14th International Conference on Intelligent Autonomous Systems (IAS), Shanghai, China, July 2016.

Supervised Autonomy for Exploration and Mobile Manipulation in Rough Terrain

Max Schwarz, Sebastian Schüller, Christian Lenz,
David Droeschel, and Sven Behnke

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 robots that are capable to operate in unknown environments without direct human interaction. 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 our mobile manipulation robot Momaro. The robot perceives and maps previously unknown, uneven terrain using a 3D laser scanner. We assess drivability and plan navigation for the omnidirectional drive. Using its four legs, Momaro adapts to the slope of the terrain. It perceives objects with cameras, estimates their pose, and manipulates them with its two arms autonomously. For specifying missions, monitoring mission progress, and on-the-fly reconfiguration, we developed suitable operator interfaces. With the developed system, our team NimbRo Explorer solved all tasks of the DLR SpaceBot Camp 2015.

1 Introduction

In planetary exploration scenarios, robots are needed that are capable of operating autonomously in unknown environments and highly unstructured and unpredictable situations. To address this need, the German Aerospace Center (DLR) held the DLR SpaceBot Camp 2015[1]. The robots needed to tackle the following tasks: i) find and identify three previously known objects in a planetary-like environment (cup, battery, and base station); ii) take a soil sample of a previously known spot (optional); iii) pick up and deliver the cup and the battery to the base station; and iv) assemble all objects.

All tasks had to be completed in 60 min as autonomously as possible, including perception, manipulation and navigation in difficult terrain. A coarse height map with 50 cm resolution of the environment was known prior to the run. No line-of-sight between the robot and the crew was allowed and communication between the robot and the operators was restricted by a round trip latency of 4 s and scheduled blackouts.

To address these tasks, we used the mobile manipulation robot Momaro (see Fig. 1), which is configured and monitored from a ground station. Momaro is

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

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

equipped with four articulated compliant legs that end in pairs of directly driven, steerable wheels. This flexible locomotion base allows it to drive on suitable terrain and to make steps when required to overcome obstacles. Momaro has an anthropomorphic upper body with two 7 degrees of freedom (DOF) arms that end in dexterous grippers. Through adjustable base-height and attitude and a yaw joint in the spine, our robot has a work space equal to the one of an adult person.

Momaro is equipped with a 3D laser scanner, multiple color cameras, an RGB-D camera, and a fast onboard computer. The robot communicates to a relay at the landing site via WiFi and is powered by a rechargeable LiPo battery.

The developed system was tested successfully at the DLR SpaceBot Camp 2015. In this paper, we report on the robust perception, state estimation, navigation, and manipulation methods that we developed for exploration and mobile manipulation in rough terrain with supervised autonomy, i.e. autonomous robot operation under supervision of a human operator crew, which can configure and monitor the operation on a high level.

2 Related Work

The need for mobile manipulation has been addressed 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 groups use purely wheeled locomotion for their robots, e.g. [1,2]. In previous work, we developed NimbRo Explorer [3], a six-wheeled robot equipped with a 7 DOF arm designed for mobile manipulation in rough terrain encountered in planetary exploration scenarios. Compared to wheeled robots, legged robots are more complex to design, build, and control, but they have obvious mobility advantages when operating in unstructured terrains and

environments, see e.g. [4,5]. Some 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, e.g. [6,7].

In 2013, DLR held a very similar SpaceBot competition which encouraged several robotic developments [8]. Heppner et al. [9] 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. Sünderhauf et al. [10] developed a cooperative team of two wheeled robots, which had good driving capabilities, but failed due to communication issues. Schwendner et al. [11] developed the six-wheeled Artemis rover able to passively cope with considerable terrain slopes (up to 45°). In contrast, Momaro employs active balancing strategies (see Sec. 5.3).

In our previous work [3], we describe the Explorer robot used in the 2013 competition and its local navigation system [12]. Compared to the 2013 system, we improve on i) capabilities of the mechanical design (e.g. execution of stepping motions and bimanual manipulation); ii) degree of autonomy (autonomous execution of full missions, including assembly tasks at the base station); iii) situational awareness of the operator crew; and iv) 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 robots, Momaro can drive 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 [13,14].

Supervised autonomy has been proposed by Cheng et al. [15], 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 for the large latencies of space communication. Gillett et al. [16] use supervised autonomy for an unmanned satellite servicing system that must perform satellite capture autonomously. The survey by Pedersen et al. [17] highlights the 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 of sensible autonomy in space exploration.

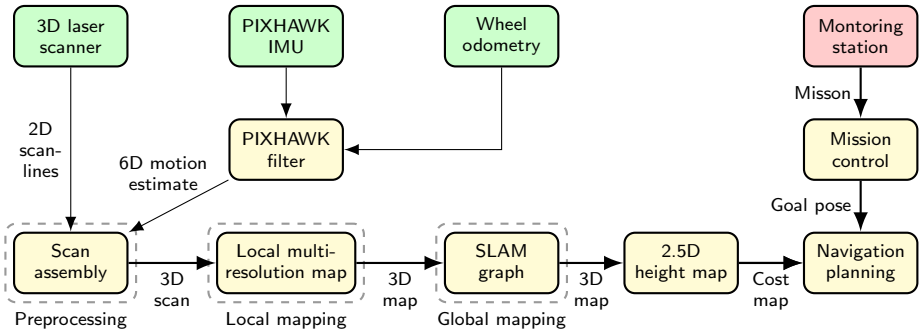

Fig. 2. Overview of mapping, localization, and navigation. Laser-range measurements are processed (Sec. 3.1). Scans are registered with and stored in a local multiresolution map (Sec. 3.2). Keyframe views of local maps are registered against each other in a SLAM graph (Sec. 3.3). A 2.5D height map is used to assess drivability. A 2D grid-based approach is used for planning (Sec. 5).

