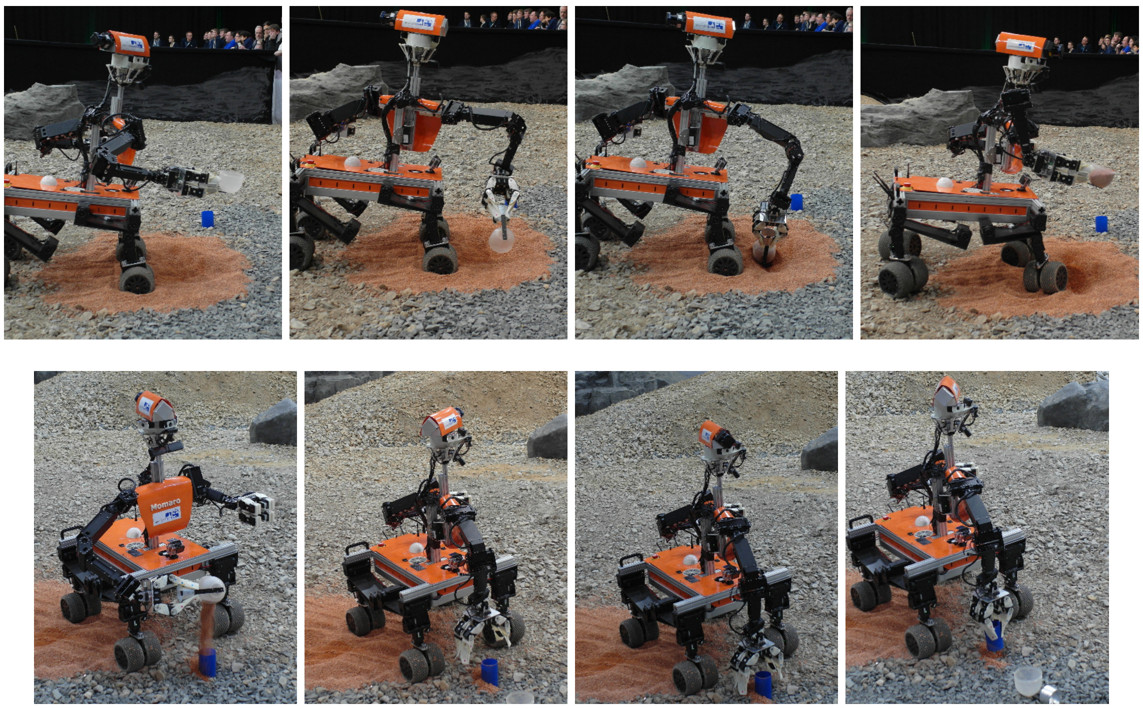


Figure 2. Manipulation capabilities. (a) Momaro is using a scoop to take a soil sample. (b) After filling the blue cup with previously scooped soil, Momaro discards the scoop and grasps the cup to deliver it to a base station.

shoulder using a knee actuator, since the knees were hardly used in this demonstration. Fortunately we acquired a spare actuator in time. Details can be found in Section 11.

3.1.3 Simplicity

Since state-of-the-art approaches for bipedal locomotion on rough terrain are prone to falls and current generation robots are mostly not able to recover after these falls by themselves, we decided to equip Momaro with a total of four legs to minimize the probability of falling. As robot locomotion using stepping is comparably slow, the legs end in pairs of steerable wheels. The legs have three pitch joints in hip, knee and ankle, allowing the adjustment of the wheel pair position relative to the trunk in the sagittal plane. Furthermore, the ankle can rotate around the yaw axis and the two wheels can be driven independently. This allows the robot to drive omnidirectionally on suitable terrain, while also stepping over obstacles too high to drive over. In other words, the additional complexity of stepping is only used when the terrain requires it.

3.1.4 Low Weight

Momaro is relatively lightweight (58 kg) and compact (base footprint 80 cm$\times$70 cm), which means that it can be carried comfortably by two people, compared to larger crews and equipment like gantries needed to carry other robots of comparable size. Since the legs and upper body can be detached, the robot can be transported in standard suitcases.

Momaro mostly consist of carbon fiber and aluminum, e.g. the leg segments are carbon fiber springs, thus providing passive adaptation to terrain. The forelegs can extend by 40 cm from the lowest to the highest configuration. The hind legs are 15 cm longer to allow the robot to climb steeper inclines. The wheels are soft foam-filled rubber wheels, which provide ample traction. Their radius of 8 cm and the flexible suspension formed by the carbon fiber springs allows the robot to ignore most obstacles lower than approximately 5 cm.

3.2 Sensing

Momaro's main sensor for environmental perception is a 3D rotating laser scanner on its sensor head (see Fig. 3). It consists of a Robotis Dynamixel MX-64 actuator, which rotates a Hokuyo UTM-30LX-EW laser scanner around the vertical axis.

A PIXHAWK IMU is mounted close to the laser scanner, which is used for motion compensation during scan aggregation and state estimation.

Three 1080p color cameras are also attached to the sensor head for a panoramic view of the environment in front of the robot and a top-down wide angle camera is used to observe the movement of the arms of the robot and its interaction with the environment. Each hand is equipped with a camera which is located between its fingers. For object detection, Momaro features an ASUS Xtion Pro Live RGB-D camera. Finally, the robot also carries a downward-facing wide-angle camera under its base which allows the operators to monitor the wheels and the surface beneath Momaro.

3.3 Electronics

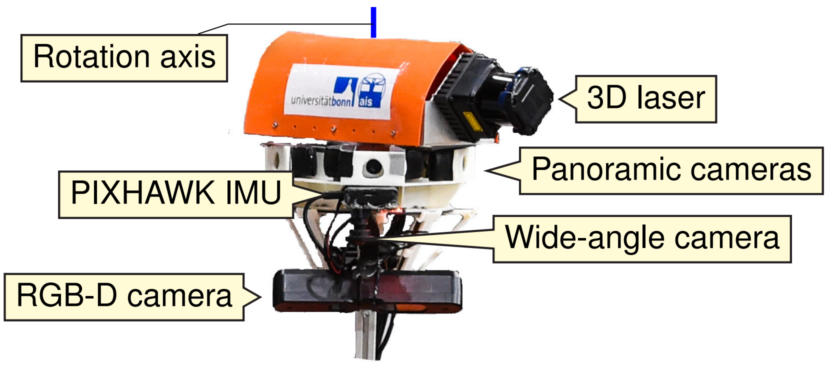
Figure 3 gives an overview of the electrical components of Momaro. In its base, Momaro carries an onboard computer with a fast CPU (Intel Core i7-4790K @4–4.4 GHz) and 32 GB RAM. For communication with the ground station, it is equipped with a NETGEAR Nighthawk AC1900 WiFi router, which allows 2.4 GHz and 5 GHz transmission with up to 1300 Mbit/s.

We make use of a total of six (one for each leg and arm) Crumb2560 microcontroller boards, which bridge high-level USB commands from the computer to low-level servo control commands and vice versa. Performance of the joint actuators is continuously monitored. Feedback information includes measured position, applied torque, and actuator temperature. The boards also provide the capability for low-level emergency stops.

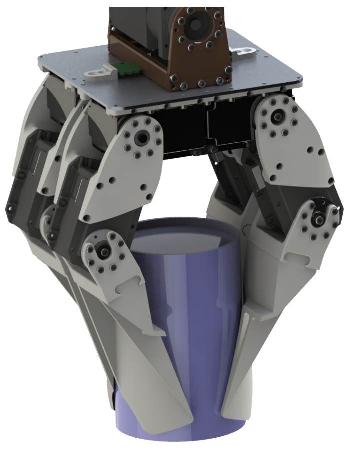
In case of undesirable actions or emergencies, Momaro can be emergency-stopped through two emergency stop switches. One is mounted on the base of the robot for easy access during development, the other one a wireless E-Stop system. The E-stops are connected to the actuator control microcontrollers. If the robot is E-stopped, it stops all currently active servo commands.

Like the microcontroller boards, all cameras, the servo for rotation of the laser, and the PIXHAWK IMU are connected via USB 2.0 for a total of 16 USB devices. The laser scanner is connected via 100 Mbit/s LAN through a slip ring.

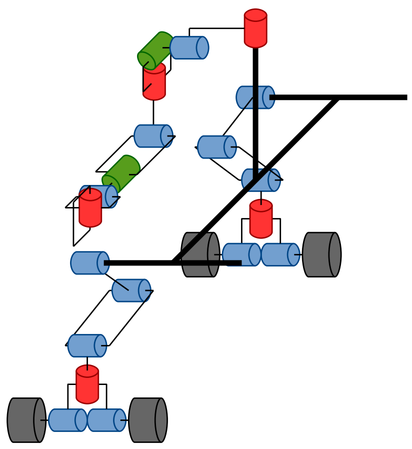
Power is supplied to the robot by a six-cell 355 Wh LiPo battery with 16 Ah capacity at 22.2 V nominal voltage, which yields around 1.5–2 h run time, depending on the performed tasks. Batteries are hot-swappable and thus can be easily exchanged while running. For comfortable development and debugging, they can also be substituted by a power supply.



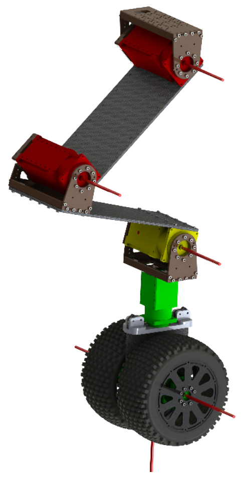
(a) Sensor head



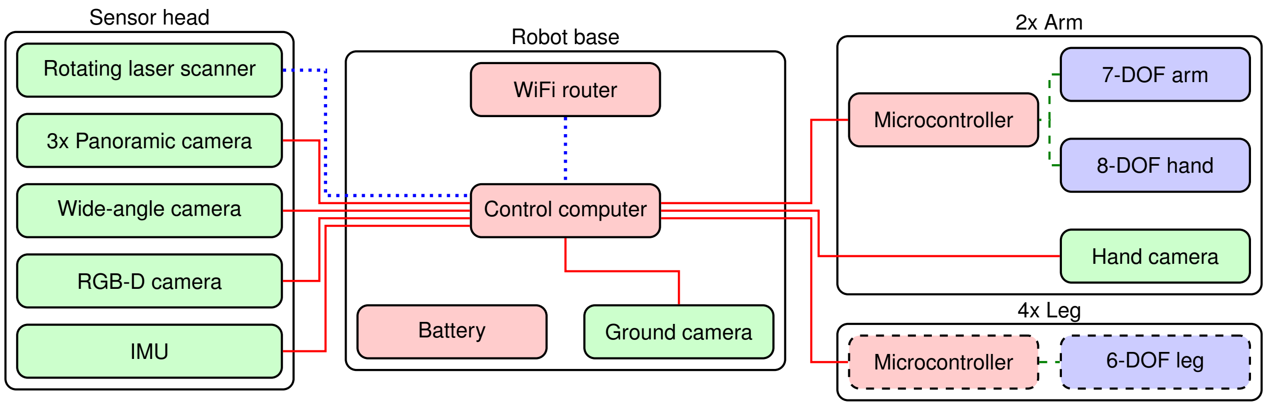
(b) 8-DOF hand



(c) Kinematic tree



(d) 6-DOF leg



(e) Simplified eletrical schematic

Figure 3. Hardware components. (a) Sensor head carrying 3D laser scanner, IMU, four cameras and an RGB-D camera. (b) The 8-DOF hand has specialized fingers for grasping the objects. (c) Kinematic tree of Momaro. For clarity, the figure only shows half of the robot and does not show the hand with its additional eight DOF. Proportions are not to scale. (d) CAD rendering of the front left leg. The six joint axes in hip, knee, ankle pitch, ankle yaw, and wheels are marked with red lines. (e) Simplified overview of the electrical components of Momaro. Sensors are colored green, actuators blue, and other components red. We show USB 2.0 data connections (red), LAN connections (dotted, blue), and the low-level servo bus system (dashed, green).

