Journal of Field Robotics, 33(4): 411-430, Wiley, 2016.

NimbRo Explorer: Semi-Autonomous Exploration and Mobile Manipulation in Rough Terrain

Jörg Stückler
stueckler@ais.uni-bonn.de

Max Schwarz
max.schwarz@uni-bonn.de

Mark Schadler
mark.schadler@gmail.com

Angeliki Topalidou-Kyniazopoulou
angeliki@uni-bonn.de

Sven Behnke
behnke@cs.uni-bonn.de

Computer Science Institute VI, Autonomous Intelligent Systems
Rheinische Friedrich-Wilhelms-Universität Bonn
Friedrich-Ebert-Allee 144, 53113 Bonn

Abstract

Fully autonomous exploration and mobile manipulation in rough terrain are still beyond the state of the art—robotics challenges and competitions are held to facilitate and benchmark research in this direction. One example is the DLR SpaceBot Cup 2013, for which we developed an integrated robot system to semi-autonomously perform planetary exploration and manipulation tasks. Our robot explores, maps, and navigates in previously unknown, uneven terrain using a 3D laser scanner and an omnidirectional RGB-D camera. We developed manipulation capabilities for object retrieval and pick-and-place tasks. Many parts of the mission can be performed autonomously. In addition, we developed teleoperation interfaces on different levels of shared autonomy which allow for specifying missions, monitoring mission progress, and on-the-fly reconfiguration. To handle network communication interruptions and latencies between robot and operator station, we implemented a robust network layer for the middleware ROS. The integrated system has been demonstrated at the DLR SpaceBot Cup 2013. In addition, we conducted systematic experiments to evaluate the performance of our approaches.

1 Introduction

Robotic systems that explore and manipulate their environment have many potential applications such as search and rescue, remote inspection, or planetary exploration (e.g. [Mehling et al., 2007,Deegan et al., 2006, Guizzo and Deyle, 2012, Stilman et al., 2009, Borst et al., 2009, Schenker et al., 2003]). Robots operating within such real-world environments will have to navigate in complex terrain and reach, physically interact, pick up, retrieve and manipulate a variety of objects. Such mobile manipulation is an activity that humans naturally perform by combining two capabilities: locomotion and manipulation. The need of 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, chains, or legs.

While for many application domains fully autonomous operation in the complex, unstructured environments is not yet feasible, it is desirable to increase the level of autonomy in order to reduce the operator workload and to overcome communication restrictions. Autonomy also enables faster operations, which is an important feature in time-critical application scenarios such as search and rescue operations.

In this paper, we present an advanced robotic system for semi-autonomous exploration and manipulation tasks. With our system we competed in the DLR SpaceBot Cup 2013 robotics competition. In this competition, a robot has to explore previously unknown planetary-like terrain and build a 3D map of the environment. The robot needs to find and retrieve objects such as a cup-like container and a battery, and install them at a base station. One further difficulty of the competition setting is a communication link between operator station and robot that is affected by several seconds of latency and intermittent wireless connection.

Our Explorer robot, shown in Fig. 1, carries multiple 3D sensors, has a fast onboard computer, and a robust wireless communication link. With its six-wheeled drive and a 7 DoF manipulator, it can operate with a high level of autonomy. We developed efficient methods for allocentric and egocentric mapping of the environment, assessment of drivability, allocentric and egocentric path planning, object manipulation, intuitive teleoperation interfaces, and a robust communication system. In this article, we describe robot hardware and our novel software contributions which utilize the rich sensory equipment of our robot. We evaluate the system in several field experiments and report on the performance at the DLR SpaceBot Cup 2013 competition.

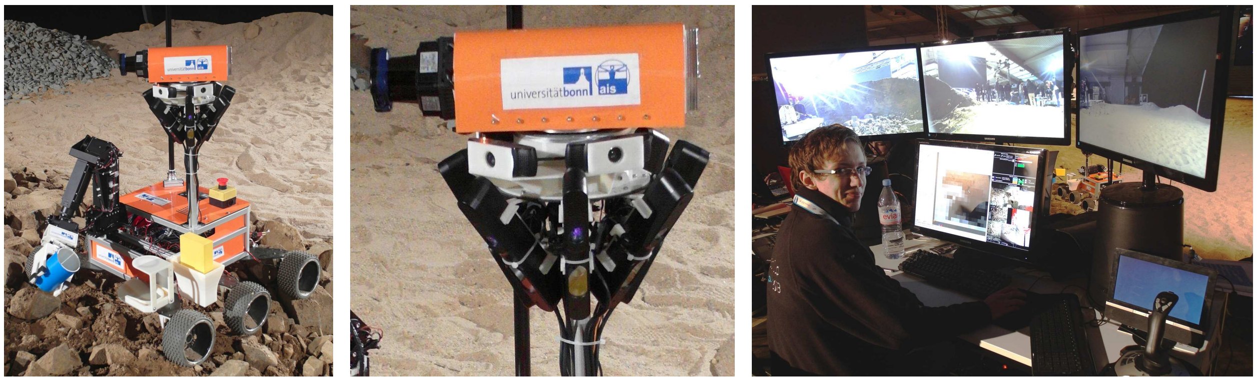


Figure 1: Left: Explorer robot with six-wheeled drive and 7 DoF arm for mobile manipulation in rough terrain. Center: The sensor head carries a 3D laser scanner, eight RGB-D cameras, and three HD cameras. Right: Our operator station consisting of four monitors and input devices including a joystick.

2 Related Work

Several robot application domains include navigation in unknown, rough terrain, mobile manipulation, and shared autonomy. E.g., the tasks performed by rescue robots range from exploration and mapping to mobile manipulation. The books of [Tadokoro, 2010] and [Murphy, 2014] give a good overview about the state-of-the-art in disaster-response robots. Space applications for exploration robots have a stronger focus on navigation with shared autonomy under communication latency and restrictions in the availability of communication [Chhaniyara et al., 2012, Maimone et al., 2007, Biesiadecki et al., 2007, Seelinger et al., 2012].

2.1 Exploration and Navigation in Rough Terrain

Frequently, the application domains require the robot to navigate in uneven, rough terrain. Recently, much research has been devoted to simultaneous localization and mapping (SLAM) in 3D, exploration, and navigation in rough terrain.

For mapping and localization in a previously unknown environment, early research focused on the 2D sub problem—assuming the robot moves on flat ground. As robots are increasingly deployed in less constrained environments, 3D mapping and localization has become an active research topic. To be applicable on autonomous robot platforms, sufficient onboard sensors are required, and the methods must be memory and run-time efficient.

Using multi-resolution maps to maintain high performance and low memory consumption has been investigated by several groups. [Hornung et al., 2013], for example, implement a multi-resolution map based on octrees (OctoMap). [Ryde and Hu, 2010] use voxel lists for efficient look-up. Both of these approaches consider mapping in 3D with a voxel being the smallest map element. Similar to our approach, the 3D-NDT [Magnusson et al., 2007] represents point clouds as Gaussian distributions in voxels at multiple resolutions. Our multi-resolution surfel maps (MRSMap [Stückler and Behnke, 2014]) adapt the maximum resolution with distance to the sensor to incorporate measurement characteristics. Our registration method matches 3D scans on all resolutions concurrently, utilizing the finest common resolution available between both maps, which also makes registration efficient. By exploiting the neighborhood of scan lines due to the continuous rotating 3D scan acquisition, map aggregation can also be made very efficient.

For SLAM, pose graph optimization methods are state-of-the-art. They adjust the scan poses to minimize scan registration errors. A stop-and-scan scheme as used in our approach can be beneficial, as it simplifies mapping and avoids inaccuracies that could arise with approaches performing mapping under motion. This comes with a trade-off for exploration speed, though. Some SLAM approaches integrate scan lines of a continuously rotating laser scanner into 3D maps while the robot is moving, e.g. [Bosse and Zlot, 2009, Elseberg et al., 2012, Stoyanov and Lilienthal, 2009, Maddern et al., 2012, Anderson and Barfoot, 2013, Droeschel et al., 2014, Holz and Behnke, 2014].

For online localization in a 3D map, several approaches have been proposed. [Kuemmerle et al., 2007], for example, apply Monte Carlo localization in multi-level surface maps [Triebel et al., 2006], which represent occupied height intervals on a 2D grid. [Klaess et al., 2012] model the environment in surfel maps in a fixed resolution, similar to the 3D-NDT [Magnusson et al., 2007]. They then localize in these maps using a tilting 2D laser by matching line elements extracted from the 2D scan lines in a particle filter framework, assuming motion of the robot in the horizontal plane. Our approach does not need to make this assumption and is suitable for rough terrain.

Path planning for navigating in planar 2D indoor environments is a well-studied topic in robotics [Hornung et al., 2012, Klaess et al., 2012]. For navigation on non-flat terrain, several approaches generate 2D cost maps from sensor readings to essentially treat path planning in 2D [Lee et al., 2008, Gerkey and Konolige, 2008, Ferguson and Likhachev, 2008]. Frequently, the terrain is modeled in elevation grid maps, on which planning can be performed [Kwak et al., 2008, Kewlani et al., 2009]. Recently, [Stoyanov et al., 2010] proposed a wavefront-propagation path planner in 3D-NDT maps. We use efficient search-based planners and propose a robust approach to transition between neighboring 3D scans.

Judging traversability of terrain and avoiding obstacles with robots—especially planetary rovers—has been investigated before. [Chhaniyara et al., 2012] provide a detailed survey of different sensor types and soil characterization methods. Most commonly employed are LIDAR sensors, e.g. [Gingras et al., 2010, Hebert and Vandapel, 2003], which combine wide depth range with high angular resolution. Chhaniyara et al. investigate LIDAR systems and conclude that they offer higher measurement density than stereo vision, but do not allow terrain classification based on color. Our RGB-D terrain sensor provides high-resolution combined depth and color measurements at high rates in all directions around the robot. We employ omnidirectional depth cameras, which provide instantaneous geometry information of the surrounding terrain with high resolution. In many environments, color or texture do not provide sufficient traversability information, so 3D geometry is needed.

We present an integrated system for obstacle-avoiding driving, SLAM, and path planning in rough terrain. In the context of space robotics, integrated navigation systems with shared autonomy have been successfully deployed for planetary exploration missions. The NASA Mars Exploration Rovers demonstrated semi-autonomous navigation on Mars for several years [Maimone et al., 2007, Biesiadecki et al., 2007]. They use stereo visual odometry to accurately track the position of the rover, such that slip in difficult terrain can be detected, and hazardous commands can be aborted. Operators plan paths via waypoints which are then executed using hazard-avoiding control on the robot. Since command sequences can be transmitted typically up to once per day, autonomous navigation into the unknown has proven to be an important feature to utilize mission time well.