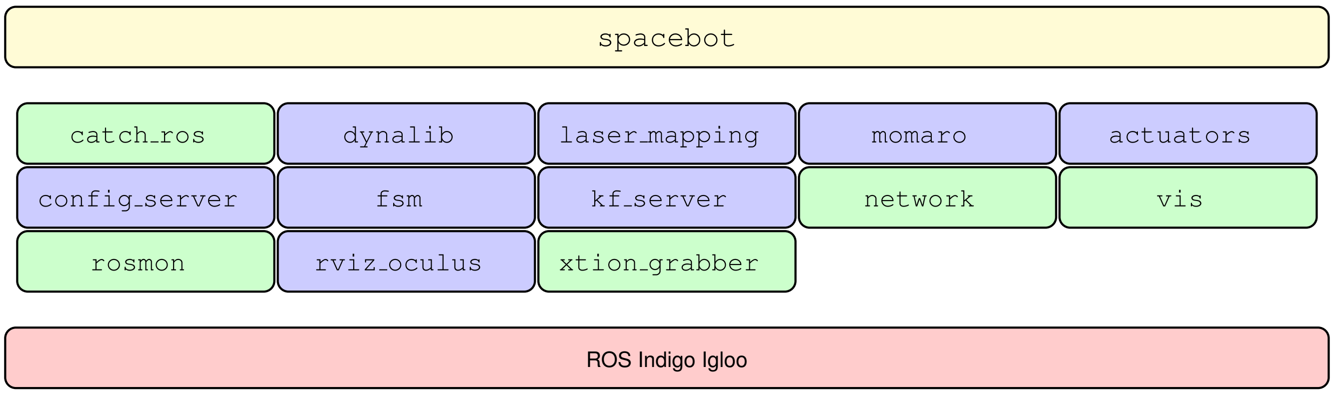
4 SOFTWARE ARCHITECTURE

Both the Momaro robot and the scenarios we are interested in require highly sophisticated software. To retain modularity and maintainability and encourage code re-use, we built our software on top of the popular ROS (Robot Operating System, Quigley et al. (2009)) middleware. ROS provides isolation of software components into separate processes (nodes) and inter- and intraprocess communication via a publisher/subscriber scheme. ROS has seen widespread adoption in the robotics community and has a large collection of freely available open-source packages.

To support the multitude of robots and applications in our group[2], we have a set of common modules, implemented as Git repositories. These modules (blue and green in Fig. 4) are used across projects as needed. On top of the shared modules, we have a repository for the specific application (e.g. DLR SpaceBot Camp 2015, yellow in Fig. 4), containing all configuration and code required exclusively by this application. The collection of repositories is managed by the `wstool` ROS utility.

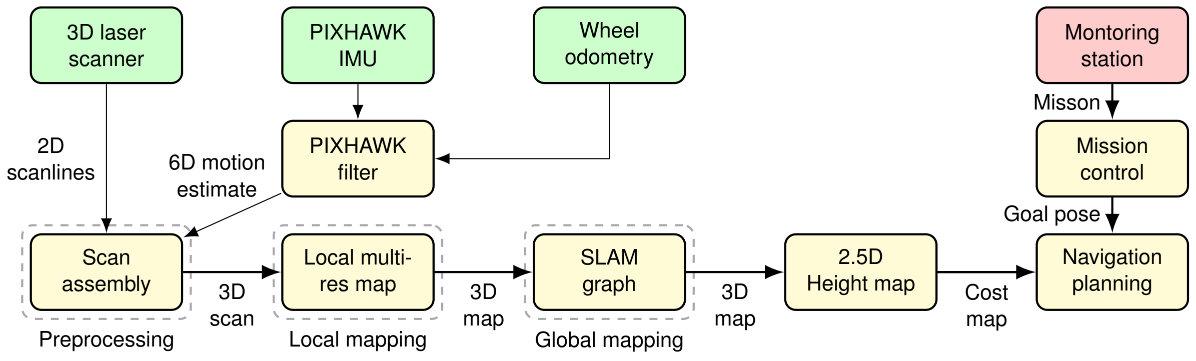
Protection against unintended regressions during the development process is best gained through unit tests. The project-specific code is hard to test, though, since it is very volatile on one hand, and testing would often require full-scale integration tests using a simulator. This kind of integration tests have not been developed yet. In contrast, the core modules are very stable and can be augmented easily with unit tests. Unit tests in all repositories are executed nightly on a Jenkins server, which builds the entire workspace from scratch, gathers any compilation errors and warnings, and reports test results.

[2] `http://ais.uni-bonn.de/research.html`

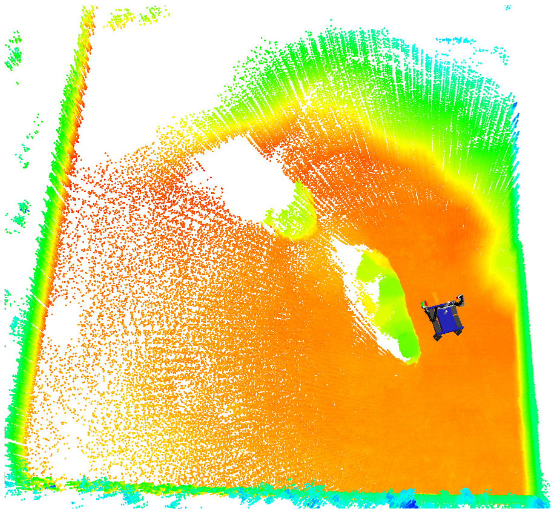


Package	Description
`fsm`	Finite state machine library
`kf_server`	Keyframe editing and interpolation, see Section 8
`laser_mapping`	Laser scanner SLAM using Multi-resolution Surfel Maps, see Section 5
`momaro`	Hardware support for the Momaro robot
`network`	Robust network transport for ROS, see Section 9
`robotcontrol`	Plugin-based real-time robot control node
`rosmon`	ROS process monitoring
`rviz_oculus`	Oculus Rift integration for RViz

Figure 4. Organization of software modules. At the base, the ROS middleware is used. The blue colored boxes correspond to software modules, shared across robots, projects and competitions. Finally, the spacebot module contains software, specific to the SpaceBot Camp. Modules colored in green have been released as open source, see `https://github.com/AIS-Bonn`.



(a) Overview of the SLAM and navigation pipeline


(b) 3D point cloud

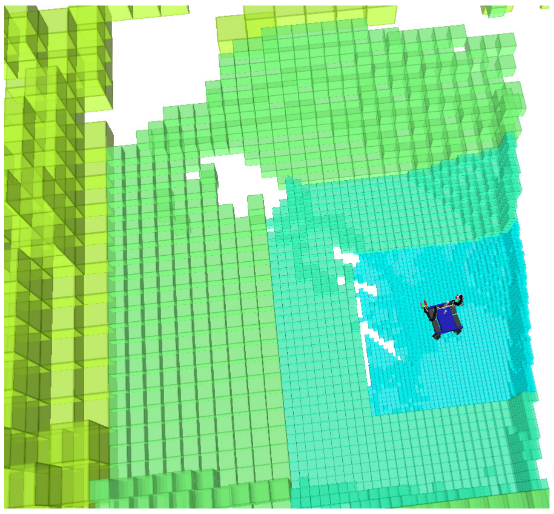

(c) Local multiresolution map

Figure 5. SLAM and navigation architecture. (a) Overview of our mapping, localization and navigation system. The measurements are processed in preprocessing steps, described in Section 5.1. The resulting 3D point cloud is used to estimate the transformation between the current scan and the map. Registered scans are stored in a local multiresolution map (Section 5.2). Keyframe views of local maps are registered against each other in a SLAM graph (Section 5.3). A 2.5D height map is used to assess drivability. A standard 2D grid-based approach is used for planning (Section 6). (b) 3D points stored in the map on the robot. Color encodes height from ground. (c) The grid-based local multiresolution map. Cell size (indicated by color) increases with the distance from the robot.

5 MAPPING AND LOCALIZATION

For autonomous navigation during a mission, our system continuously builds a map of the environment and localizes within this map. To this end, 3D scans of the environment are aggregated in a robot-centric local multiresolution map. The 6D sensor motion is estimated by registering the 3D scan to the map using our efficient surfel-based registration method (Droeschel et al., 2014a). In order to obtain an allocentric map of the environment—and to localize in it—individual local maps are aligned to each other using the same surfel-based registration method. A pose graph that connects the maps of neighboring key poses is optimized globally. The architecture of our perception and mapping system is outlined in Fig. 5.

5.1 Preprocessing and 3D Scan Assembly

The raw measurements from the laser scanner are subject to spurious measurements at occluded transitions between two objects. These so-called *jump edges* are filtered by comparing the angle of neighboring measurements. After filtering for jump edges, we assemble a 3D scan from the 2D scans of a complete rotation of the scanner. Since the sensor is moving during acquisition, we undistort the individual 2D scans in two steps.

First, measurements of individual 2D scans are undistorted with regards to the rotation of the 2D laser scanner around the sensor rotation axis. Using spherical linear interpolation, the rotation between the acquisition of two scan lines is distributed over the measurements.

Second, the motion of the robot during acquisition of a full 3D scan is compensated. Due to Momaro's flexible legs, it is not sufficient to simply use wheel odometry to compensate for the robot motion. Instead we estimate the full 6D state with the PIXHAWK IMU attached to Momaro's head. Here we calculate a 3D attitude estimate from accelerometers and gyroscopes to compensate for rotational motions of the robot. Afterwards, we filter the wheel odometry with measured linear acceleration to compensate for linear motions. The resulting 6D state estimate includes otherwise unobservable motions due to external forces like rough terrain, contacts with the environment, wind, etc. It is used to assemble the individual 2D scans of each rotation to a 3D scan.

5.2 Local Mapping

Distance measurements from the laser-range sensor are accumulated in a 3D multiresolution map with increasing cell sizes from the robot center. The representation consists of multiple robot-centered 3D grid-maps with different resolutions. On the finest resolution, we use a cell length of 0.25 m. Each grid-map is embedded in the next level with coarser resolution and doubled cell length. The stored points and grid structure are shown in Fig. 5.

We use a hybrid representation, storing 3D point measurements along with occupancy information in each cell. Similar to Hornung et al. (2013) we use a beam-based inverse sensor model and ray-casting to update the occupancy of a cell. For every measurement in the 3D scan, we update the occupancy information of cells on the ray between the sensor origin and the endpoint. Point measurements of consecutive 3D scans are stored in fixed-sized circular buffers, allowing for point-based data processing and facilitating efficient nearest-neighbor queries.

After a full rotation of the laser, the newly acquired 3D scan is registered to the so far accumulated map to compensate for drift of the estimated motion. For aligning a 3D scan to the map, we use our surfel-based registration method (Droeschel et al., 2014a)—designed for this data structure, it leverages the multiresolution property of the map and gains efficiency by summarizing 3D points to surfels that are used for registration. Measurements from the aligned 3D scan replace older measurements in the map and are used to update the occupancy information.

5.3 Allocentric Mapping

To estimate the motion of the robot, we incorporate IMU measurements, wheel odometry measurements and the the local registration results. While these estimates allow us to control the robot and to track its pose over a short period of time, they are prone to drift and thus are not suitable for localization on a larger time scale. Furthermore, they do not provide a fixed allocentric frame for the definition of mission-relevant poses. Thus, we build an allocentric map by means of laser-based SLAM and localize towards this map during autonomous operation.

This allocentric map is built by aligning multiple local multiresolution maps, acquired from different view poses (Droeschel et al., 2014b). We model different view poses as nodes in a graph that are connected by edges. A node consists of the local multiresolution map from the corresponding view pose. Each edge in the graph models a spatial constraint between two nodes.

After adding a new 3D scan to the local multiresolution map as described in Section 5.2, the local map is registered towards the previous node in the graph using the same registration method. A new node is generated for the current local map, if the robot moved sufficiently far. The estimated transformation between a new node and the previous node models a spatial constraint and is maintained as the value of the respective edge in our pose graph. In addition to edges between the previous node and the current node, we add spatial constraints between close-by view poses that are not in temporal sequence.

From the graph of spatial constraints, we infer the probability of the trajectory estimate given all relative pose observations. Each spatial constraint is a normally distributed estimate with mean and covariance. This pose graph optimization is efficiently solved using the g^{2}o framework (Kuemmerle et al., 2011), yielding maximum likelihood estimates of the view poses.

5.4 Localization

While traversing the environment, the pose graph is extended and optimized whenever the robot explores previously unseen terrain. We localize towards this pose graph during mission to estimate the pose of the robot in an allocentric frame. When executing a mission, e.g., during the SpaceBot Camp, the robot traverses goal poses w.r.t. this allocentric frame.

Since the laser scanner acquires complete 3D scans with a relatively low rate, we incorporate the egomotion estimate from the wheel odometry and measurements from the IMU to track the pose of the robot. The egomotion estimate is used as a prior for the motion between two consecutive 3D scans. In detail, we track the pose hypothesis by alternating the prediction of the robot movement given the filter result and alignment of the current local multiresolution map towards the allocentric map of the environment.

The allocentric localization is triggered after acquiring a 3D scan and adding it to the local multiresolution map. Due to the density of the local map, we gain robustness. We update the allocentric robot pose with the resulting registration transform. To achieve real-time performance of the localization module, we track only one pose hypothesis.

During the SpaceBot Camp, we assumed that the initial pose of the robot was known, either by starting from a predefined pose or by means of manually aligning our allocentric coordinate frame with a coarse height map of the environment. Thus, we could navigate to goal poses in the coarse height map by localizing towards our pose graph.

5.5 Height Mapping

As a basis for assessing drivability, the 3D map is projected into a 2.5D height map, shown in Fig. 6. In case multiple measurements are projected into the same cell, we use the measurement with median height. Gaps in the height map (cells without measurements) are filled with are local weighted mean if the cell has at least two neighbors within a distance threshold (20 cm in our experiments). This provides a good approximation of occluded terrain until the robot is close enough to actually observe it. After filling gaps in the height map, the height values are spatially filtered using the fast median filter approximation using

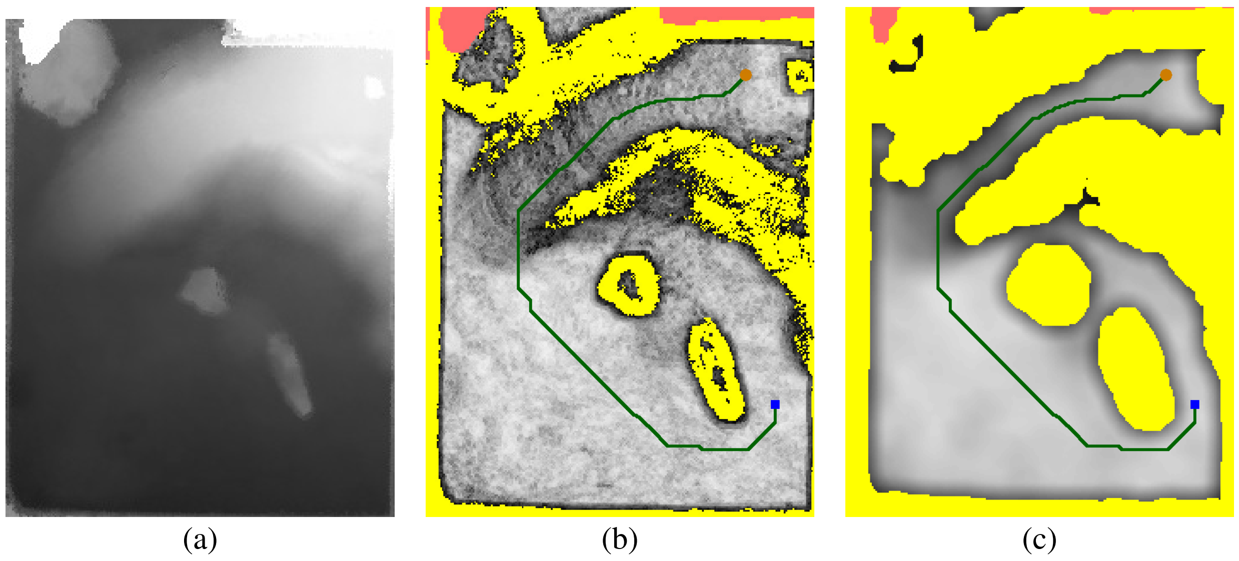


Figure 6. Navigation planning. (a) 2.5D height map generated by projecting the 3D map. (b) Calculated traversability costs for each cell. (c) Inflated costs used for A* path planning. The orange dot represents the current robot position, the blue square the target position. Yellow regions represent absolute obstacles, red regions indicate missing measurements.

local histograms (Huang et al., 1979). The resulting height map is suitable for navigation planning (see Section 6).

6 NAVIGATION

Our autonomous navigation solution consists of two layers: The global path planning layer and the local trajectory planning layer. Both planners are fed with cost maps calculated from the aggregated laser measurements.

6.1 Local Height Difference Maps

Since caves and other overhanging structures are the exception on most planetary surfaces, the 2.5D height map generated in Section 5.5 suffices for autonomous navigation planning.

The 2.5D height map H is transformed into a multi scale height difference map. For each cell (x, y) in the horizontal plane, we calculate local height differences D_l at multiple scales l. We compute $D_l(x, y)$ as the maximum difference to the center cell (x, y) in a local l-window:

$$D_l(x,y) := \max_{\substack{|u-x|<l; u\neq x \\ |v-y|<l; v\neq y}} |H(x,y) - H(u,v)| \,. \tag{1}$$

$H(u, v)$ values of NaN are ignored. In the cases where the center cell $H(x, y)$ itself is not defined, or there are no other defined $l-$neighbors, we assign $D_l(x, y)$:=NaN. Small, but sharp obstacles show up on the D_l maps with lower l scales. Larger inclines, which might be better to avoid, can be seen on the maps with a higher l value.

6.2 Path Planning

During the SpaceBot Camp, we used the standard ROS `navfn`[3] planner. Afterwards, we replaced it with a custom A* planner to consider gradual costs fully, which the ROS planner was not designed to do. We transform the height difference map into a cost map that can be used for path planning.

A combined difference map, $\widetilde{D}$ is generated by linear combination of different D_l maps to comprise information about smaller obstacles and larger inclines. The summands from the D_3 and D_6 maps are constrained to a response of $\frac{1}{2}$ to prevent the creation of absolute obstacles from a single scale alone. The smallest scale D_1 is allowed to create absolute obstacles, since sharp obstacles pose great danger to the robot:

$$\widetilde{D}(x,y) := \sum_{l\in\{1,3,6\}} \begin{cases} \lambda_l D_l & \text{if } l = 1 \\ min\{0.5; \lambda_l D_l\} & \text{otherwise.} \end{cases} \tag{2}$$

The λ_1, λ_3, and λ_6 parameter values for drivability computation were empirically determined as 2.2, 3.6, and 2.5 respectively.

6.2.1 Global Path Planning

For global path planning, we implemented an A* graph search on the 2D grid map. The Euclidean distance (multiplied with the minimum cost in the grid map) is used as the heuristic function for A*. This planning does not account for the robot foot print and considers the robot as just a point in the 2D grid. To ensure the generation of a safe path, we inflate obstacles in the cost map to account for the risk closer to obstacles. The inflation is done in two steps. The cells within the distance of robot radius from absolute obstacles are elevated to absolute obstacle cost, yielding cost map $\bar{D}$. Then for all other cells, we calculate local averages to produce costs D_D that increase gradually close to obstacles:

$$P(x,y) := \{(u,v) : (x-u)^2 + (y-v)^2 < r^2\}, \tag{3}$$

$$D_D(x,y) := \begin{cases} 1 & \text{if } \bar{D}(x,y) = 1 \\ \sum\limits_{(u,v)\in P(x,y)} \frac{\bar{D}(x,y)}{|P(x,y)|} & \text{otherwise.} \end{cases} \tag{4}$$

Figure 6 shows a planned path on the height map acquired during our mission at the SpaceBot Camp.

6.2.2 Local Trajectory Rollout

The found global path needs to be executed on a local scale. To this end, we use the standard ROS `dwa_local_planner`[4] package, which is based on the Dynamic Window Approach (Fox et al. (1997)). The `dwa_local_planner` accounts for the robot foot print, so cost inflation is not needed.

In order to prevent oscillations due to imperfect execution of the planned trajectories, we made some modifications to the planner. The `dwa_local_planner` plans trajectories to reach the given goal pose (x, y, θ) first in 2D (x, y) and then rotates in-place to reach θ (this is called "latching" behavior). Separate cartesian and angular tolerances determine when the planner starts turning and when it reports navigation success. We modified the planner to keep the current "latching" state even when a new global plan is

[3] http://wiki.ros.org/navfn

[4] http://wiki.ros.org/dwa_local_planner

received (every 4 s), as long as the goal pose does not change significantly. We also wrote a simple custom recovery behavior that first warns the operator crew that the robot is stuck and then executes a fixed driving primitive after a timeout.

6.3 Omnidirectional Driving

The wheel positions $\mathbf{r}^{(i)}$ relative to the trunk determine the footprint of the robot, but also the orientation and height of the robot trunk. During autonomous operation, the wheel positions are kept in a configuration with a base height.

Either autonomous navigation or manual operator input generates a velocity command $\mathbf{w} = (v_x, v_y, \omega)$ with horizontal linear velocity $\mathbf{v}$ and rotational velocity ω around the vertical axis. The velocity command is first transformed into the local velocity at each wheel i:

$$\begin{pmatrix} v_x^{(i)} \\ v_y^{(i)} \\ v_z^{(i)} \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix} \times \mathbf{r}^{(i)} + \dot{\mathbf{r}}^{(i)}, \tag{5}$$

where $\mathbf{r}^{(i)}$ is the current position of wheel i relative to the base. The kinematic velocity component $\dot{\mathbf{r}}^{(i)}$ allows simultaneous leg movement while driving. The wheels rotates to yaw angle $\alpha^{(i)} = \text{atan2}(v_y^{(i)}, v_x^{(i)})$ first and then moves with the velocity $||(v_y^{(i)}, v_x^{(i)})^T||$. While driving, the robot continuously adjusts the orientation of the ankle, using IMU information to keep the axis vertical and thus retains omnidirectional driving capability.