2.2 Mobile Manipulation

Telemanipulation systems have been developed for applications that are inaccessible or hazardous for humans such as minimally invasive surgery [Ballantyne and Moll, 2003, Hagn et al., 2010], explosive ordnance disposal [Kron et al., 2004], and undersea [Whitcomb, 2000] or space applications [Pedersen et al., 2003]. The premise of telemanipulation systems is to utilize the cognitive capabilities of a human operator to solve dexterous manipulation tasks such as object pick-and-place, opening a door, or tool-use. However, this approach also leaves significant load to the operator. Implementing the required cognitive capabilities in autonomous systems is an open research topic. Many approaches to grasp and motion planning in unstructured environments assume that the geometry of objects is known and identify a set of stable grasps on objects in an offline phase [Miller et al., 2003, Berenson and Srinivasa, 2008, Nieuwenhuisen et al., 2013]. To grasp the objects in the actual scene, the grasp set is pruned by identifying those grasps that are reachable under kinematic and free-space constraints. Hsiao *et al.* [Hsiao et al., 2010] proposed a reactive approach to grasping that plans grasps at raw partial 3D measurements of objects, evaluating the resulting grasps for reachability by time-costly motion planning. In [Stückler et al., 2012], this approach has been extended to provide grasps and reaching motions very fast in situations, where parametric arm motion primitives can be used instead of costly planning of reaching motions.

Manipulation scenes can be segmented visually into object candidates through bottom-up cues without explicit knowledge about specific objects. For instance, Rusu *et al.* [Rusu et al., 2009] propose to describe and classify the local context of points by Fast Point Feature Histograms. They regularize the classification in a conditional random field on the point cloud to obtain coherent object segments. We adapt the segmentation approach of [Holz et al., 2012], which uses integral images to quickly estimate normals from RGB-D images and segments objects on flat support surfaces.

3 System Design

3.1 Explorer Robot Hardware

Our *Explorer* robot, shown in Fig. 1, is equipped with three wheels on each side for locomotion. The wheels are mounted on carbon composite springs to adjust to irregularities of the terrain in a passive way. As the wheel orientations are fixed, the robot is turned by skid-steering, similar to a differential drive system.

In addition to the omnidirectional RGB-D sensor, our robot is equipped with a rotating 2D laser scanner for allocentric navigation, an inertial measurement unit (IMU) for measuring the slope, and four consumer cameras for teleoperation. Three of these cameras cover 180° of the forward view in Full HD and one wide-angle overhead camera is looking downward and provides an omnidirectional view around the robot (see Fig. 5a).

For object manipulation tasks, the robot is equipped with a 7 DoF arm build from Robotis Dynamixel Pro actuators. These actuators come in three sizes which are distributed from strongest on the robot base (ca. 40 Nm continuous torque at 855 g) to weakest in the wrist (ca. 5.6 Nm at 340 g). The actuators have little backlash and are backdrivable. We built a custom gripper with two 3D printed fingers on rotary joints, actuated by Dynamixel MX-64 servos. Our lightweight and inexpensive gripper is specifically designed for the objects relevant to the DLR SpaceBot Cup.

3.1.1 Omnidirectional RGB-D Sensor

For local navigation, our robot is equipped with eight RGB-D cameras (ASUS Xtion Pro Live) capturing RGB and depth images. The RGB-D cameras are spaced such that they create a 360° representation of the immediate environment of the robot (Fig. 5b). Each camera provides 480×640 resolution depth images with 45° horizontal angle. Because the sensor head is not centered on the robot, the pitch angle of the cameras varies from 29° to 39° to ensure that the robot does not see too much of itself in the camera images. The camera transformations (from the optical frame to a frame on the robot's base) were calibrated manually. We carefully treat potential inaccuracies at the interfaces between the RGB-D images in later steps of the processing pipeline, such that, e.g., no artificial obstacles occur in the local navigation map. We observed no significant changes in this calibration, despite driving on rough terrain for many operation hours.

The high data rate of eight RGB-D cameras poses a challenge for data acquisition, as a single camera already consumes the bandwidth of a USB 2.0 bus. To overcome this limitation, we equipped the onboard PC with two PCI Express USB cards, which provide four USB host adapters each. This allows to connect each RGB-D camera on a separate USB bus which is not limited by transmissions from the other devices. Additionally, we wrote a custom driver for the cameras, as the standard driver (*openni_camera*) of the ROS middleware [Quigley et al., 2009] was neither efficient nor stable in this situation. Our driver can output VGA color and depth images at up to 30 Hz frame rate. The source code of the custom driver is available online[1].

3.1.2 Continuously Rotating 3D Laser Sensor

To map the terrain and for localization, our robot is equipped with a continuously rotating 3D laser scanner. It consists of a Hokuyo UTM-30LX-EW 2D laser range finder (LRF) which is rotated by a Dynamixel MX-64 servo actuator to gain a 3D FoV. The scanning plane is parallel to the axis of rotation (pointing upwards) and the heading direction of the scanner is always orthogonal to it. The 2D LRF is electrically connected by a slip ring, allowing for continuous rotation of the sensor.

The Hokuyo LRF has an apex angle of 270° and an angular resolution of 0.25°, resulting in 1,080 distance measurements per 2D scan, called a *scan line*. Due to its restricted FoV, the sensor has a blind spot of 90°. We point the center measurement beam of the sensor 225° downwards (if 0° points upwards). Pointing the laser downwards allows for 3D scanning the terrain in each half rotation. Additionally the laser is mounted

[1] `https://github.com/AIS-Bonn/ros_openni2_multicam`

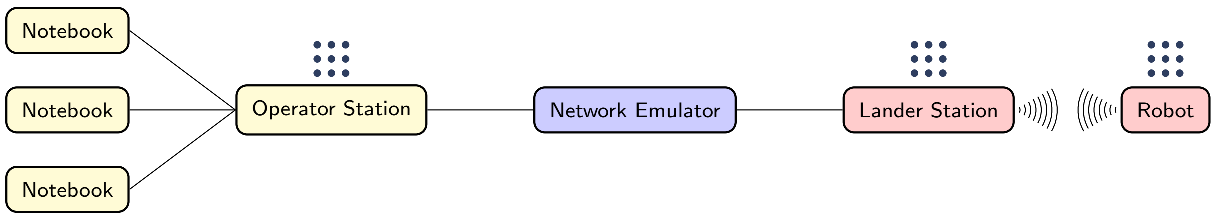


Figure 2: Overview of our communication link. Terrestrial components are shown in yellow, components at the destination site in red. The systems marked with the ROS logo (9 dots) run a ROS master.

with its scan plane off the axis of rotation. In this way, the FoV close around the robot is increased, and also occlusions by robot parts such as the WiFi antenna are reduced. The Dynamixel actuator rotates the 2D LRF at 1 Hz, resulting in 40 scan lines and 43,200 distance measurements per full rotation. Slower rotation is possible if a higher angular resolution is desired. For our setup, a half rotation leads to a full 3D scan of most of the environment. Hence, we can acquire 3D scans with up to 21,600 points with 2 Hz.

The 2D laser scanner has a size of 62×62×87.5 mm and a weight of 210 g. Together with the actuator (72 g) and the slip ring, the total weight of the 3D scanner is approximately 400 g.

3.2 Mission Control Station

Our robot can be teleoperated from a dedicated Operator Station. The Operator Station consists of a workstation with four monitors (see Fig. 1). It is connected to the robot over the communication link described in Section 3.3. The three upper monitors always show the images from the three Full-HD webcams mounted in the sensor head. The lower monitor shows telemetry from the robot and contains controls for teleoperation.

3.3 Communication Link

Our communication system has been designed according to the requirements of the DLR SpaceBot Cup, which include latency and communication interruptions. It consists of a chain of four devices (see Fig. 2): The operator station, a network latency emulator, a lander station, and finally the robot. To ensure mobility, the link between the lander station and the robot is implemented wirelessly, while all other elements of the chain are connected using Fast Ethernet.

The operator station is built from a standard PC (see section 3.2). It offers two Ethernet interfaces—one directly connected to the network emulator, the other one available to connect further PCs or notebooks for

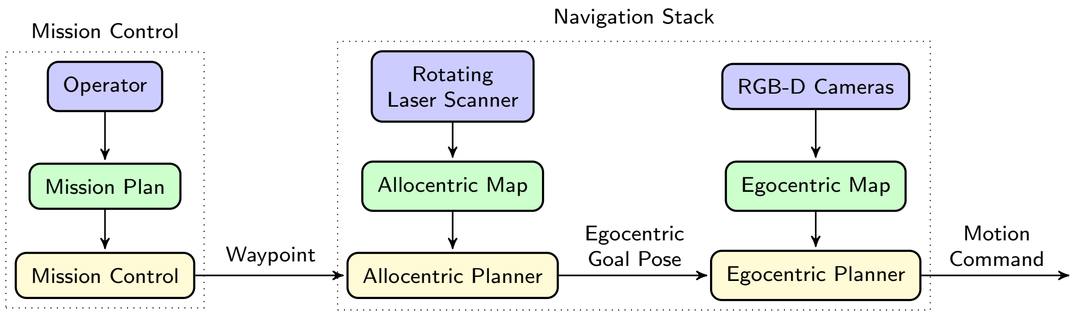


Figure 3: General overview of our navigation stack.

further visualization or debugging tasks.

The network emulator introduces a communication delay into the communication link. This simulates a deep-space connection, where long distance creates latency imposed by the speed of light. Note that deep-space connections can often still guarantee data transmission with high probabilities, as they use advanced Forward Error Correction (FEC) codes and multiple redundant links.

The lander station contains a small form factor PC (Apple Mac mini). The decision to include processing power at the lander station was taken to have the ability to buffer packetized telemetry at the lander station, as will be explained below. Additionally, the lander station possesses a high-gain wireless antenna (Ubiquiti Networks Bullet M5-HP). The robot carries a matching high-gain wireless antenna (see Fig. 1). The wireless communication takes place using the proprietary Ubiquiti Networks AirMax protocol, which offers better scalability and robustness in outdoor scenarios than standard WiFi (IEEE 802.11). Note that this local wireless link is not affected by space latency—a fact that we will exploit below.

4 Rough Terrain Navigation

To navigate in previously unknown rough terrain, the robot needs to possess advanced navigation capabilities ranging from autonomous mapping over localization to navigation. Furthermore, the operator needs to be able to interact with the navigation process on different levels. To facilitate this, we designed a two-layer navigation stack (see Fig. 3) consisting of a high-level SLAM approach—including path planning—and a low-level local navigation method. The higher, allocentric layer calculates an egocentric goal pose which is then approached by the lower, egocentric layer. The layers run on different time scales and are highly decoupled, even using different sensors for terrain sensing. While such a two-layer hierarchical approach is not new, we developed efficient algorithms on both levels that allow for 3D SLAM, rough terrain traversability analysis, and path-planning on-line.

The following sections will describe the navigation stack in more detail. Full descriptions of the allocentric and egocentric navigation components can be found in [Schadler et al., 2014] and [Schwarz and Behnke, 2014], respectively.

4.1 Visual Odometry

Our local navigation uses odometry to propagate waypoints specified in an ego-centric coordinate frame while driving towards a waypoint. Accurate odometry is especially useful when the operator is manually giving goal poses in this ego-centric frame, which can be necessary when higher levels of autonomy are not available. Odometry is used then to shift the given local goal pose as the robot moves. Note that during autonomous navigation, our allocentric path planner updates the ego-centric waypoint of the local planner at higher frequency, such that accuracy in odometry becomes less important.

The odometry measured from wheel movements can be very imprecise, especially in planetary missions with sand-like ground. Visual odometry is not affected by wheel slip and can provide accurate estimates even on difficult terrain. Hence, we first estimate movement for each camera with fovis [Huang et al., 2011] from gray-scale and depth images. Individual camera movement, wheel odometry and IMU data are then combined into a fused odometry estimate.

4.2 Egocentric Navigation

Figure 4 gives an overview of our local navigation pipeline. The input RGB-D point clouds are projected on a gravity-aligned plane, locally differentiated and then used as cost maps for existing navigation components.

4.2.1 Omnidirectional Height Map

For wheeled navigation on rough terrain, slopes and obstacles have to be perceived. Because we can measure the gravity direction with the integrated IMU, we calculate a 2.5D height map of the immediate environment which contains all information necessary for local navigation. This step greatly reduces the amount of data to be processed and allows planning in real time.

To create an egocentric height map, the eight separate point clouds are transformed into a local coordinate system on the robot base, which is aligned with the gravity vector measured by the IMU.

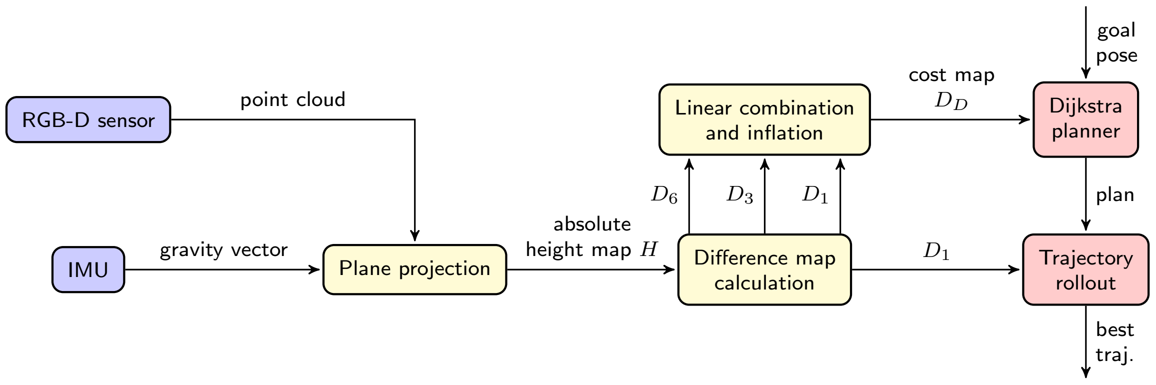


Figure 4: Data flow of our local navigation method. Sensors are colored blue, data filtering/processing modules yellow and navigation components in red.

The robot-centric 2.5D height map is represented as a 8×8 m grid with a resolution of 5×5 cm. For each map cell $H(x, y)$, we calculate the maximum height of the points whose projections onto the horizontal plane lie in the map cell. If there are no points in a cell, we assign a special NaN value. The maximum is used because small positive obstacles pose more danger to our robot than small negative obstacles. The amount of memory used by the resulting height map is two orders of magnitude smaller than the original point clouds of the eight cameras.

4.2.2 Drivability Assessment

An absolute height map is not meaningful for planning local paths or for avoiding obstacles. To assess drivability, the omnidirectional 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:

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

$H(u, v)$ values of 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.

Furthermore, we do not allow taking differences across camera boundaries. In cells which are visible from two cameras, the local height differences are calculated for each camera independently and the maximum value is used. This prevents small transformation errors between the cameras from showing up in the drivability

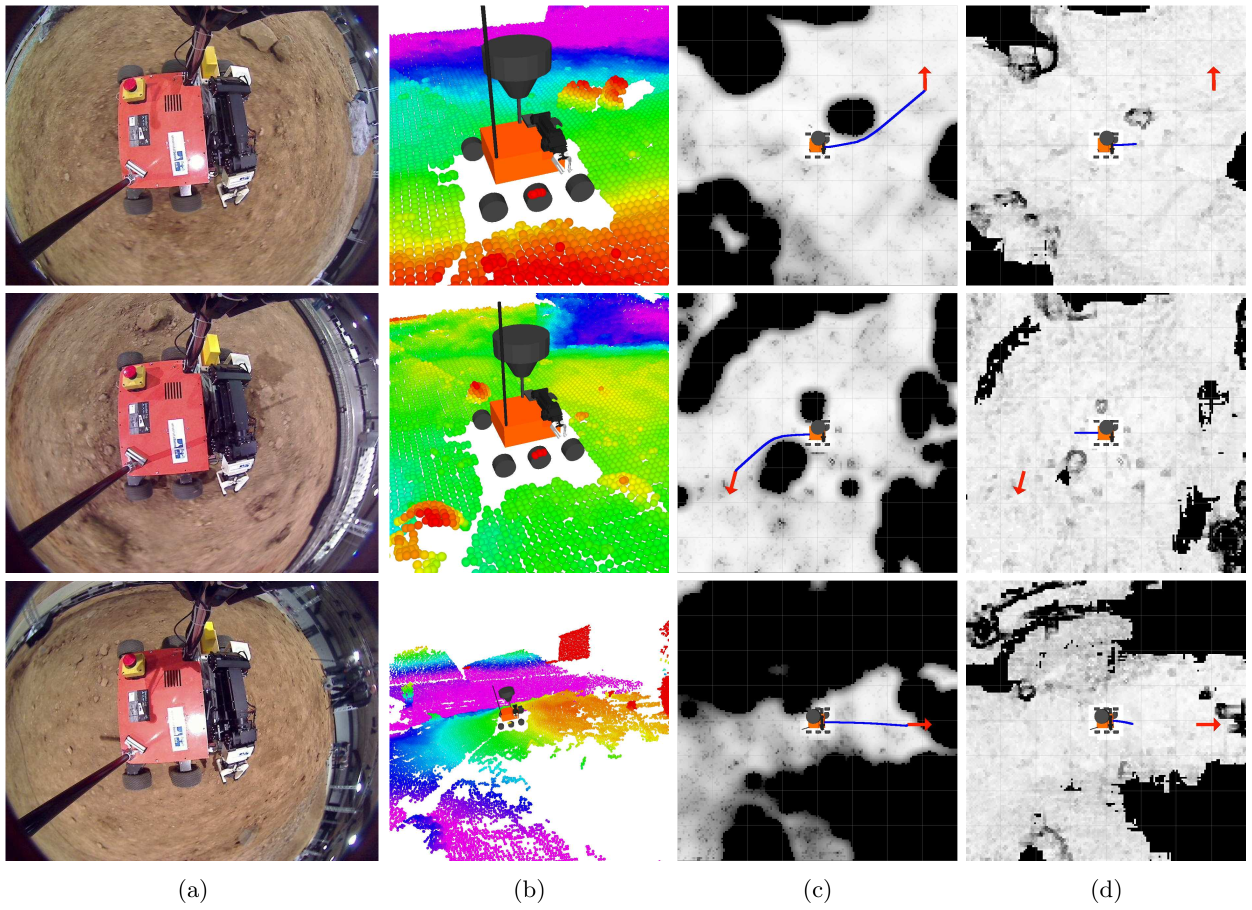


Figure 5: Several exemplary navigation situations. **a)** Wide-angle overhead camera. **b)** Downscaled point cloud, color-coded according to height. The color scale is chosen in each situation to highlight the important features. Red is high, blue/purple is low. **c)** Drivability map D_D with planned path (blue) to local goal (red). **d)** Obstacle map D_1 for precise trajectory rollout with optimal trajectory (blue).
First row: Slightly sloped ground, one close pair of rocks. Second row: Driving backwards through the opening between two rocks. Third row: Driving towards persons standing at the top of a large ramp.

maps as obstacles. In the relevant ranges close to the robot, this additional constraint does not reduce information, because the camera fields of view overlap. The required precision of the camera transformation calibration is thus greatly reduced.

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.

4.2.3 Local Navigation Planning

For path planning, the existing ROS solutions were employed. Our method creates custom obstacle maps for a Dijkstra planner (*navfn*) and a trajectory rollout planner (*base_local_planner*). Both components had to be slightly modified to allow for gradual, non-binary costs in the obstacle maps.

The Dijkstra planner is provided with a linear combination of D_l maps to include information about small obstacles as well as larger inclines. The summands are capped at a value of 0.5 to ensure that one D_l map alone cannot create an absolute obstacle of cost ≥ 1:

$$\tilde{D} := \sum_{l \in \{1,3,6\}} \min\{0.5; \lambda_l D_l\}$$

In addition, costs are inflated, because the Dijkstra planner treats the robot as a point and does not consider the robot footprint. In a first step, all absolute obstacles (with a cost greater or equal 1 in any of the maps D_l), are enlarged by the robot radius. A second step calculates local averages to create the effect that costs increase gradually close to obstacles:

$$\begin{aligned} P(x,y) &:= \{(u,v) | (x-u)^2 + (y-v)^2 < r^2\}, \\ D_D(x,y) &:= \max\left\{ \tilde{D}(x,y), \sum_{(u,v)\in P(x,y)} \frac{\tilde{D}(u,v)}{|P(x,y)|} \right\}. \end{aligned}$$

4.2.4 Robot Navigation Control

To determine forward driving speed and rotational speed that follow the planned trajectory and avoid obstacles, we use the ROS trajectory rollout planner (*base_local_planner*). It already sums costs under the robot polygon. As larger obstacles and general inclines are managed by the Dijkstra planner, we can simply pass the D_1 map as a cost map to the trajectory planner, which will then avoid sharp, dangerous edges. Replanning is done at least once every second to account for robot movement and novel terrain percepts.

4.3 Allocentric Mapping and Navigation

In order to accomplish navigation and exploration on an allocentric scale, a number of problems must be solved, including environment mapping, localization, drivability assessment, and path planning. The difficulty in these challenges increases in unconstrained 3D environments such as those encountered during planetary exploration missions. Additionally, environment sensing hardware should be accurate, efficient, affordable, and obey auxiliary constraints including weight and space restrictions.

To fulfill sensing hardware requirements, we use a continuously rotating laser scanner for environment perception. Multi-resolution surfel maps allow for a 3D environment and compact surface representation, efficient data-extraction from laser scanners, and fast alignment of maps. This makes the maps particularly well suited for our purpose, compared to other map representations that may lack one of the features such as fast registration, for instance. We acquire maps of the environment in a stop-and-go scanning scheme (Fig. 12). The 3D scans are registered to each other and aligned into a global map using graph optimization of the 3D scan poses.

The 6 DoF robot pose is continuously tracked while the robot is driving, based on the 2D scan lines of the continuously rotating 3D laser scanner. From multi-resolution surfel maps, we efficiently determine observation likelihoods of 2D scans allowing for real-time tracking of the 6D robot pose, as detailed below.

Drivability assessment is performed using an allocentric surfel map created during exploration. Due to the structure of a surfel map, surfaces can be easily extracted and evaluated for drivability characteristics including incline, smoothness, and neighbor continuity—resulting in a navigation graph. Using search algorithms, we determine drivable paths in the map. An intermediate waypoint on the current path is forwarded to the local navigation module which implements safe obstacle avoiding driving along the path. This waypoint is determined either at a specific distance ahead the robot on the planned path (e.g., 3 m), at the next pose with high curvature on the path, or at the goal if it is closer than the intermediate waypoint distance.