6.4 Base Orientation Control

To prevent the robot from pitching over on the high-incline areas in the arena, we implemented a pitch control mechanism. The pitch angle of the robot is continuously measured using the IMU. We then use a simple proportional controller to compensate for the disturbance. With the commanded angle w, disturbance z, controller gain K_p, plant gain K_s and plant disturbance gain K_{sz}, the steady state error e_b of the linearized proportional plant evolves with

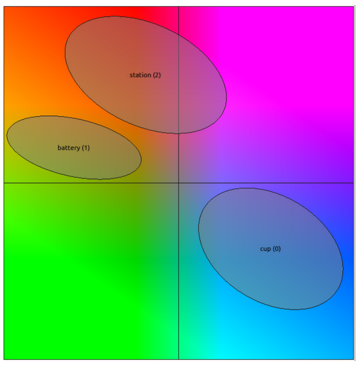
$$e_b = \frac{1}{1 + K_s \cdot K_p} \cdot w - \frac{K_{sz}}{1 + K_s \cdot K_p} \cdot z. \tag{6}$$

Since the incline is directly measured, $K_s = 1$ and $K_{sz} = 1$. We found $K_p = 0.8$ to sufficiently stabilize for inclines present at the SpaceBot Camp. When driving up the ramp with $z \approx 15°$, and setpoint $w = 0°$ the resulting error (robot pitch) is $e_b \approx 8.3°$.

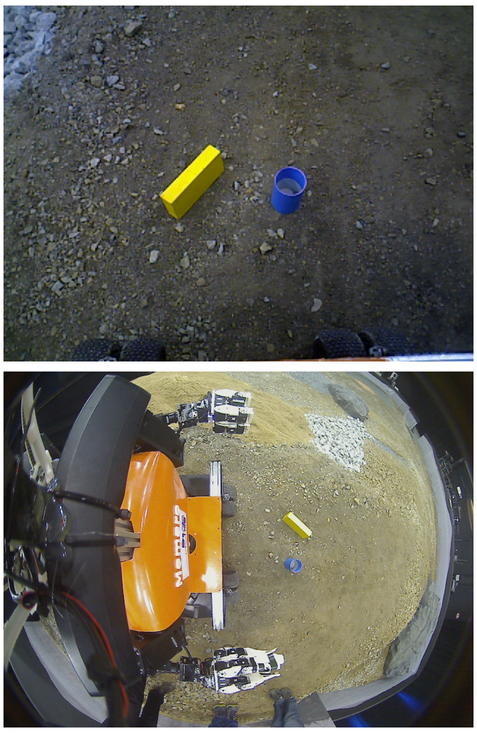
We found that this compensation enables Momaro to even overcome inclines greater than $20°$ without pitching over. Due to the lack of integral control, the robot is even ($e_b = 0°$) only on a completely flat surface. Since this poses no balance problem, there is no need for integral control.

7 OBJECT PERCEPTION

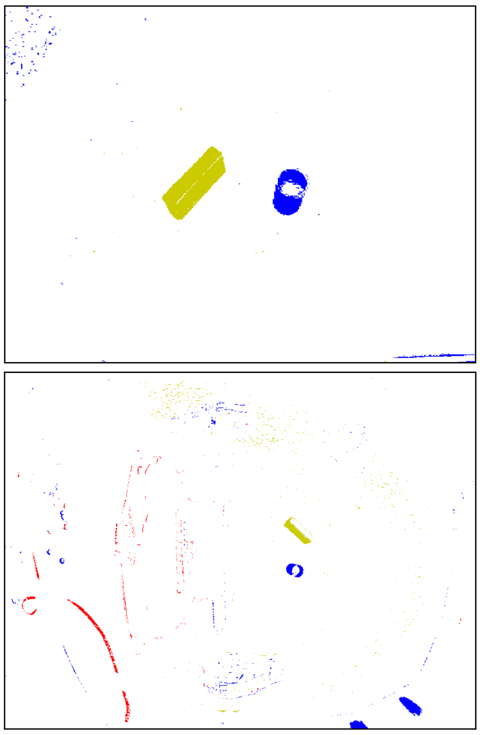
For approaching objects and adapting motion primitives to detected objects, RGB images and RGB-D point clouds from the wide-angle camera and ASUS Xtion camera, mounted on the sensor head are used. We differentiate between object detection (i.e. determining an approximate object position) and object registration (i.e. determining the object pose accurately).



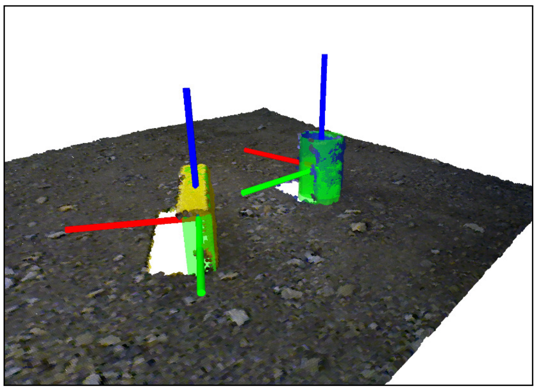
(a) YUV space



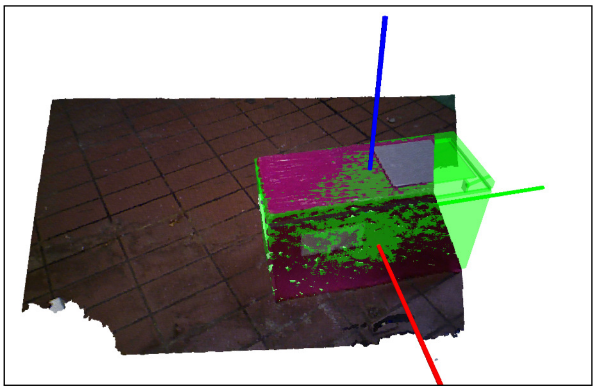
(b) RGB input image



(c) Classification result



(d) Registration of battery and cup



(e) Registration of the base station

Figure 7. Object perception. (a) Classification ellipses in UV space. (b) RGB input image (first row: Xtion camera, second row: RGB wide-angle camera). (c) Pixel classes (white = unknown). (d) RGB-D point cloud showing the cup and battery objects on SpaceBot Camp terrain. The registered models are shown in green. (e) Registration of the base station. Although neither the left nor the right face is visible, the pose ambiguity is resolved correctly.

The objects provided by DLR are color-coded. We classify each pixel by using a precomputed lookup table in YUV space. The lookup table is generated from a collection of ellipses for each color class in UV space (see Fig. 7), and lower/upper limits in brightness (Y). Thus, we assume that the object color measurements are governed by a gaussian mixture model in the UV plane. In practice, a single ellipse sufficed for each of the SpaceBot Camp objects.

When approaching an object, object detection is initially performed with the downwards-facing wide-angle camera mounted on the sensor head (see Fig. 7). Using the connected component algorithm, we obtain object candidate clusters of same-colored pixels. An approximate pinhole camera model calculates the view ray for each cluster. Finally, the object position is approximated by the intersection of the view ray with the local ground plane. The calculated object position is precise enough to allow approaching the object until it is in the range of other sensors.

As soon as the object is in range of the head-mounted ASUS Xtion camera, the connected component algorithm can also take Cartesian distance into account. We use the PCL implementation of the connected component algorithm for organized point clouds. Since the depth measurements allow us to directly compute the cluster centroid position, and the camera is easier to calibrate, we can approach objects much more precisely using the RGB-D camera.

When the object is close enough, we use registration of a CAD model to obtain a precise object pose (see Fig. 7). Since color segmentation often misses important points of the objects, we perform a depth-based plane segmentation using RANSAC and Euclidean clustering as detailed by Holz et al. (2012) to obtain object clusters. The clusters are then registered using Generalized ICP (Segal et al., 2009).

ICP approaches often have problems with partially observed box shapes. For example, only the front and the top face of a box may be visible if the box is partially outside of the camera view frustum. To resolve the resulting ambiguity, we initialize the ICP pose using PCA under the assumption that the visible border of the object which is close to the image border is not an actual object border but is caused by the camera view frustum. In practice, this problem particularly occurs with the large base station object (see Fig. 7).

The ICP pose is then normalized respecting the symmetry axes/planes of the individual object class. For example, the cup is symmetrical around the Z axis, so the X axis is rotated such that it points in the robot's forward direction (see Fig. 7).

8 MANIPULATION

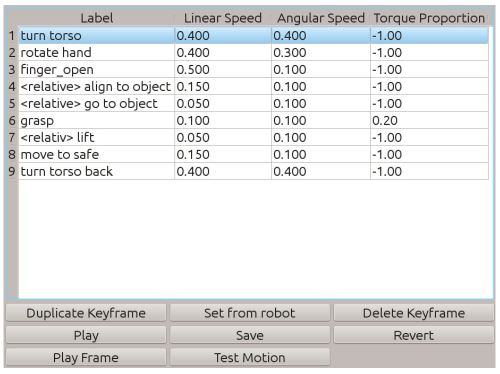
Since Momaro is a unique prototype, the time used for development and testing had to be balanced between individual submodules. To reduce the need for access to the real robot, we made extensive use of simulation tools. For manipulation tasks, we developed a Motion Keyframe Editor GUI to design motion primitives offline. Finished motions are then tested and finalized on the real robot with the original objects to be manipulated in the field. We show the Motion Keyframe Editor GUI in Figure 8. With its help, we designed dedicated motions for all specific tasks in the SpaceBot Camp. We give an overview of our custom motions and their purpose in Table 8.

8.1 Kinematic Control

We use straight-forward kinematic control for Momaro (see Fig. 9). Both arms and the torso yaw joint are considered independently.

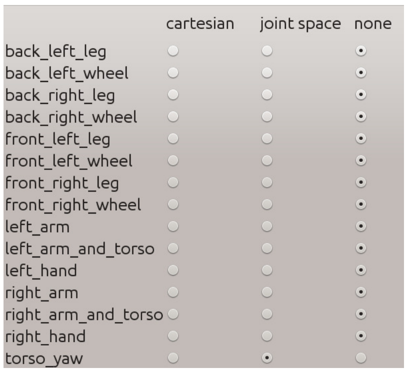
A goal configuration is specified by telemanipulation (see Section 10) or predefined keyframe sequences either in Cartesian or in joint-space. To interpolate between current and goal configuration, the Reflexxes Motion Library (Kröger, 2011) is used. Goals for different limbs can be defined concurrently; the interpolation is configured in a way that goals for all limbs are reached simultaneously. Cartesian poses are converted to joint-space configurations, using inverse kinematics after interpolation. We use the selectively damped least squares approach (SDLS) described by Buss and Kim (2005) to calculate the inverse kinematics of the arms. Before the configurations are sent to the hardware controllers for execution, they are checked for self-collisions using the MoveIt! library[5]. Detecting a collision will abort motion

[5] `http://moveit.ros.org`



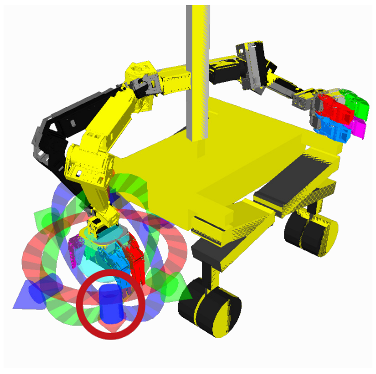
	Label	Linear Speed	Angular Speed	Torque Proportion
1	turn torso	0.400	0.400	-1.00
2	rotate hand	0.400	0.300	-1.00
3	finger_open	0.500	0.100	-1.00
4	<relative> align to object	0.150	0.100	-1.00
5	<relative> go to object	0.050	0.100	-1.00
6	grasp	0.100	0.100	0.20
7	<relativ> lift	0.050	0.100	-1.00
8	move to safe	0.150	0.100	-1.00
9	turn torso back	0.400	0.400	-1.00