4.3.1 Simultaneous Localization and Mapping

Consistent 3D Mapping We use multi-resolution surfel maps (MRSMaps [Stückler and Behnke, 2014]) to efficiently represent environments. Octrees are the natural data structure for multiple-resolution information storage in 3D. Within octree voxels, both the surface shape parameters and surface reflectance distribution are stored as a surface element (surfel). Surfels approximate points within a voxel, which are considered normally distributed, by a sample mean and covariance. These statistical properties are stored through all resolutions in the octree. The maximum resolution at a measured point is determined in linear dependency of the distance of the point from the sensor. This corresponds to the decreasing sampling density and increasing noise, which are caused by the constant angular resolution and the measurement principle of our sensor, respectively.

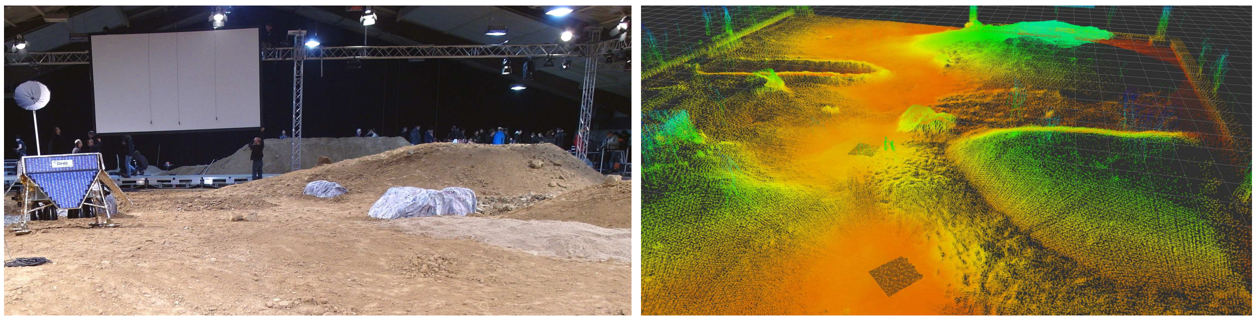


Figure 6: Left: Image of the DLR SpaceBot Cup arena. Right: Perspective view on aligned 3D scans of the DLR SpaceBot Cup arena (color codes height). Black rectangular areas occur at the scan poses at which cubic volumes around the robot have been removed from the 3D scans.

The registration of MRSMaps is implemented in two main steps: surfel association and pose optimization. Surfels are efficiently associated between maps from the finest resolution to coarser resolutions until associations have been determined for the entire map.

Associations are made between surfels having the closest Euclidean distance in position and shape-texture descriptors [Stückler and Behnke, 2014] within the query volume. To find the registration between two MRSMaps, we model pose optimization as finding the pose that maximizes the likelihood of observing one map at the pose in the other map. This procedure can be efficiently implemented using Levenberg-Marquardt's and Newton's method. Since this optimization method has local convergence property, we obtain an initial guess using particle filter localization (see Sec. 4.3.1).

Graph optimization is used to globally optimize the tracked pose from 3D laser scans. For each input 3D scan, a key-view is generated and globally aligned to a reference key-view, which is generated at the start of the mission. This alignment implies a geometric constraint between the key-views and is thus maintained as an edge in a key-view graph. As an additional step, all key-views deemed "close" are registered against each other to add edges to the constraint graph. The key-views are optimized in a probabilistic pose graph using the g2o framework [Kuemmerle et al., 2011].

After each 3D scan has been converted into a local MRSMap and inserted into the key-view graph, the updated poses of all key-views are used to create an allocentric MRSMap. This allocentric map is used by the particle filter for matching observations. Fig. 6 shows the resulting aligned 3D scans of the DLR SpaceBot Cup arena.

Localization The 6 DoF pose of the laser scanner between key-views in the surfel map is tracked with a particle filter. This pose estimate is used to initialize the registration transform for the next key-view.

Compared to other filtering methods that allow for non-linear motion and measurement models such as an Extended Kalman Filter (EKF), the particle filter allows for simple integration of a scan line-to-map measurement model and also has the ability to solve the global localization problem for future applications. The general idea of a particle filter is not discussed here; see [Chen, 2003] for a detailed introduction.

To compute a suitable state estimate, we model the *state transition* with a time-discrete linear dynamics model. The state estimation problem is posed as the estimation of the full 6 DoF configuration of the laser scanner. Odometry is also given as input to the propagation model with assumed Gaussian noise in translational and rotational parts. For our rover-type robot, we project the propagated pose onto the drivable surface. With an inertial measurement unit (IMU) given, the state propagation model also constrains pitch and roll orientation of the robot.

The *observation model* of a scan line is similar to the model for 3D scan registration—a single scan line does not contain enough information to register it to a target surfel map in 6 DoF, though. We circumvent this issue using the particle filter and determine the likelihood of a scan line, given a particle pose x. The observation model measures the alignment of the scan line $Z = \{z_i\}^n \in \mathbb{R}^3$ with n point measurements to the current target map m_t, given a particle pose x:

$$\begin{aligned} p(Z|x, m_t) &= \prod_{i=1}^{n} p(z_i|x, A(z_i, x, m_t)), \\ A(z_i, x, m_t) &= \operatorname*{argmin}_{s_t} d_{\text{plane}}(s_t, T(x)z_i), \end{aligned} \tag{1}$$

where $A(z_i, x, m_t)$ associates the transformed measurement point $z_{\text{transformed}} = T(x)z_i$ to the surfel within the target map having the smallest distance d_{plane} between the surfel plane and the target point. We only consider surfels for matching in a local volume around the transformed measurement point. The observation likelihood of a measurement to a surfel is given by the distance of the measurement end-point from the surfel plane under the normal distribution of the surfel.

$$\begin{aligned} p(z_i|x, A(z_i, x, m_t)) = p(z_i|x, s_t) &= \mathcal{N}(d_{plane}(s_t, T(x)z_i; x); 0, \Sigma_{sz}(x)), \\ \Sigma_{sz}(x) &= \Sigma_{s_m} + R(x)\Sigma_{z_i}R(x)^T, \end{aligned} \tag{2}$$

where Σ_{z_i} represents the point measurement covariance in $\mathbb{R}^3$ and $d_{plane}(s_t, T(x)z_i)$ is defined as the distance between the s_t surfel plane and the transformed measurement $T(x)\, z_i$. Since we use the motion model as proposal distribution in the particle filter, the importance weight for each particle is given by its observation likelihood.

4.3.2 Allocentric Navigation Planning

For navigation planning, we assume that the drivable surface can be projected onto a horizontal 2D representation, i.e., there are no overhanging structures the robot needs to drive under or upon. Thus, we choose a two-dimensional surfel grid as the data structure for drivability assessment and planning. Surface information must be stored within the grid to allow for terrain assessment—surfels lends themselves well to this task.

Using the allocentric localization map for planning could be prone to slight misalignment errors between the 3D scans. To give our approach more robustness, we fuse the MRSMaps of each individual 3D scan at a predefined resolution in the 2D grid. We overcome discretization effects by generating the MRSMaps in a common reference frame. By iterating over all surfels in each map, we can generate a global 2D surfel grid. Surfels within a specific distance from the origin of a scan, the ones supported by many measurements, are integrated directly. Outside of this distance, we keep the surfel with higher mean height, if the height difference between the surfels is larger than a threshold. Otherwise, the height difference is small and we choose the surfel supported by the largest number of measurements. Within a 3D scan, we merge surfels along the height direction to avoid the discretization of inclines.

We find the drivable cells in the 2D grid using region growing. We evaluate the drivable characteristics of the current grid cell from the robot coverage region in the grid map. This region is modeled by a bounding circle—an overestimate that neglects the effects of orientation on cost and allows for arbitrary steering. The drivability of an individual surfel is determined from several features which we detail in the following.

Data Coverage Data coverage is modeled by the number of cells containing valid surfels divided by the total number of cells within the region. The recommended region coverage depends on several factors including environment properties, safety considerations, and mapping resolution. It is, however, not possible to require a coverage of 100% due to occlusions and similar data limitations causing missing surfel data.

Bumpiness We define local bumpiness $b(c)$ of a cell c as the maximum difference between the surfel mean height $\mu_z(c)$ of the cell and the mean height of all eight directly neighboring surfels $n \in \mathcal{N}(c)$:

$$b(c) = \max_{n \in \mathcal{N}(c)} (\mu_z(c) - \mu_z(n)). \tag{3}$$

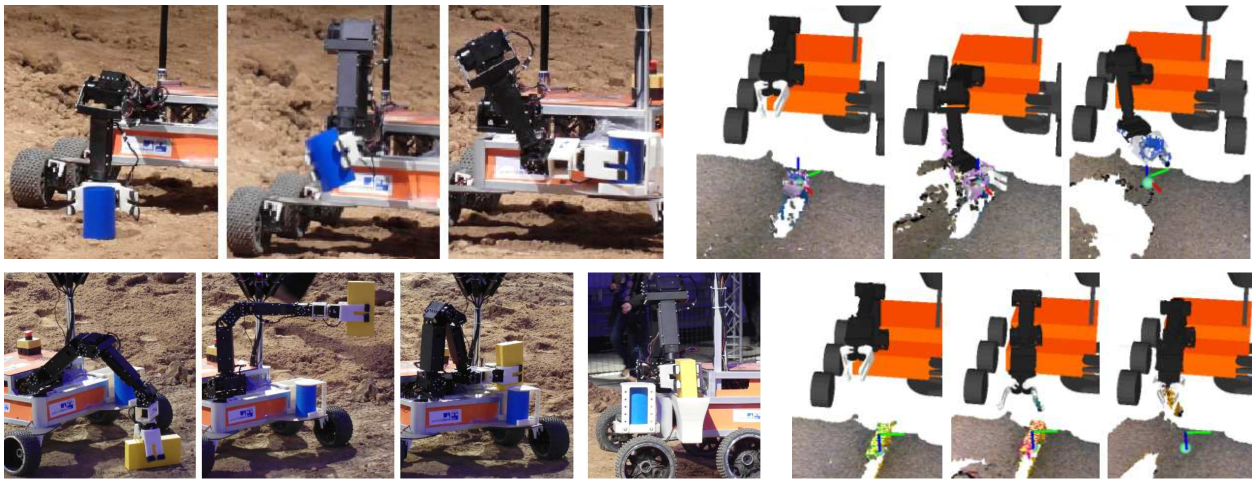


Figure 7: Top: Picture sequence of grasping a water-filled cup and placing it in its holder at the DLR SpaceBot Cup. Bottom: Grasping a battery pack, placing it in its holder, and removing it again. For both, photographs (left) and RGB-D image segmentation with estimated grasp poses (right) are shown.

The bumpiness threshold should be set depending on the robot ground coverage, robot wheel size, and general planning safeness required.

Incline Incline refers to the angle of a surface with reference to gravity, measured by the IMU. To determine the incline of a cell, we aggregate the statistical data of all surfels within the robot region to model a “region surfel”. The normal vector of this surfel is compared against the gravity vector for determining incline $i_r(c)$. The incline threshold should be related to the driving characteristics of the robot considering both motor capabilities and tipping resistance.

Drivability Cost The cost of a cell $C(c)$ is dependent upon both region bumpiness and incline. The cost is calculated as $C(c) = \alpha b_r(c) + \beta i_r(c)$, where α and β are the weight parameters for bumpiness and incline, respectively. Thresholding these combined cell costs offers extra safety limitations against the simultaneous effects of bumpiness and incline.

Path Planning Cost-optimal paths are planned using the A* algorithm in the resulting 2D grid. We use the Euclidean distance as its heuristic and consider traversion and drivability costs as edge and node costs.

5 Object Manipulation

Our *Explorer* robot can grasp objects and place them in its object holders (see Fig. 7). The holders are designed for a cup filled with water and a battery as used in the DLR SpaceBot competition.

Each object requires a specific type of grasp. We use side grasps for the cup and top grasps for the battery. The type of grasps is not only constrained by the environment and kinematics but also by the placement in the object holders.

Our approach segments RGB-D images for the desired object using the method of [Holz et al., 2012]. We describe the shape of each object segment by mean position, direction and length of its principal axes, and its maximum height above the floor.

Based on the object position and orientation, we plan all possible grasps for the desired object. The planning algorithm only considers the arm, which means that the robot base will not move for grasping the object. If no grasp is found, the robot drives again towards the object and tries to plan a grasp again. Given the segmented RGB-D image, we compute the two principal axes of the object and its sizes. For the calculation of the side grasp, we compute an ellipse that fits the boundaries of the object and sample poses from the computed ellipse every $10°$. For the top grasps, we sample poses from the larger principal axis. Our objective is that we can grasp the battery in a manner that we can carry it without dropping it, and place it in its holder. Thus, we sample the grasp poses from a small area around the center of mass of the battery. We use three different positions and another three different orientations which results in 9 grasps per battery position. The orientation sampling is performed only on the pitch axis where we deviate from a pitch angle of $90°$ due to arm kinematic constraints. All computed grasps are ranked according to reachability criteria which depend on the distance and the angle between the grasp pose and the base of the arm. We discard all grasp poses that have distance larger than the length of the arm or form an angle larger than $90°$. We rank the remaining grasp poses with respect to the similarity of the grasp pose orientation and the angle formed between the object and the base of the arm. The top ranked grasp is the one with the smallest difference of these two angles.

For the top-ranked grasps, we find a feasible trajectory for the robot arm. We feed the trajectory planning algorithm with a sequence of end effector poses and the type of interpolation from one pose to the next. The sequence of poses include a start pose, a pre-grasp pose, the grasp pose, and a final holding pose that interfaces with a predefined placement motion of the object into its holder. For each pair of poses, we interpolate, in 3D Cartesian space or in polar space in the XY plane, with constant acceleration for the first half of the motion and constant deceleration for the second half. When a new pose is computed, we obtain the corresponding values in joint-space through inverse kinematics. In order to not spill water in the cup, we choose a slow and smooth motion, while the cup must be carried in upright orientation at all times during the motion. Thus, we interpolate in XY plane and not in 3D Cartesian space. Besides that, the motion does not follow any other particular velocity or acceleration constraints. In order to not spill water onto the robot, the robot pours a small portion from full cups before placing the cup in its holder.

Once a trajectory is successfully planned, the robot performs the motion.

6 Communication

The communication link imposes constraints on all communication between the system components (see Section 3.3). In particular:

1. The deep-space link has large latency.
2. The local planet-side communication over WiFi is unreliable.

Our approach makes the following additional assumptions:

1. The deep-space link is reasonably reliable (could be implemented by strong FEC algorithms and multiple redundant communication links).
2. The planet-side communication has small latency.

Our software stack deals with the imposed constraints on two levels. A low-level communication module provides network transparency for the ROS systems running on each PC. Furthermore, we designed the high-level software to exploit the different characteristics of each communication link.

6.1 Low-Level Communication Software

Since both the operator station and the robot run the ROS infrastructure which offers built-in network transparency, we investigated this built-in capability. In the default configuration, ROS uses the TCP protocol for all connections over the network. This is unacceptable in our setting, since the TCP protocol enforces a 3-way handshake for opening connections, which quickly causes high delays since the connection may be lost and needs to be re-opened during the mission. Furthermore, TCP gives a transmission guarantee for all data packets. This (usually desired) property is not needed for all types of data (e.g. streamed, live camera images) and can cause the bandwidth to be exhausted with re-transmissions of data which is not interesting, because it is too old already.

Hence, we decided to design our own communication infrastructure which hooks into ROS and provides network transparency without the aforementioned problems. Our communication module offers sending/receiving ROS topic data over TCP or UDP, and ROS service calls over TCP. Its features include automatic rate limiting (with upper and lower bounds) and transparent BZip2 compression for large data packets. Our solution is a so-called multimaster approach, since we use autonomous ROS instances on the control station PC and on the robot.

The UDP protocol option is used whenever possible, since it is handshake- and connectionless and thus offers low latency and high robustness against temporary communication blackouts. If a transmission guarantee is needed (see Sec. 6), we use the TCP transport. However, even in this situation our package has advantages over the standard ROS approach, since we use a single permanent TCP connection for all communications. This avoids delays caused by opening connections during service calls. Furthermore, we activate TCP timeouts in the Linux kernel, which prevents hang-ups.

Note that ROS offers its own UDP protocol option (without the advanced rate-limiting and compression features of our module). Sadly, even if it is enabled, TCP is still used for topic discovery and connection setup, which makes the built-in ROS solution not suitable for our application.

6.2 High-Level Design Decisions

The two constraining elements in our communication link (see Fig. 2) are the Ethernet link over the network emulator (introducing latency) and the WiFi link between lander station and robot (unreliable). Since the two effects of latency and unreliability occur in different sections, we can deal with them separately.

For instance, the maps generated by our SLAM component need to be displayed to the operator to increase situational awareness. Since the maps are generated very infrequently, we want to be sure that each map arrives at the operator station. A naive approach would be to use the TCP protocol with a direct connection from robot to the operator station. But this solution greatly increases communication latency, since each missing packet can only be re-transmitted after a 4 s delay. Instead, we first transfer them to the lander

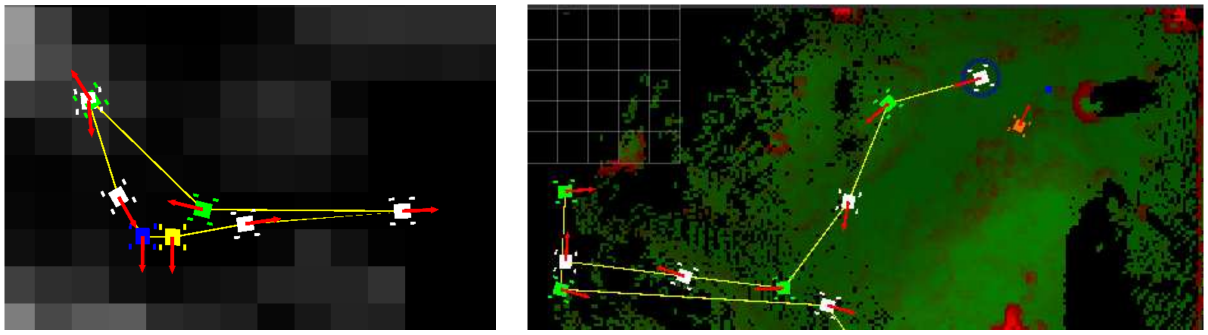


Figure 8: Left: Pre-mission planning on a rough height map. Right: Interactive plan refinement on an updated height map. Red cells indicate that the height of the refined map deviates by a great amount from the rough heightmap. This information is used by the operator to align the two maps precisely.

station using the TCP protocol, since re-transmissions of dropped packets are not expensive over the WiFi link. The complete map is then transmitted to the operator station in a single UDP burst. This way, we eliminate the unreliability of the WiFi link without increasing latency.

7 Mission Control Interfaces

Our system exposes three levels of control to the operating crew. On the highest level, it is possible to specify entire missions, which the robot then carries out autonomously. The mid level allows the operator to specify short-term tasks, such as grasping an object or navigating to a specified pose. Finally, the lowest level can be used for direct remote control, e.g. driving the robot with a joystick. The following sections will describe each level in more detail, starting at the highest level.

7.1 Mission Planning

Without a high grade of autonomy, it would not be possible to complete the tasks defined by the SpaceBot Cup in acceptable time. Our mission control layer is thus able to carry out most of the tasks autonomously, except for the manipulation at the base station, which is done by means of semi-autonomous teleoperation.

Since a rough height map (1 m horizontal, 0.3 m vertical resolution) was available before the competition, the operator can specify the initial mission tasks on this rough map (see Fig. 8, left). The displayed height map is refined from the SLAM map that is aggregated from the robot 3D scans (see Section 4.3.1) during the mission. During an initial setup phase, the height map is aligned manually with the SLAM map.

The operator receives new terrain maps after each time the robot stops for mapping (see Section 4.3.1).

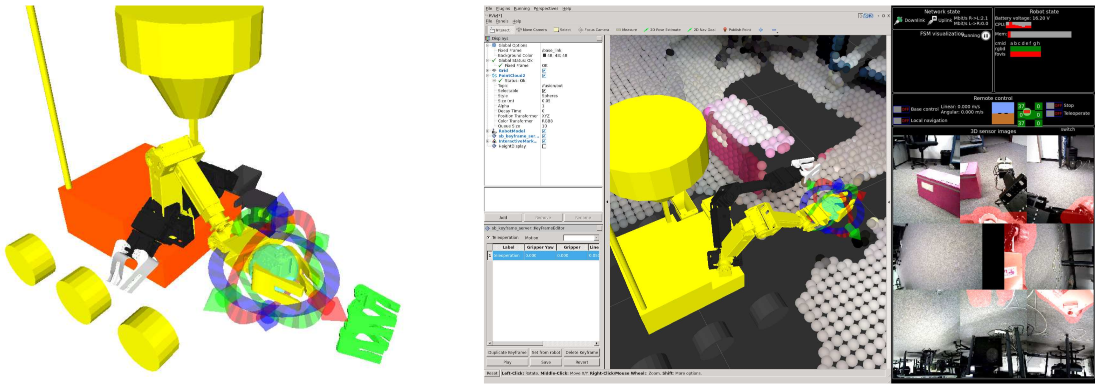


Figure 9: Left: Motion Keyframe Editor GUI for semi-autonomous robot arm teleoperation. The real position of the arm is indicated in true colors (black/white). The currently edited keyframe is shown in yellow. Interactive markers can be used to modify the keyframe pose in 6D. Other keyframes of the same motion can be seen in the lower right, rendered only as end effector positions (green) to reduce screen cluttering. Right: The same GUI embedded in our teleoperation GUI, also showing a transmitted downsampled point cloud in the same 3D view and images from the eight RGB-D sensors on the right.

Since the updated data may indicate that the mission plan needs to be changed, the operator can modify the mission plan on the updated map (see Fig. 8, right) and upload it at any point in time to the robot. The mission control layer then automatically aborts current tasks and immediately starts on the new plan.

If the operator judges that the robot is not able to perform the needed tasks autonomously, he or she can pause the mission plan execution and perform the problematic task using the lower-level controls. After deleting the task from the mission plan, autonomous mission execution can be resumed.

7.2 Semi-Autonomous Control

The mid layer consists of smaller sub-tasks, which are called either from the mission control layer or directly by the operator.

The available manipulation tasks include variable keyframe motions either in joint-space or in Cartesian space. To facilitate motion entry, we designed a keyframe editor GUI on top of the standard ROS rviz GUI (see Fig.9 left). The operator can specify arm motion keyframes in the form of 6D end effector poses. End effector velocity and joint velocity can be constrained. The interpolation between keyframes can be executed in joint space or Cartesian end effector space. The operator can also see a downsampled RGB-D point cloud of the robot environment to increase situation awareness (see Fig.9 right).