(a) Motion schedule



	cartesian	joint space	none
back_left_leg	○	○	●
back_left_wheel	○	○	●
back_right_leg	○	○	●
back_right_wheel	○	○	●
front_left_leg	○	○	●
front_left_wheel	○	○	●
front_right_leg	○	○	●
front_right_wheel	○	○	●
left_arm	○	○	●
left_arm_and_torso	○	○	●
left_hand	○	○	●
right_arm	○	○	●
right_arm_and_torso	○	○	●
right_hand	○	○	●
torso_yaw	○	●	○

(b) Interpolation selection



(c) Motion simulation

(d) Designed motions

Motion	Purpose	Reference type
scoop	fill scoop tool with soil sample	absolute
fill cup	pour soil sample into the cup and discard scoop tool	relative (cup)
grasp cup right hand	grasp cup with right hand from above	relative (cup)
grasp battery left hand	grasp battery with left hand from above	relative (battery)
place cup	place cup on base station	relative (base station)
place battery	put battery into base station	relative (base station)
toggle switch	toggle switch on side of base station	relative (base station)
grasp abort {left,right} arm	motion to initial position when grasp is aborted	absolute
reset {left,right} arm	move all individual joints of the arm in defined position (resolves singularity-induced c-space ambiguities)	absolute
reset torso	move torso into initial position	absolute
cheer	cheer to the audience	absolute

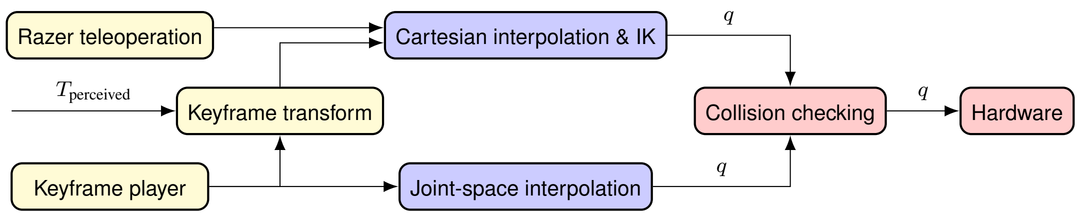
Figure 8. Keyframe Editor GUI. (a) Motions are designed step by step and can be absolute or relative to perceived objects. (b) The user can select which joint groups are included in the currently edited keyframe and if interpolation between keyframes is Cartesian or joint space. (c) The real position of the robot is indicated in black. The currently edited keyframe target is shown in yellow. Interactive markers can be used to modify the keyframe pose in 6D (here only for the right hand). A model of the cup (blue, circled red) is placed in front of the robot to assist designing relative motions.

execution. For safety reasons, different methods of manipulation control (i.e. telemanipulation and the keyframe player) will preempt each other.

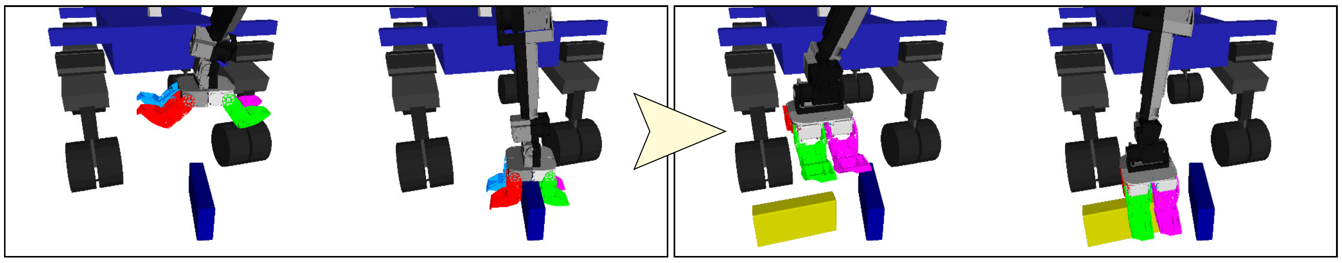
8.2 Motion Adaption

Since it is often impossible or too slow to precisely approach an object in all 6 dimensions, we relax the assumption of absolute positioning. Motions can be designed around a reference object $T_{\text{reference}}$. When the motion is executed, the predefined endeffector pose $T_{\text{endeffector}}$ is transformed in selected keyframes i to match the perceived object $T_{\text{perceived}}$:

$$T_{\text{relative}} = T^{(i)}_{\text{perceived}} \left(T_{\text{reference}}\right)^{-1} T^{(i)}_{\text{endeffector}} \tag{7}$$



(a) Manipulation pipeline



(b) Motion adaption

Figure 9. Object manipulation. (a) Kinematic control architecture for Momaro. Joint configurations can be generated using magnetic trackers or the keyframe player. Cartesian poses in keyframes can be adapted to a measured pose p. The interpolated configurations $T_{\text{perceived}}$ are checked for collisions before they are sent to the hardware. (b) Grasping objects dynamically using motion adaption. Left: The blue reference object is grasped as the primitive was designed in the Keyframe Editor. Right: The primitive is automatically adapted to the perceived pose of the yellow object.

Figure 9 shows how a motion, designed relative to a reference object, is adapted to a perceived object pose to account for imprecise approach of the object.

As described in Section 7, the perceived objects are represented in a canonical form, removing all ambiguities resulting from symmetries in the original objects. For example, the rotation-symmetric cup is always grasped using the same yaw angle. After adaption, the Cartesian keyframes are interpolated as discussed above.

9 COMMUNICATION

Communication between the ground station and a planetary rover is typically very limited – in particular it has high latency due to the speed of light and the large distances involved. The SpaceBot Camp addressed this limitation by imposing several constraints on the network link:

- Packets were delayed by 2 s in each direction, as expected to occur on a lunar mission,
- the uplink from the ground station to the robot could only be opened for 5 min at a time, and
- the 60 min schedule included two 4 min windows where uplink communication was not possible (e.g. due to planetary occlusions).

Furthermore, our system uses a wireless data link inside the arena, which introduces packet loss.

The main idea of our communication system is to minimize latency by exploiting the different characteristics of the local wireless link inside the arena and the simulated inter-planetary network.

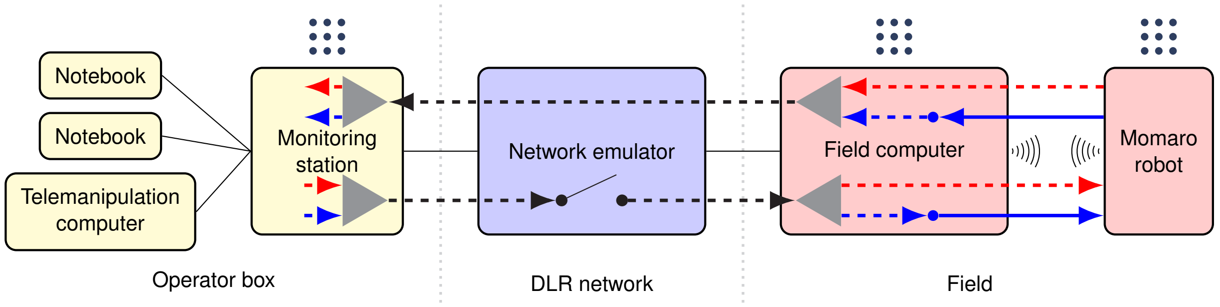


Figure 10. Communication architecture. Components in the vicinity of the operators are shown in yellow, DLR-provided components in blue, components in the “arena”-network in red. Solid black lines represent physical network connections. Thick lines show the different channels, which stream data over the network (dotted: UDP, solid: TCP). The ROS logo (⁝⁝⁝) indicates a ROS master. UDP tunnel endpoints are designated by triangles. Streaming links (Section 9.1.2) are colored red, message links (Section 9.1.3) are shown in blue.

9.1 Communication Architecture

Our communication architecture is shown in Fig. 10. The DLR-provided network emulator is the central element limiting all communication between robot and operator crew. To be able to exploit the different link characteristics, we place an additional field computer between the network emulator and the robot. Thus, it is connected to the network emulator via a reliable ethernet connection, and communicates directly with the robot over WiFi. As the WiFi link is unreliable, but has low latency, while the network emulator link is reliable, but has high latency, this places the field computer in an ideal position to exploit both link characteristics.

As the network emulator allows communication only through a single port per direction, we use the Linux `tun` interface to create a network tunnel over two ports. For UDP tunneling, we adapted code from the `quicktun` project[6]. The tunnel wraps all packets in UDP packets, transmitted over the two designated ports. This allows us to use multiple communication channels without interference.

Since we use the ROS middleware for all software components, separate ROS masters run on the robot, the field computer, and the ground station. Multiple operator computers can be connected to the ROS master running on the ground station to provide additional views and means for intervention.

9.1.1 Communication Software Module

Since our participation in the DLR SpaceBot Cup 2013 (Stückler et al., 2015), our group develops a robust software module (`nimbro_network`) for communication between multiple ROS masters over unreliable and high-latency networks. We used it with very good results in the DLR SpaceBot Cup 2013 and in the DARPA Robotics Challenge (Schwarz et al., 2016). Since the DRC, the module is now freely available[7] under BSD-3 license. It provides transport of ROS topics and services over TCP and UDP protocols. Since it does not need any configuration/discovery handshake, it is ideally suited for situations where the connection drops and recovers unexpectedly. Compared to custom-engineered protocols for specific data types or competitions, this module makes fast development possible, as topics can be added on-the-fly in configuration files, without developing specific transport protocols.

[6] `http://wiki.ucis.nl/QuickTun`

[7] `https://github.com/AIS-Bonn/nimbro_network`

Several specific transports or compression exist, such as a ROS log transport, `tf` snapshotting or H264 video stream compression.

For large messages, a transparent BZip2 compression can be enabled. Automatic rate limiting with configurable upper and lower bounds ensures that bandwidth limits are met.

`nimbro_network` also allows forward error correction (FEC), i.e. augmenting the sent packets with additional packets allowing content recovery from arbitrary subsets of sufficient size of transmitted packets. Depending on the message size, a Reed-Solomon codec (Lacan et al., 2009) or a LDPC-Staircase codec (Roca et al., 2008) is chosen.

Note that in principle ROS offers built-in network transparency. Since this functionality heavily relies on the TCP protocol for topic discovery and subscription, even when the "UDPROS" transport is chosen, this is unsuitable for unreliable and high-latency networks.

9.1.2 Streaming Data

Most high-bandwidth data from the robot is of *streaming* type. The key feature here is that lost messages do not lead to system failures, since new data will be immediately available, replacing the lost messages. In this particular application, it even would not make sense to repeat lost messages because of the high latencies involved. This includes

- video streams from the onboard cameras,
- transform information (TF),
- servo diagnostic information (e.g. temperatures),
- object detections, and
- other visualizations.

In the uplink direction, i.e. commands from the operator crew to the robot, this includes e.g. direct joystick commands.