For semi-autonomous navigation, the operator can specify goal poses relative to the robot. Using the local navigation component (see Section 4.2), the robot navigates to the goal position while shifting it using the visual odometry (see Section 4.1). Since this mode of operation is not dependent on localization, it can be used even if the robot is delocalized.

7.3 Direct Teleoperation Interfaces

Finally, the operator can directly control the robot drive using a connected joystick. Of course, it is not possible to drive fast using direct teleoperation, since the communication delay limits reaction speed. For this reason, this mode of operation is only used for small movements, e.g. when the higher-level navigation controls fail to reach the target position.

8 Evaluation

We demonstrated our system during the DLR SpaceBot Cup 2013, which focused on exploration and mobile manipulation in rough terrain. It simulated a planetary exploration mission with a simulated communication link including a 2 s one-way latency in both directions. In addition, we conducted systematic experiments to evaluate the performance of our navigation and manipulation sub-systems.

8.1 Egocentric Navigation System

Fig. 10 shows a series of drivability maps taken from a single drive on an S-shaped path through the opening between two obstacles. The goal pose was given once and then shifted with visual odometry as the robot approached it. The processing pipeline from eight 240×320 RGB-D point clouds to the two drivability maps runs in around 50 ms on the onboard computer, which has an Intel Core i7-4770K processor with 3.5 GHz.

We also evaluated our custom camera driver. The standard ROS driver is not stable enough with eight cameras for direct comparison, so we can only provide absolute metrics for our driver. Capturing VGA color and depth images and converting them to RGB-D point clouds at 30 Hz is possible and creates a CPU utilization of about 40 % on the same processor. At the reduced setting of QVGA resolution and 10 Hz rate, the CPU utilization is at 16 %.

Of course, obstacle detection is dependent on obstacle size and distance to the obstacle. The limiting factors here are depth sensor resolution and noisiness of the depth image. To investigate this, we placed several obstacles, shown in Fig. 11, at varying distances to the robot on flat ground (carpet). In each situation, 120 frames of the local obstacle map D_1 were recorded from a static robot position. The objects have been placed

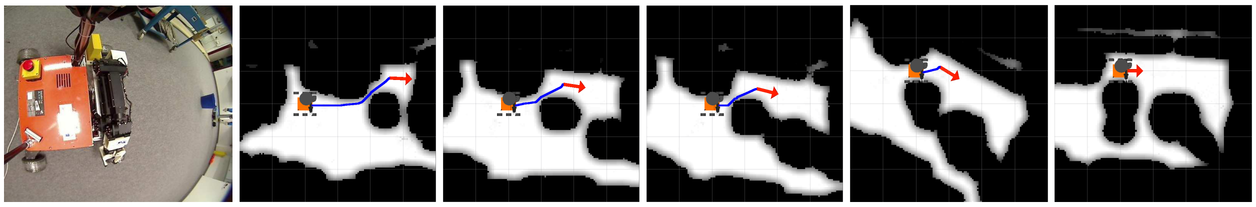
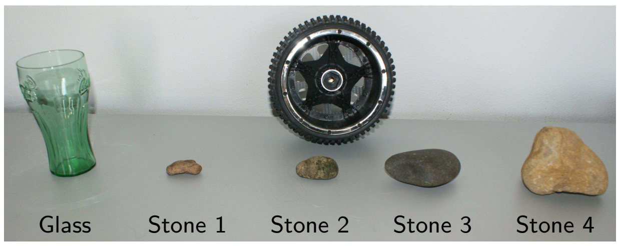


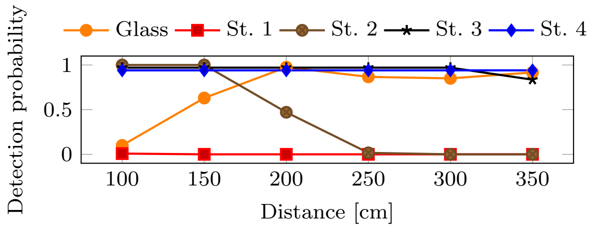
Figure 10: Lab driving experiment. The first image shows the initial situation observed from the robot overhead camera. Subsequent drivability maps show the planned path (blue line) towards the goal (red arrow).



Object dimensions.

Object	Glass	Stone 1	St. 2	St. 3	St. 4
Height [cm]	14.5	1.5	2.5	3.5	7.5
Width [cm]	7.5	4.5	5.0	9.0	10.0

(a) Investigated objects and robot wheel (for comparison).



Object	100	150	200	250	300	350
Glass	0.10	**0.63**	0.97	0.87	0.85	0.92
Stone 1	0.01	0.00	0.00	0.00	0.00	0.00
Stone 2	1.00	1.00	**0.47**	0.02	0.00	0.00
Stone 3	1.00	1.00	1.00	1.00	1.00	0.87
Stone 4	1.00	1.00	1.00	1.00	1.00	1.00

(b) Detection probabilities at increasing distance.

Figure 11: Obstacle detection experiment. The objects are placed at various distances on carpet in front of the robot and the detection probabilities are estimated by comparing height differences to a threshold.

in approximately the relative angle towards the robot shown in Fig. 11. To estimate detection probabilities, we count the number of frames containing cells near the expected obstacle location with a difference value higher than a pre-defined threshold. As can be seen in Fig. 11, the system can reliably detect obstacles which pose danger to the robot up to 3.5 m away.

8.2 Allocentric Navigation System

Using data collected at the DLR SpaceBot Cup 2013, results are presented illustrating the accuracy of the SLAM system and the ability to plan trajectories spanning several 3D scans. The data was processed in real-time using the robot on-board Intel Core i7-4770K (max 3.5 GHz) with 32 GB RAM. Localization is performed using 250 particles. To ensure dense point clouds from 3D laser scans, we rotate the laser at a slow speed ($\frac{1}{15}$ Hz) when creating full 3D scans. For tracking, we wish to maximize visible space per time period and rotate the laser at 1 Hz. Note that during the recording, persons have walked through the arena causing spurious measurements. Shown in Fig. 6, the SpaceBot Cup arena models a rough-terrain

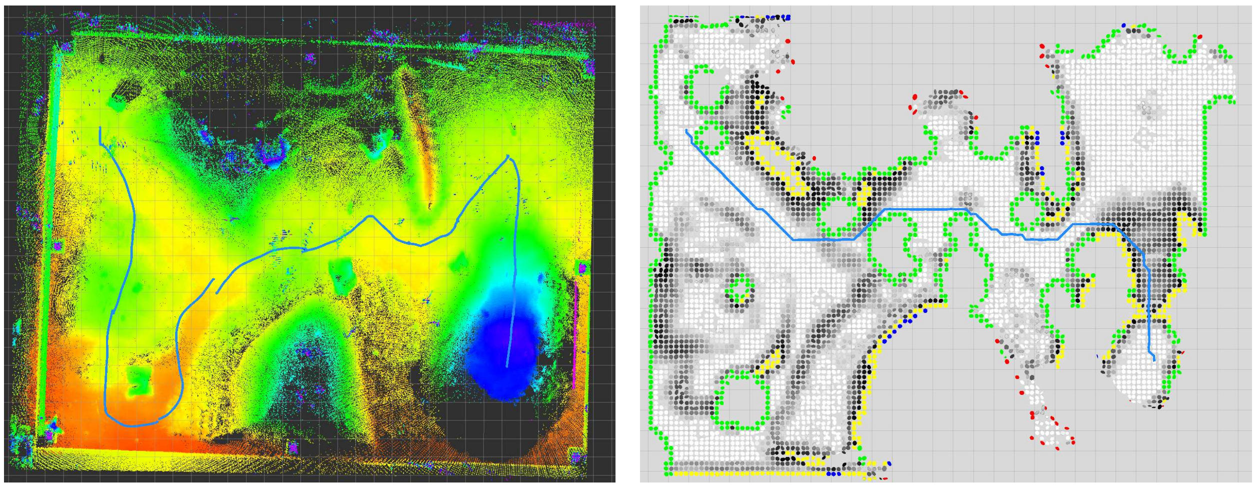


Figure 12: Left: Top-view on aligned 3D scans of the DLR SpaceBot Cup arena (color codes height). Localization estimate in blue. Right: Drivable region. Grayscale cells colored by cost. Undrivable cells due to lack of data coverage (red), large incline (blue), large region bumpiness (green), and combined cost (yellow). Planned path in blue.

terrestrial environment similar to those potentially visited by space exploration. Within the arena, the robot has been controlled manually to take 14 3D scans at an average distance of 4.33 m. To more closely resemble the open-environments in space, we have removed ceiling and wall measurements from the 3D scans before processing. Fig. 12 shows aligned scans, localization estimate, a planning map, and a planned path. The 3D scans align very accurately, and the localization estimate follows the actual path of the robot well. The visible gap in the localization estimate shows the difference between the localization estimate at one of the scan poses and its registered, graph-optimized pose. Such drift, for example, occurs through strong drift of the wheel odometry, or if the laser scan alignment is not constrained along a direction. While localization can slowly compensate such drift, scan alignment at the next scan pose typically removes the drift.

A typical 3D scan has about 516,000 points, for which a MRSMap is created. For pose graph optimization, this map is registered with maps of other 3D scans. Such a map-to-map registration takes on average 0.24 s using a maximum voxel resolution of 0.125 m. Building the MRSMap representation takes on average 0.49 s in this resolution. The time consumed for graph optimization itself is negligible in comparison and is in the milliseconds for this size of graphs. Planning time is typically in the seconds and depends on the distance between start and goal pose and the difficulty of the terrain in between.

We also evaluated the performance of our localization and SLAM system in environments such as a courtyard and a parking garage (s. Fig. 13). In these experiments, we could measure ground truth distances between the 3D scan poses and compare these relative measurements with the results of localization and pose graph

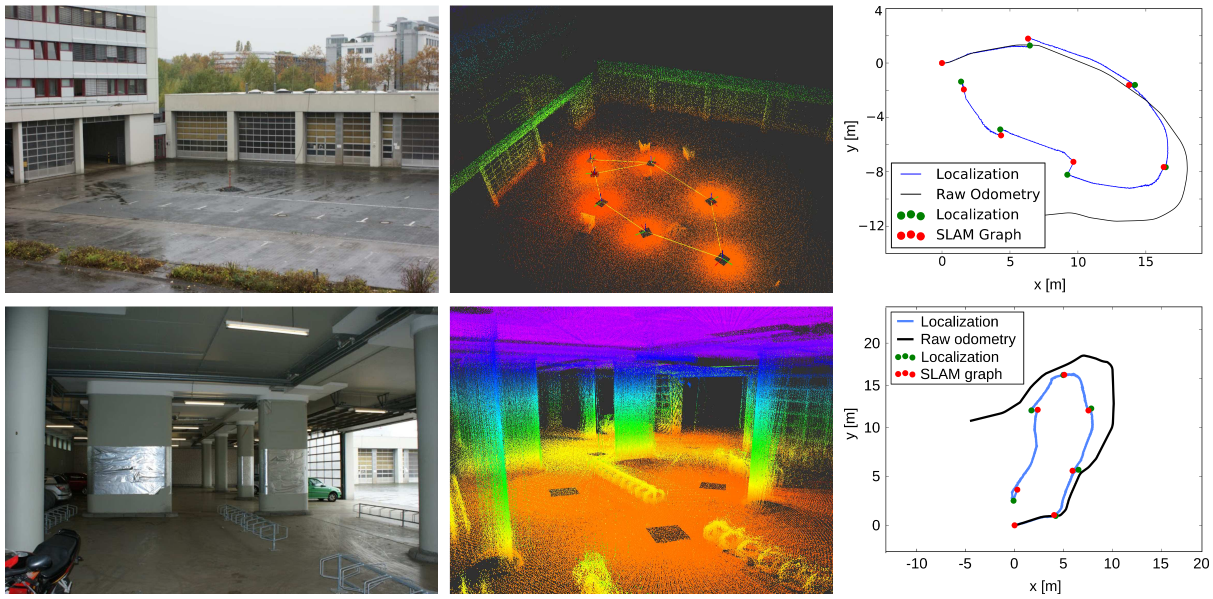


Figure 13: SLAM experiments in a courtyard (top) and a parking garage (bottom). Left: image of the environment. Center: aligned 3D scans. Right: Odometry (black), localization path (blue), 3D scan pose estimates from localization (green) and pose graph optimization (red).