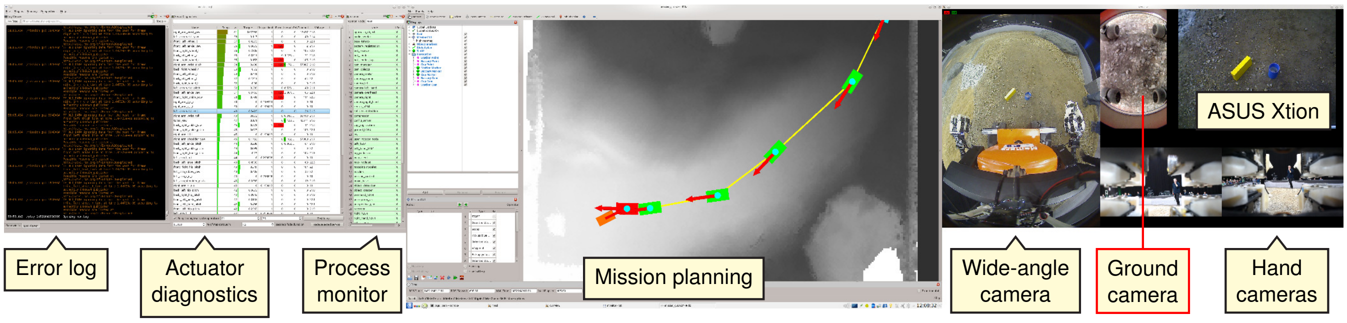
Consequently, we use the `nimbro_network` UDP transport for streaming data (red in Fig. 10). The transport link between robot and field computer uses the FEC capability of `nimbro_network` with 25% additional recovery packets to compensate WiFi packet loss without introducing new latency.

9.1.3 Message Data

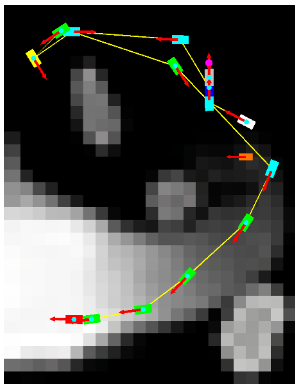
Other data is of the *message* type, including

- Laser pointclouds,
- SLAM maps,
- SLAM transforms,
- ROS action status messages, and
- ROS service calls.

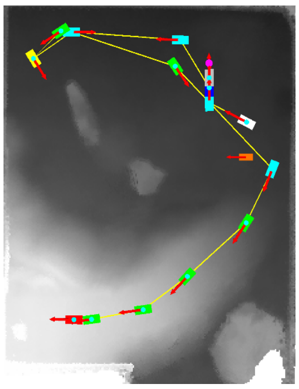
Here, a message loss might be costly (e.g. SLAM maps are only generated on every scanner rotation) or might even lead to system failure (e.g. loss of a ROS action state transition). Therefore, the TCP transport is used for this kind of messages over the WiFi link to eliminate the possibility of packet loss. The link over the network emulator is still implemented with the UDP protocol, since there is no packet loss here and the high latencies prohibit TCP handshakes. The message links are colored blue in Fig. 10.



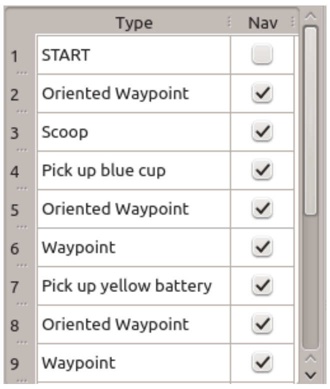
(a) Central operator GUI



(b) Rough height map

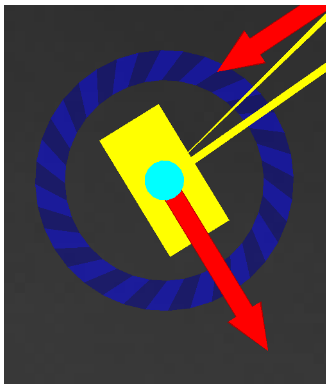


(c) SLAM height map



	Type	Nav
1	START	☐
2	Oriented Waypoint	✓
3	Scoop	✓
4	Pick up blue cup	✓
5	Oriented Waypoint	✓
6	Waypoint	✓
7	Pick up yellow battery	✓
8	Oriented Waypoint	✓
9	Waypoint	✓

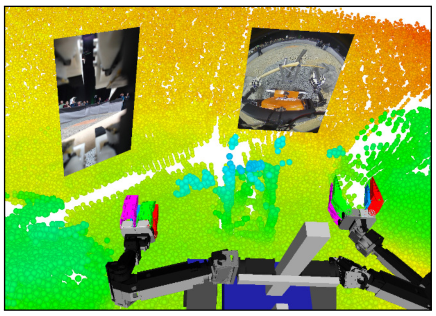
(d) List representation



(e) Pose editing

(f) Oculus Rift

(g) Razer Hydra



(h) Manipulation operator view

Figure 11. Operator interfaces. (a) Overview of the GUI shown on the three lower screens of the main ground station. The left, center and right screens are dedicated to system monitoring and diagnosis, mission planning, and camera images, respectively. (b) Mission plan on rough height map provided by DLR. (c) Mission plan on detailed height map generated from the SLAM map. (d) List representation of the first 8 poses. The "Nav" column can be used to disable navigation (e.g. start grasping an object immediately). (e) Pose editing using interactive marker controls. The position can be modified by dragging the rectangle. The pose is rotated by dragging on the blue circle.
Teleoperation interfaces: Operator uses (f) Oculus Rift DK2 HMD and (g) Razer Hydra 6 DOF controllers for immersive teleoperation. (h) 3rd person view of the scene rendered in the Oculus HMD during debris cleaning (see Fig. 13).

10 MISSION CONTROL INTERFACES

For the operator crew, situational awareness is most important. Our system shows camera images, 3D visualization and diagnosis information on a central ground station with four monitors (see Fig. 11).

In order to cope with the degraded communication link, the system needs to be as autonomous as possible, while retaining the ability to interrupt, reconfigure or replace autonomous behavior by manual intervention. To this end, our system provides three levels of control to the operator crew. On the highest level, entire missions can be specified and executed. The intermediate level allows configuration and triggering of individual autonomous behaviors, such as grasping an object. On the lowest level, the operators can directly control the base velocity using a joystick or move individual DOF of the robot.

The last aspect of our control paradigm is remote debugging. Operators need to be able to directly introspect, debug and manipulate the software on the robot in order to prevent relatively simple problems from escalating to mission failures.

We describe the developed operator interfaces in the following.

10.1 Mission Planning and Execution

Our mission control layer is able to execute all required tasks in the SpaceBot Camp specification. The mission can be specified fully in advance on a rough height map, and can later be interactively refined as the mission progresses and a more detailed map of the environment is created.

A specified mission consists of a list of 2D poses in the height map frame. Attached to each pose is an optional action, which is executed when the robot reaches the pose. Poses without an associated action are just used as navigation targets. Supported actions include:

- Taking a soil sample using the scoop in one hand,
- approaching and grasping the battery,
- approaching the cup, filling it with the soil sample and grasping it, and
- approaching the base station and performing all station manipulation tasks,

The mission can be configured and monitored using our Mission GUI (see Fig. 11). During the mission, execution can be stopped at any time, mission updates can be performed, and the execution resumed. Missions can also be spliced in the sense that the currently performed action is carried out and then execution switches to a new mission.

In the case of a failure of the mission control level, or if the operator judges that the system will not be able to carry out the mission autonomously, the execution can be interrupted and the task in question can be carried out using the lower control levels. Afterwards, the mission can be resumed starting after the completed task.

10.2 Semi-Autonomous Control

The semi-autonomous control level gives direct access to all individual, less autonomous behaviors. This includes

- approaching an object,
- grasping an object,
- performing single manipulation tasks, and
- navigating to a goal pose.

10.3 Low-level Control

If all autonomous behaviors fail, the operators can also directly teleoperate the robot. For manipulation, our operators can choose between on-screen teleoperation using 6D interactive markers in either Cartesian or joint space, or immersive 3D telemanipulation (see Fig. 11) using an Oculus Rift HMD and 6D magnetic trackers (see Rodehutskors et al. (2015) for details).

For navigation, the operator can use a joystick to directly control the base velocity. Teleoperation speed is of course limited by the high feedback latency, so that this method is only used if the navigation planners get stuck. Finally, several macros can be used to influence the robot posture or recover from servo failures such as overheating.

10.4 Remote Introspection and Debugging

To be able to react to software problems or mechanical failures, operators first need to be aware of the problem. Our system addresses this concern by

- providing direct access to the remote ROS log,
- showing the state of all ROS processes, and
- transmitting and displaying 3D visualization data from the autonomous behaviors.

Once aware of the problem, the operators can interact with the system through ROS service calls over our `nimbro_network` solution, parameter changes, or ROS node restarts through `rosmon`. In extreme cases, it is even possible to push small Git code patches over the network and trigger re-compilation on the robot. If everything else fails, the operators can access a remote command shell on the robot using the `mosh` shell (Winstein and Balakrishnan, 2012), which is specifically optimized for high-latency, low-bandwidth situations. The shell gives full access to the underlying Linux operating system.

11 EVALUATION

Momaro has been evaluated in several simulations and lab experiments as well as in the DARPA Robotics Challenge (DRC) Finals in June 2015, during the DLR SpaceBot Cup Qualification in September 2015, and the DLR SpaceBot Camp in November 2015 (Kaupisch et al. (2015)). For details on our performance at the DRC Finals, we refer to Schwarz et al. (2016). Here, we will focus on our performance at the SpaceBot Qualification and Camp.

In preparation for the DLR SpaceBot finals, the SpaceBot Cup Qualification tested basic capabilities of the robotic system. To qualify, participants had to solve three tasks which involved exploration and mapping of an arena and manipulation of the cup and the battery, but no assembly. In contrast to the finals, the communication uplink time was unlimited, which lowered the required autonomy level. Using our intuitive telemanipulation approaches, our team was the only team to successfully qualify in the first attempt. Further information about our performance is available on our website[8]. Since only two other teams managed to qualify using their second attempt, the planned SpaceBot Cup competition was changed to an open demonstration, called the SpaceBot Camp.

The SpaceBot Camp required participants to solve mapping, locomotion, and manipulation tasks in rough terrain. As detailed in Section 1, the battery and cup (with soil sample) had to be found and transported to the base station object, where an assembly task was to be performed. The participants were provided

[8] `http://www.ais.uni-bonn.de/nimbro/Explorer`

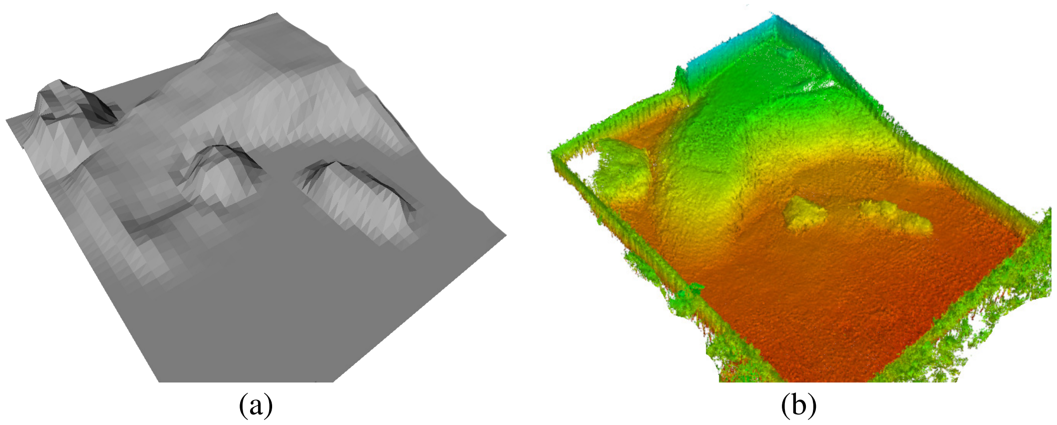


Figure 12. Map refinement. (a) Rough map of the SpaceBot Camp 2015 arena. (b) The resulting global map from data acquired during the competition.

with a coarse map of the environment that had to be refined by the robot's mapping system. As detailed in Section 9, the communication link to the operator crew was severely constrained both in latency (2 s per direction) and in availability.