Table 1: Parking garage distances between 3D scans in meters.

3D Scans	0-1	1-2	2-3	3-4	4-5	5-6	Average Error
Ground Truth	4.20	4.94	6.53	4.45	4.54	8.54	
Localization	4.30	5.26	6.39	4.44	4.85	9.40	0.29
SLAM	4.18	4.90	6.37	4.40	4.49	8.43	0.07

optimization, i.e. SLAM. From Tables 1 and 2, we see that particle filter localization provides good initial estimates with average errors of 0.29 m and 0.20 m, respectively. Pose graph optimization, which utilizes spatial constraints from several combinations of scan-to-scan registrations, yields accurate estimates in the centimeters. Notably, average accuracy is slightly higher in the courtyard experiment. The parking garage dataset includes occlusion situations and structures seen from two sides (e.g. the pillars) which can negatively affect scan registration accuracy.

Compared to the rugged terrain in the DLR SpaceBot Cup arena, the surfaces in these environments appear flatter. In MRSMaps, we adapt the maximum resolution in the map with distance from the sensor in order to adapt the representation to the observable sampling rate. Hence, we obtain accurate scan registration also in the case of rugged terrain as demonstrated in Fig. 12.

Table 2: Courtyard distances between 3D scans in meters.

3D Scans	0-1	1-2	2-3	3-4	4-5	5-6	Average Error
Ground Truth	6.60	8.25	6.59	6.69	5.69	4.37	
Localization	6.59	8.26	6.47	7.27	5.97	4.57	0.20
SLAM	6.58	8.18	6.55	6.66	5.67	4.35	0.03

Table 3: Planning time per motion and per grasp and planning attempts until success.

	Average	Median	Maximum	Minimum	Std. deviation
Time per motion (in s)	1.693	1.813	4.602	0.005	1.246
Time per grasp (in s)	5.622	4.938	14.481	1.677	3.689
Number of attempts	3	3	6	2	2.627

8.3 Object Manipulation

To evaluate the autonomous object manipulation capabilities of our robot, we placed either object (cup/battery) in a rectangular area in front of the robot which is confined in the forward direction in the interval $[0.45\,m, 0.7\,m]$ and $[-0.25\,m, 0.1\,m]$ in sidewards direction. This area is defined by the workspace of the robot and the field of view of the two frontal cameras that are used for object perception. Before grasping, our robot drives towards the objects to bring them into this predefined area.

Grasping has been performed at 10 random positions and orientations for the battery and the cup inside the predefined area. For each object pose, grasp pose candidates are determined and ranked. These poses are fed to the planning algorithm which attempts to find a feasible motion in order to achieve the desired pose. We consider the grasp successful if there is one successful planning attempt among the top-ranked grasp pose candidates and the robot manages to pick the object as expected and lifts it.

Our evaluation results are shown in Table 3. All grasps were successful in each situation, while not all grasp candidates yielded a valid motion plan. As shown, we need on average three planning attempts per grasp. This can be due to a collision or to a missing inverse kinematics solution at a point of the necessary approach motion. For each planning attempt, we need on average 1.693 s, which is mainly constituted by the computation of the inverse kinematics and the collision checking along the trajectory at our controller sampling rate.

The amount of time per grasp depends on the number of planning attempts needed, which in turn is affected by the position of the object within the workspace. More difficult planning situations occur close to the border of the workspace. If the object is closer to the robot, there is high probability of collisions between

Table 4: Execution time results of teleoperated cup placement on the base station with latency of 2 s.

Time (min:sec)	Average	Median	Maximum	Minimum	Std. deviation
Cup removal from robot	2:21	2:24	2:39	2:10	0:11
Remote Driving	1:10	0:50	1:50	0:42	0:34
Remote Cup Placement	2:03	2:04	2:53	0:57	0:43
Complete Task	5:34	5:26	6:21	3:52	0:38

Table 5: Execution time results of teleoperated cup placement on the base station, without latency.

Time (min:sec)	Average	Median	Maximum	Minimum	Std. deviation
Cup removal from robot	2:10	2:02	2:10	2:02	0:04
Remote Driving	0:32	0:33	0:40	0:23	0:08
Remote Cup Placement	1:16	1:16	1:22	1:11	0:04
Complete Task	3:48	3:47	4:00	3:39	0:08

the arm and the robot, because there is limited space between the robot and the object. Such difficult situations are eventually resolved when the robot reattempts to position itself towards the object if a valid grasp cannot be computed. Note that while our method requires several seconds to plan and execute a grasp, it is 100 % successful in our experiment.

8.4 Direct Teleoperation

Our direct teleoperation interfaces allow us to remotely drive the robot and manipulate objects. Our robot autonomously can pick up objects and place them in its holders in many situations in which the object can be segmented from its surrounding. In difficult situations, and for installing the object at the base station or manipulating its switch, the robot can be teleoperated.

We evaluated our teleoperation interface in experiments to measure execution time and the effect of latency on timing. For this purpose, our trained and experienced operator removed the cup from the robot and placed it on the base station and turned on the switch of the base station 5 times. We performed the experiment with network latency of 2 s and without network latency. The results of the above experiment are shown in Tables 4, 6, 5 and 7 respectively. As expected, the robot and the operator needed more time

Table 6: Execution time results of teleoperated switch manipulation with latency of 2 s.

Time (min:sec)	Average	Median	Maximum	Minimum	Std. deviation
Remote Driving	0:44	0:48	0:56	0:23	0:13
Remote Manipulation	2:33	2:27	2:58	2:02	0:23
Complete Task	3:10	3:10	2:48	3:46	0:22

Table 7: Execution time results of teleoperated switch manipulation, without latency.

Time (min:sec)	Average	Median	Maximum	Minimum	Std. deviation
Remote Driving	0:38	0:34	0:54	0:25	0:11
Remote Manipulation	1:23	1:16	1:37	1:14	0:11
Complete Task	2:01	2:05	2:23	1:41	0:17

with the 2 s latency. All attempts were successful. For the experiment, we placed the base station in 1.5 m distance from the robot in various orientations and positions. Our operator was in a different room and he obtained information about the environment only from the robot sensors (see Fig. 9).

8.5 Full System Performance at DLR SpaceBot Cup

During our competition run in the SpaceBot Cup, our robot for the first time had the chance to enter the arena. First, our system created a 3D map of the environment from an initial scan taken at the starting pose. The robot then autonomously started the execution of a predefined mission. After one of the manipulation objects, a small cup filled with water, became visible, we took a first check-point to update the mission plan accordingly. We teleoperated the robot in front of the object and triggered the autonomous grasping of the object and storing in its appropriate holder. The cup was autonomously grasped and stowed on the robot without spilling any of the contents (see Fig. 7, top left). Afterwards, we had a lengthy pause caused by a failure of the driver of our rotating laser scanner. The autonomous execution of the mission could not be continued and the run was aborted.

Directly after our run, we demonstrated the capabilities of our local navigation component by driving trough the entire the arena (including a large hill) in a semi-autonomous mode with manually specified ego-centric waypoints (see Fig. 14).

After the end of the competition, we could enter the arena again and demonstrated autonomous grasping of the other manipulation object (a battery) and teleoperated placement of both objects on a base station object (see Fig. 7, Fig. 14). A video of our participation at DLR SpaceBot Cup is available online [2].

During the competition, we could demonstrate all components of our robot system in an integrated fashion. Even getting the system started into the competition has turned out to be difficult for many other teams. A robust and well tested communication link and a well designed teleoperation interface have been key to get the robot start its competition mission. Autonomous navigation has been shown by only a few teams. Only

[2]`https://www.youtube.com/watch?v=gHso1nIyEu0`

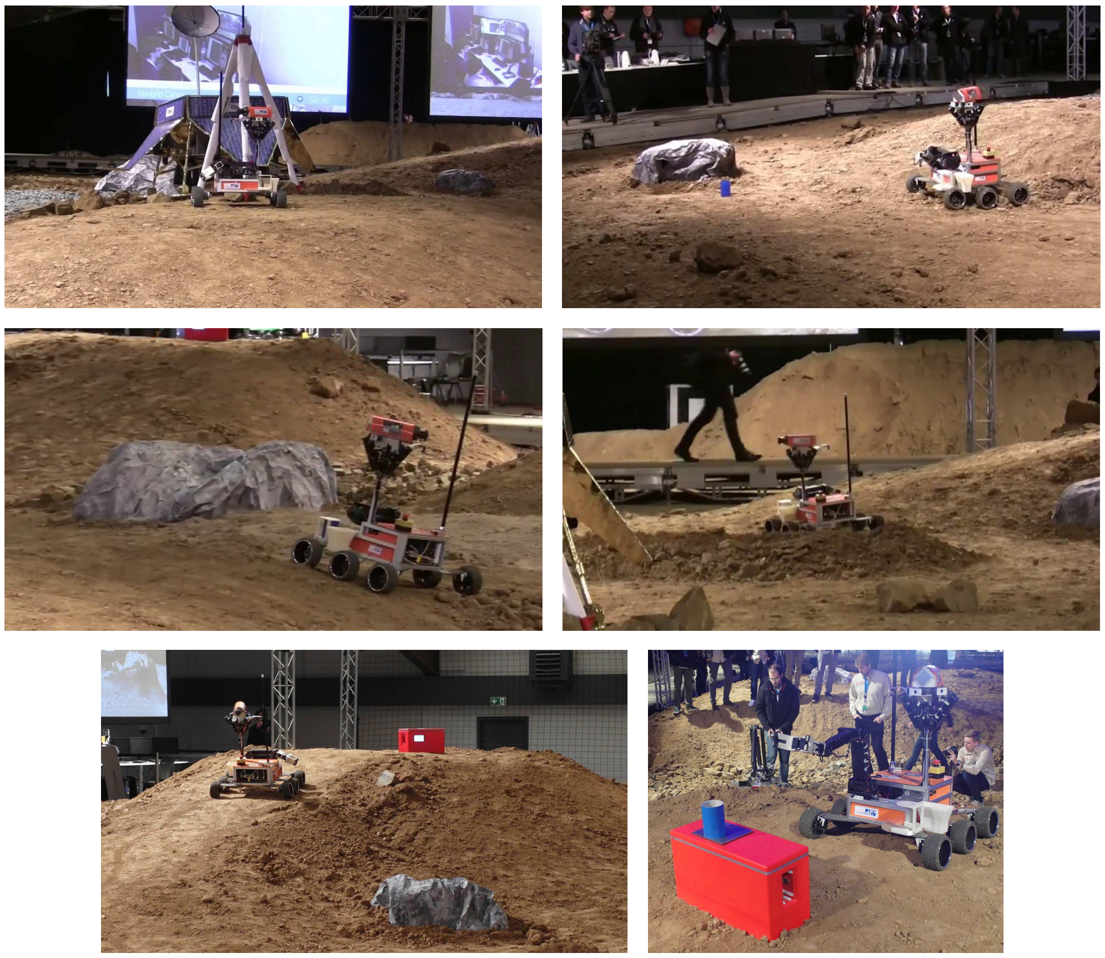


Figure 14: Stills from our evaluation at the DLR SpaceBot Cup. The top row contains images from our competition run, the other images were taken after our run.

two of the ten participating teams could demonstrate obstacle avoidance. Our omnidirectional 3D multi-camera and laser setup gave us a large advantage in perceiving obstacles, analyzing terrain for traversability, and path-planning. We were the only team that could demonstrate the manipulation of one of the objects. For this, the interplay between a specific design of the manipulator and its gripper, fast object perception, and versatile grasp and motion planning have been important. While teleoperated manipulation was not shown by us in the competition run, we have been one of the few teams that could demonstrate this capability on-site to the public. In future work, the situational awareness of the operator needs to be improved, e.g., using head-mounted and 3D display. The creation of motion plans for the virtual robot could be made more intuitive through better user interfaces such as 3D space mice or haptic displays.

Our system was very competitive and could show many functionalities that are required by the competition. Unfortunately, while our control station interface provides operational status of most system components, the specific failure of the laser scanner was not displayed. This hints us to the necessity of a generic approach to automatic failure detection and recovery in our system. The general performance of all teams at the competition demonstrated that much more research and development needs to be invested to fully solve the complex problem defined by the DLR SpaceBot Cup.

9 Conclusion

In this article, we presented an integrated robot system for exploration and mobile manipulation in rough terrain. While our system can handle many parts of the mission autonomously, all aspects from direct control to mission specification can be teleoperated through intuitive operator interfaces. Because we realized a robust communication layer on top of ROS, our system also copes well with degraded network communication between the robot and the operator station.

Our wheeled robot has been designed for driving in rough terrain. It is equipped with a multitude of sensors such as a continuously rotating 3D laser scanner, an omnidirectional RGB-D camera consisting of eight individual cameras, and three HD wide-angle cameras. Autonomous navigation in rough terrain is tackled in a two-level approach. On the egocentric level, the robot drives to local waypoints within the view of the omnidirectional RGB-D camera. It plans obstacle-avoiding paths, and finds smooth driving controls. This level also supports teleoperation to select and drive to waypoints in the local surrounding of the robot.

On the allocentric level, a 3D map of the environment is built through registration and SLAM of 3D laser scans of the environment. The robot localizes by fusing wheel odometry, IMU measurements, and visual odometry with observations of its pose of the incrementally build environment map. Exploration missions can be specified as a list of waypoints in a coarse environment map.

For object manipulation, the robot detects objects in front and plans feasible grasps. It executes the grasps using parametrized trajectories, and stores them in specialized holders. Teleoperation is possible by specifying trajectory waypoints in an external 3D visualization of the robot and its perceived environment.

The integrated system has been demonstrated at the DLR SpaceBot Cup 2013. During the official run, the robot autonomously explored and navigated in the arena. We also conducted isolated experiments to show the quality of our approaches. Our SLAM system is capable of building a visually accurate map of the environment. Our local navigation approach finds smooth obstacle-avoiding paths, supported by high frame-rate processing of the images of all eight RGB-D cameras.

In future work, we will further improve the robot design to operate in more difficult rough terrain. For instance, a legged robot could traverse through debris fields. Also, the use of the manipulator to open passages by moving debris, or to open doors could be investigated. A dexterous hand or a second arm would allow the robot to perform more complex manipulation tasks such as tool-use. For improving shared autonomy, automatic failure detection could be integrated into the system, such that the robot reports failures and recommends a suitable semi-autonomous control mode for recovery.

References

Anderson, S. and Barfoot, T. D. (2013). Towards relative continuous-time SLAM. In *Robotics and Automation (ICRA), IEEE International Conference on.*

Ballantyne, G. H. and Moll, F. (2003). The da Vinci telerobotic surgical system: The virtual operative field and telepresence surgery. *Surg Clin North Am*, 83(6):1293–304, vii.

Berenson, D. and Srinivasa, S. (2008). Grasp synthesis in cluttered environments for dexterous hands. In *Humanoid Robots (Humanoids), IEEE-RAS International Conference on.*

Biesiadecki, J. J., Leger, C., and Maimone, M. W. (2007). Tradeoffs between directed and autonomous driving on the mars exploration rovers. *Int. Journal of Robotics Research*, 26(1):91–104.

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 *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1597 –1598.

Bosse, M. and Zlot, R. (2009). Continuous 3D scan-matching with a spinning 2D laser. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4312–4319.

Chen, Z. (2003). Bayesian filtering: From Kalman filters to particle filters, and beyond. *Statistics*, pages 1–69.

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

Deegan, P., Thibodeau, B., and Grupen, R. (2006). Designing a self-stabilizing robot for dynamic mobile manipulation. In *Robotics: Science and Systems - Workshop on Manipulation for Human Environments*, Philadelphia, Pennsylvania.

Droeschel, D., Stückler, J., and Behnke, S. (2014). Local multi-resolution surfel grids for MAV motion estimation and 3D mapping. In *Intelligent Autonomous Systems (IAS), 13th Int. Conference on*.

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

Ferguson, D. and Likhachev, M. (2008). Efficiently using cost maps for planning complex maneuvers. In *Proc. of the ICRA Workshop on Planning with Cost Maps*.

Gerkey, B. P. and Konolige, K. (2008). Planning and control in unstructured terrain. In *Proc. of the ICRA Workshop on Path Planning on Costmaps*.

Gingras, D., Lamarche, T., Bedwani, J.-L., and Dupuis, É. (2010). Rough terrain reconstruction for rover motion planning. In *Computer and Robot Vision (CRV), Canadian Conference on*, pages 191–198. IEEE.

Guizzo, E. and Deyle, T. (2012). Robotics trends for 2012. *IEEE Robotics and Automation Magazine*.

Hagn, U., Konietschke, R., Tobergte, A., Nickl, M., Jörg, S., Kübler, B., Passig, G., Gröger, M., Fröhlich, F., Seibold, U., Tien, L. L., Albu-Schäffer, A., Nothhelfer, A., Hacker, F., Grebenstein, M., and Hirzinger, G. (2010). DLR MiroSurge: a versatile system for research in endoscopic telesurgery. *Int. J. Computer Assisted Radiology and Surgery*, 5(2):183–193.

Hebert, M. and Vandapel, N. (2003). Terrain classification techniques from ladar data for autonomous navigation. *Robotics Institute*, page 411.

Holz, D. and Behnke, S. (2014). Mapping with micro aerial vehicles by registration of sparse 3D laser scans. In *Intelligent Autonomous Systems (IAS), 13th International Conference on*.

Holz, D., Holzer, S., Rusu, R., and Behnke, S. (2012). Real-time plane segmentation using RGB-D cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *LNCS*, pages 306–317. Springer.

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

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.

Hsiao, K., Chitta, S., Ciocarlie, M., and Jones, E. G. (2010). Contact-reactive grasping of objects with partial shape information. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2011). Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*.

Kewlani, G., Ishigami, G., and Iagnemma, K. (2009). Stochastic mobility-based path planning in uncertain environments. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 1183–1189.

Klaess, J., Stueckler, J., and Behnke, S. (2012). Efficient mobile robot navigation using 3D surfel grid maps. In *Proc. German Conf. on Robotics (ROBOTIK)*.

Kron, A., Schmidt, G., Petzold, B., Zah, M., Hinterseer, P., and Steinbach, E. (2004). Disposal of explosive ordnances by use of a bimanual haptic telepresence system. In *Robotics and Automation (ICRA), IEEE International Conference on*.

Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 3607–3613.

Kuemmerle, R., Triebel, R., Pfaff, P., and Burgard, W. (2007). Monte Carlo localization in outdoor terrains using multi-level surface maps. In *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*.

Kwak, J., Pivtoraiko, M., and Simmons, R. (2008). Combining cost and reliability for rough terrain navigation. In *9th Int. Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*.

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

Maddern, W., Harrison, A., and Newman, P. (2012). Lost in translation (and rotation): Fast extrinsic calibration for 2D and 3D LIDARs. In *Robotics and Automation (ICRA), IEEE International Conference on*.

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

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics, Special Issue on Space Robotics*, 24.

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

Miller, A. T., Knoopt, S., Christensen, H. I., and Allent, P. K. (2003). Automatic grasp planning using shape primitives. In *Robotics and Automation (ICRA), IEEE International Conference on.*

Murphy, R. R. (2014). *Disaster Robotics.* MIT Press.

Nieuwenhuisen, M., Droeschel, D., Holz, D., Stückler, J., Berner, A., Li, J., Klein, R., and Behnke, S. (2013). Mobile bin picking with an anthropomorphic service robot. In *Robotics and Automation (ICRA), IEEE International Conference on.*

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

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.

Rusu, R. B., Holzbach, A., Blodow, N., and Beetz, M. (2009). Fast geometric point labeling using conditional random fields. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on.*

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

Schadler, M., Stückler, J., and Behnke, S. (2014). Rough terrain 3D mapping and navigation using a continuously rotating 2D laser scanner. *German Journal on Artificial Intelligence (KI)*, 28.

Schenker, P., Huntsberger, T., Pirjanian, P., Baumgartner, E., and Tunstel, E. (2003). Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization. *Autonomous Robots*, 14(2-3):103–126.

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

Seelinger, M., Yoder, J.-D., and Baumgartner, E. T. (2012). Autonomous go-and-touch exploration (AGATE). *Journal of Field Robotics*, 29(3):469–482.

Stilman, M., Wang, J., Teeyapan, K., and Marceau, R. (2009). Optimized control strategies for wheeled humanoids and mobile manipulators. In *Humanoid Robots (Humanoids), IEEE-RAS International Conference on*, pages 568–573.

Stoyanov, T. and Lilienthal, A. (2009). Maximum likelihood point cloud acquisition from a mobile platform. In *Advanced Robotics (ICAR), Int. Conf. on*.

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

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

Stückler, J., Steffens, R., Holz, D., and Behnke, S. (2012). Efficient 3D object perception and grasp planning for mobile manipulation in domestic environments. *Robotics and Autonomous Systems*.

Tadokoro, S., editor (2010). *Rescue Robotics – DDT Project on Robots and Systems for Urban Search and Rescue*. Springer.

Triebel, R., Pfaff, P., and Burgard, W. (2006). Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

Whitcomb, L. (2000). Underwater robotics: out of the research laboratory and into the field. In *Robotics and Automation (ICRA), IEEE International Conference on*.