11.1 Locomotion

While Momaro was mainly evaluated on asphalt at the DRC (Schwarz et al., 2016), the SpaceBot Camp arena included various types of soil and stones (see Fig. 14). We did not experience any problems on the main traversable area, which was covered with flattened soil mixed with stones. During our run, we avoided the gravel and sand areas. We also traversed the soil sample area (loose granulate), and parts of the slopes covered with gravel, as long as the inclination permitted. Testing after our run confirmed that Momaro's wheels were not suited for the fine sand areas on the edge of the ramp, causing the robot to get stuck.

While preparing for the SpaceBot Camp, we learned that our pitch stabilization control method works reliably, even under extreme conditions. Being able to reliably overcome ramps with inclines greater than 20°, we were confident that locomotion would not pose a problem during the competition. Unfortunately, we only employ stabilization in pitch direction. Turning around the yaw axis on a pitched slope can result in a dangerous roll angle. We dealt with this issue during our final run by placing enough waypoints on the primary slope in the course to ensure proper orientation (see Fig. 11).

11.2 Mapping and Self-localization

Our mapping system continuously built an allocentric map of the environment during navigation, guided by waypoints specified on the coarse height map. The coarse map and the allocentric map, generated from our mapping system is shown in Fig. 12. While showing the same structure as the coarse map, the resulting allocentric map is accurate and precisely models the environment. During a mission, the map is used for localization and to assess traversability for navigation. The estimated localization poses are shown in Fig. 14,

Despite the challenging planetary-like environment—causing slip in odometry and vibrations of robot and sensor, our mapping system showed very robust and reliable performance. There was only one situation during the run where the operators had to intervene: Due to traversing the abandoned scoop tool—used to take the soil sample—the robot was exposed to a fast and large motion, resulting in a distorted 3D scan. This distorted 3D scan caused spurious measurements in the map. The operators decided to clear the SLAM

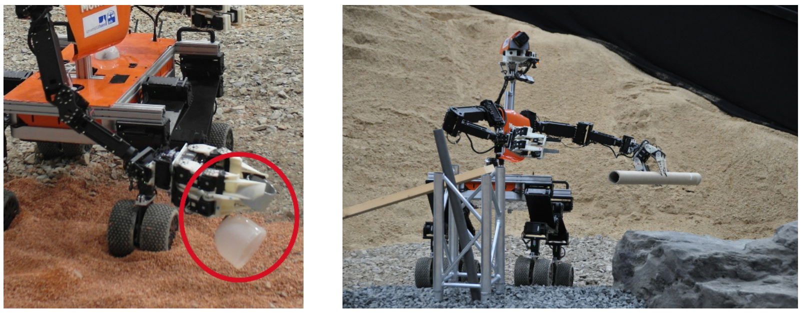

(a) Failed finger actuator (b) Debris cleaning

Figure 13. Details of our run at SpaceBot camp. (a) Due to a failed finger actuator, Momaro failed to take the soil sample in the first attempt. (b) After finishing all tasks of the SpaceBot Camp, we showed Momaro's universal capabilities by removing debris from the terrain under teleoperation.

map using a remote service call to prevent localization failures. The map was rebuilt from this point on and successfully used for the rest of the mission.

11.3 Object Manipulation

While preparing our run, we found the battery slot in the base station to have a significant resistance due to a build-in clamping mechanism. Due to our flexible motion design workflow, we were able to alter the motion so that Momaro would execute small up- and downward motions while pushing to find the best angle to overcome the resistance.

The insertion of the battery requires high precision. To account for inaccuracies in both battery and station pose, we temporarily place the battery on top of the station. After grasping the battery again, we can be sure that any offset in height is compensated.

Furthermore, we found it to be error prone to grasp the battery at the very end, which is necessary to entirely push it inside the slot. Instead, we push the battery in as far as possible until the hand touches the base station. After releasing the battery, we position the closed hand behind it and push it completly inside with part of the wrist and proximal finger segments.

Overall, our straightforward keyframe adaption approach proved itself to be very useful. Compared to motion-planning techniques it lacks collision avoidance and full trajectory optimization, but it is sufficient for the variety of performed tasks.

11.4 Full System Performance at DLR SpaceBot Camp 2015

After a restart caused by a failed actuator (described below), Momaro solved all tasks of the SpaceBot Camp with supervised autonomy. Our team was the only one to demonstrate all tasks including the optional soil sample extraction. Figure 14 gives an overview of the sequence of performed tasks. A video of our performance can be found online[9]. While overall the mission was successful, we experienced a number of problems which will be discussed in detail.

In our run, Momaro failed to take the soil sample in the first attempt. During the vigorous scooping motion, the scoop turned inside the hand (cf. Fig. 2, Fig. 13). We found the problem to be a malfunctioning finger actuator in the hand holding the scoop. Since we were confident that Momaro would be able to solve

[9] `https://youtu.be/q_p5ZO-BKWM`

Table 1. Timings of our run at the DLR SpaceBot Camp 2015.

Task	Start time [mm:ss]	End time [mm:ss]	Duration [mm:ss]
Soil sample collection	1:05	1:40	0:35
Fill and grasp cup	2:15	3:05	0:50
Grasp battery	7:00	7:40	0:40
Base station assembly	18:25	20:25	2:00
Total (including locomotion)	0:00	20:25	**20:25**

all tasks even in the remaining 50:20 minutes, we restarted the whole run after performing a software reset on the affected finger and letting it cool down.

In the second attempt, scooping succeeded and Momaro was able to complete all remaining tasks as well. See Fig. 14 for detailed images of the subtasks. Timings of the run are listed in Table 1.

Although Momaro was able to complete all tasks, this was not possible fully autonomously. While approaching the battery, a timeout aborted the process. This built-in safety-feature made operator interaction necessary to resume the approach. Without intervention, Momaro would have executed the remainder of the mission without the battery object.

As Momaro reached the main slope of the course, we also approached the time of the first communication blackout, because we lost time in the beginning due to the restart. The operator crew decided to stop Momaro at this point, as we knew that going up would be risky and intervention would have been impossible during the blackout. After the blackout, autonomous operation resumed and Momaro successfully went up the ramp to perform the assembly tasks at the base station (Fig. 14). Although the operators paused autonomous navigation at one point on the slope to assess the situation, no intervention was necessary and navigation resumed immediately.

After finishing the course in 20:25 minutes, we used the remaining time to show some of Momaro's advanced manipulation capabilities by removing debris from the terrain with Momaro and our intuitive teleoperation interface (Fig. 13).

12 LESSONS LEARNED

Our successful participation in the SpaceBot Camp was an extremely valuable experience, identifying strong and weak points of our system in a competitive benchmark within the German robotics community. Lessons learned include:

- **Mechanical Design**. While the humanoid torso raised the center of gravity and thus caused stability concerns on high terrain inclines, it allowed us to perform bimanual manipulation. Being able to carry both objects in the hands allowed us to omit storing the objects in separate holders on the robot, saving time. Furthermore, our end effector design allowed us to use a scoop to take the soil sample. The soil extraction task was not attempted by any other team. In future work, we will further improve the robot balance control to operate in more difficult rough terrain. For instance, adaptive roll stabilization could advance Momaro's locomotion capabilities.
- **Actuator Monitoring**. Our system provides extensive diagnostic actuator feedback such as temperature and current consumption. Still, this was not enough to prevent the failure of the finger

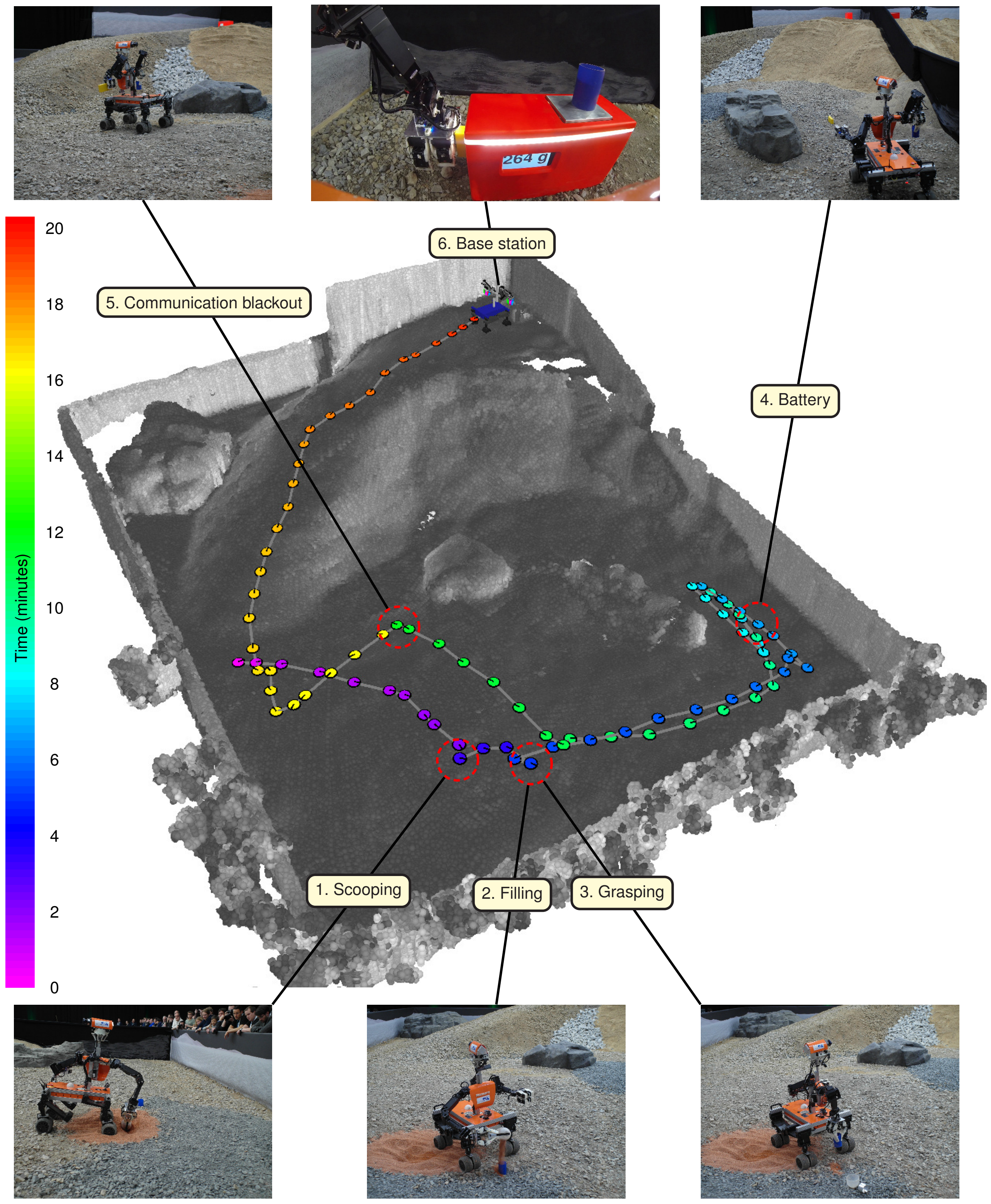


Figure 14. Overview of the executed mission at SpaceBot Camp. The mission starts by scooping the soil sample, filling it into the cup and grasping the cup, then locating and grasping the battery pack. After waiting until the end of scheduled communication blackout, the mission is concluded by Base station assembly.

actuator during our run. Actuator monitoring and damage prevention should have a high priority during development.

- **Software Design: Autonomy Follows Teleoperation**. Our unique history of competing previously in the DARPA Robotics Challenge, a competition heavily focused on intuitive teleoperation, set us apart from other teams. In particular, resulting from the DRC competition, we had extensive intuitive teleoperation abilities before starting work on the higher autonomy required by the SpaceBot Camp. We suspect that most other teams followed the opposite approach, augmenting the autonomy later on with teleoperation facilities, which can be difficult if the system was not designed for teleoperation from the start. Treating the autonomy as an additional layer on a teleoperable system ensures that the operator crew has full control of the system at all time. Furthermore, this also accelerates development, since missing autonomous functionalities can be substituted by intuitive teleoperation. We demonstrated the ability of our telemanipulation solution after our run by removing debris and thus clearing the robot's path.
- **Intelligent Progress Monitoring**. Our mission control layer included some very basic error handling, e.g. fixed timeouts on certain actions. Unfortunately, one of these timeouts resulted in an early abort of the battery approach in our run, which had to be corrected by operator action. A more intelligent system, tracking the progress of the current task, would have noticed that the approach was still progressing and would have continued the approach. In future, we will investigate such resilient progress monitoring methods in more detail.

13 CONCLUSION

In this article, we presented the mobile manipulation robot Momaro and its ground station. We provided details on the soft- and hardware architecture of the integrated robot system and motivate design choices. The feasibility, flexibility, usefulness, and robustness of our design were evaluated with great success at the DLR SpaceBot Camp 2015.

Novelties include an autonomous hybrid mobile base combining wheeled locomotion with active stabilization in combination with fully autonomous object perception and manipulation in rough terrain. For situational awareness, Momaro is equipped with a multitude of sensors such as a continuously rotating 3D laser scanner, IMU, RGB-D camera, and a total of seven color cameras. Although our system was build with comprehensive autonomy in mind, all aspects from direct control to mission specification can be teleoperated through intuitive operator interfaces. Developed for the constraints posed by the SpaceBot Camp, our system also copes well with degraded network communication between the robot and the monitoring station.

The robot localizes by fusing wheel odometry and IMU measurements with pose observations obtained in a SLAM approach using laser scanner data. Autonomous navigation in rough terrain is tackled by planning cost-optimal paths in a 2D map of the environment. High-level autonomous missions are specified as augmented waypoints on the 2.5D height map generated from SLAM data. For object manipulation, the robot detects objects with its RGB-D camera and executes grasps using parametrized motion primitives.

In the future, shared autonomy could be improved by automatic failure detection, such that the robot reports failures and recommends a suitable semi-autonomous control mode for recovery. Currently, only vision-based manipulation is supported by the system. Additional touch and force-torque sensing could potentially lead to more robust manipulation capabilities.

FUNDING

This work was supported by the European Union's Horizon 2020 Programme under Grant Agreement 644839 (CENTAURO) and by Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) under Grant No. SORA1413.

References

Adachi, H., Koyachi, N., Arai, T., Shimiza, A., and Nogami, Y. (1999). Mechanism and control of a leg-wheel hybrid mobile robot. In *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1792 –1797.

Borst, C., Wimbock, T., Schmidt, F., Fuchs, M., Brunner, B., Zacharias, F., Giordano, P. R., Konietschke, R., Sepp, W., Fuchs, S., Rink, C., Albu-Schaffer, A., and Hirzinger, G. (2009). Rollin' Justin - mobile platform with variable base. In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 1597 –1598.

Buss, S. R. and Kim, J.-S. (2005). Selectively damped least squares for inverse kinematics. *Graphics, GPU, and Game Tools*, 10(3):37–49.

Cheng, G. and Zelinsky, A. (2001). Supervised autonomy: A framework for human-robot systems development. *Autonomous Robots*, 10(3):251–266.

Cho, B.-K., Kim, J.-H., and Oh, J.-H. (2011). Online balance controllers for a hopping and running humanoid robot. *Advanced Robotics*, 25(9-10):1209–1225.

Droeschel, D., Stückler, J., and Behnke, S. (2014a). Local multi-resolution representation for 6d motion estimation and mapping with a continuously rotating 3d laser scanner. In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 5221–5226.

Droeschel, D., Stückler, J., and Behnke, S. (2014b). Local multi-resolution surfel grids for mav motion estimation and 3d mapping. In *Proc. of the Int. Conference on Intelligent Autonomous Systems (IAS)*.

Endo, G. and Hirose, S. (2000). Study on roller-walker (multi-mode steering control and self-contained locomotion). In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, volume 3, pages 2808 –2814.

Fox, D., Burgard, W., Thrun, S., et al. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33.

Gillett, R., Greenspan, M., Hartman, L., Dupuis, E., and Terzopoulos, D. (2001). Remote operation with supervised autonomy (rosa). In *Proceedings of the 6th International Conference on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2001)*.

Halme, A., Leppänen, I., Suomela, J., Ylönen, S., and Kettunen, I. (2003). WorkPartner: Interactive human-like service robot for outdoor applications. *Int. Journal of Robotics Research (IJRR)*, 22(7-8):627–640.

Hebert, P., Bajracharya, M., Ma, J., Hudson, N., Aydemir, A., Reid, J., Bergh, C., Borders, J., Frost, M., Hagman, M., et al. (2015). Mobile manipulation and mobility as manipulation—design and algorithms of robosimian. *Journal of Field Robotics (JFR)*, 32(2):255–274.

Heppner, G., Roennau, A., Oberländer, J., Klemm, S., and Dillmann, R. (2015). Laurope – six legged walking robot for planetary exploration participating in the SpaceBot Cup.

Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. (2012). Real-time plane segmentation using RGB-D cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34:189–206.

Huang, T., Yang, G., and Tang, G. (1979). A fast two-dimensional median filtering algorithm. *IEEE Trans. Acoust., Speech, Signal Processing*, 27(1):13–18.

Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., Abeles, P., Stephen, D., Mertins, N., Lesman, A., et al. (2015). Team IHMC's lessons learned from the DARPA robotics challenge trials. *Journal of Field Robotics (JFR)*, 32(2):192–208.

Joyeux, S., Schwendner, J., and Roehr, T. M. (2014). Modular software for an autonomous space rover. In *Proceedings of the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (SAIRAS)*.

Kaupisch, T. and Fleischmann, M. (2015). Mind the robot - rovers leave tracks in the artificial planetary sands. *COUNTDOWN - Topics from the DLR Space Administration*, 31:20–22. `http://www.dlr.de/rd/en/desktopdefault.aspx/tabid-4788/7944_read-45190/`.

Kaupisch, T., Noelke, D., and Arghir, A. (2015). DLR spacebot cup — Germany's space robotics competition. In *Proc. of the Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*.

Kim, M.-S. and Oh, J.-H. (2010). Posture control of a humanoid robot with a compliant ankle joint. *International Journal of Humanoid Robotics*, 07(01):5–29.

Kröger, T. (2011). Opening the door to new sensor-based robot applications—The Reflexxes Motion Libraries. In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*.

Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). G2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*.

Lacan, J., Roca, V., Peltotalo, J., and Peltotalo, S. (2009). Reed-solomon forward error correction (FEC) schemes. Technical report.

Mehling, J., Strawser, P., Bridgwater, L., Verdeyen, W., and Rovekamp, R. (2007). Centaur: NASA's mobile humanoid designed for field work. In *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 2928–2933.

Pedersen, L., Kortenkamp, D., Wettergreen, D., and Nourbakhsh, I. (2003). A survey of space robotics. In *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 19–23.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5.

Raibert, M., Blankespoor, K., Nelson, G., Playter, R., et al. (2008). BigDog, the rough–terrain quadruped robot. In *Proceedings of the 17th World Congress, The International Federation of Automatic Control*, pages 10823–10825, Seoul, Korea.

Roca, V., Neumann, C., and Furodet, D. (2008). Low density parity check (ldpc) staircase and triangle forward error correction (fec) schemes.

Rodehutskors, T., Schwarz, M., and Behnke, S. (2015). Intuitive bimanual telemanipulation under communication restrictions by immersive 3d visualization and motion tracking. In *Proc. of the IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*.

Roennau, A., Kerscher, T., and Dillmann, R. (2010). Design and kinematics of a biologically-inspired leg for a six-legged walking machine. In *3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 626 –631.

Schwarz, M. and Behnke, S. (2014). Local navigation in rough terrain using omnidirectional height. In *Proc. of the German Conference on Robotics (ROBOTIK)*. VDE.

Schwarz, M., Rodehutskors, T., Droeschel, D., Beul, M., Schreiber, M., Araslanov, N., Ivanov, I., Lenz, C., Razlaw, J., Schüller, S., Schwarz, D., Topalidou-Kyniazopoulou, A., and Behnke, S. (2016). NimbRo rescue: Solving disaster-response tasks through mobile manipulation robot Momaro. *Under minor revision for Journal of Field Robotics (JFR), available at* `http://www.ais.uni-bonn.de/papers/JFR_NimbRo_Rescue_Momaro.pdf`.

Schwendner, J., Roehr, T. M., Haase, S., Wirkus, M., Manz, M., Arnold, S., and Machowinski, J. (2014). The artemis rover as an example for model based engineering in space robotics. In *ICRA Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots*.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-ICP. In *Proc. of Robotics: Science and Systems*.

Semini, C., Tsagarakis, N., Guglielmino, E., Focchi, M., Cannella, F., and Caldwell, D. (2011). Design of HyQ–A hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849.

Stentz, A., Herman, H., Kelly, A., Meyhofer, E., Haynes, G. C., Stager, D., Zajac, B., Bagnell, J. A., Brindza, J., Dellin, C., et al. (2015). CHIMP, the CMU highly intelligent mobile platform. *Journal of Field Robotics (JFR)*, 32(2):209–228.

Stückler, J., Schwarz, M., Schadler, M., Topalidou-Kyniazopoulou, A., and Behnke, S. (2015). NimbRo Explorer: Semiautonomous exploration and mobile manipulation in rough terrain. *Journal of Field Robotics (JFR)*.

Sünderhauf, N., Neubert, P., Truschzinski, M., Wunschel, D., Pöschmann, J., Lange, S., and Protzel, P. (2014). Phobos and deimos on mars–two autonomous robots for the dlr spacebot cup. In *Proceedings of the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space-i-SAIRAS'14*. The Canadian Space Agency (CSA-ASC).

Wedler, A., Rebele, B., Reill, J., Suppa, M., Hirschmüller, H., Brand, C., Schuster, M., Vodermayer, B., Gmeiner, H., Maier, A., et al. (2015). LRU - lightweight rover unit. In *Proc. of the 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*.

Winstein, K. and Balakrishnan, H. (2012). Mosh: An interactive remote shell for mobile clients. In *USENIX Annual Technical Conference*, pages 177–182.