3 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 [18]. 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. 2.

3.1 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

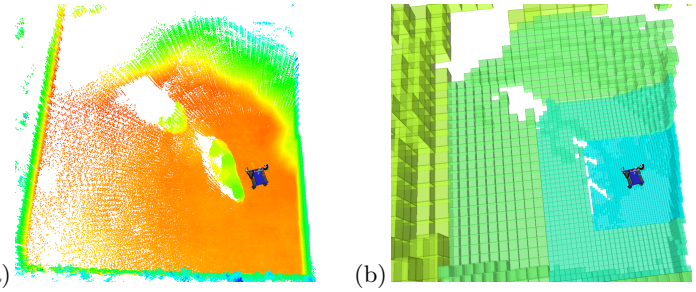

Fig. 3. Local multiresolution map. (a) 3D points stored in the map on the robot. Color encodes height. (b) Cell size increases with the distance from robot.

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.

3.2 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. 3.

We use a hybrid representation, storing 3D point measurements along with occupancy information in each cell. Similar to [19], 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 [18] designed for this data structure. It leverages the multiresolution property of the map and gains efficiency by summarizing 3D points to surfels that are then used for registration. Measurements from the aligned 3D scan replace older measurements in the map and are used to update the occupancy information.

3.3 Allocentric Mapping

To estimate the motion of the robot, we incorporate IMU measurements, wheel odometry, and 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 continuing localization. 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 [20]. 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 Sec. 3.2, 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^2o framework [21], yielding maximum likelihood estimates of the view poses.

3.4 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 the entire 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 robot pose. 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

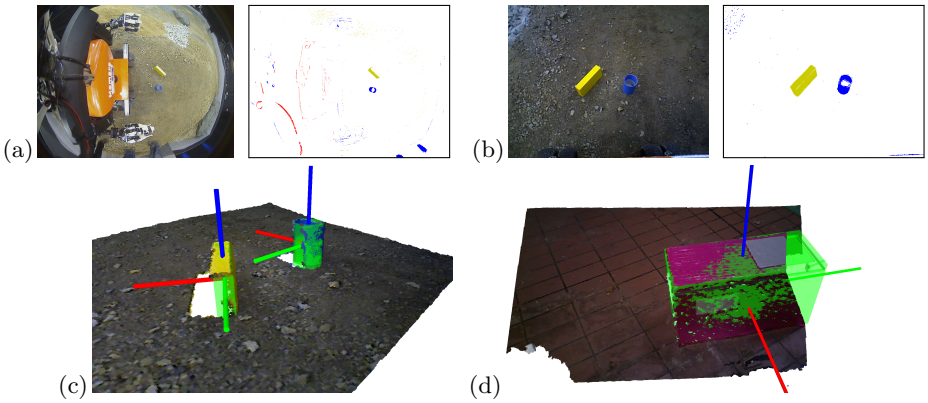

Fig. 4. Object perception. Color segmentation using (a) wide-angle camera and (b) RGB-D camera; RGB-D point clouds showing registered (c) cup and battery and (d) base station. The registered models are shown in green.

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 specified in the coarse height map by localizing towards our pose graph.

3.5 Height Mapping

As a basis for assessing drivability, the 3D map is projected into a 2.5D height map, shown in Fig. 5. 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 a 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 local histograms [22]. The height map is suitable for navigation planning (see Sec. 5).

4 Object Perception

For approaching objects and adapting motion primitives to detected objects, RGB images from a wide-angle camera and RGB-D point clouds from an Asus Xtion camera, both mounted on the sensor head, are used. We differentiate between object detection (i.e. estimating an approximate 3D object position) and object registration (i.e. determining an accurate 6D object pose).

The objects provided by DLR are color-coded. We classify each pixel by using a precomputed lookup table in YUV space. When approaching an object, object detection is initially performed with the downwards-facing wide-angle camera (Fig. 4a). 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 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 (Fig. 4b).

When the object is close enough, we use registration of a CAD model to obtain a precise object pose (Fig. 4c,d). 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 [23] to obtain object clusters. The object models are then registered to the clusters using Generalized ICP [24]. The estimated object 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. 4).

5 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.

5.1 Local Height Difference Maps

Since caves and other overhanging structures are the exception on most planetary surfaces, the 2.5D height map generated in Sec. 3.5 suffices for autonomous navigation planning.

The 2.5D height map is transformed into a multi scale height difference map. For each cell, we calculate local height differences 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

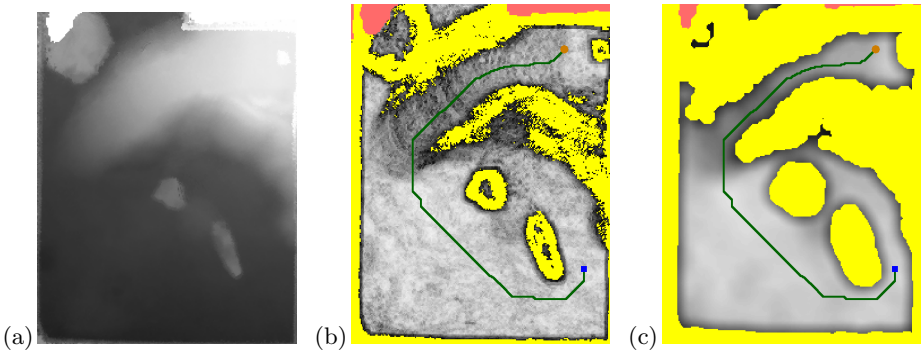

Fig. 5. Navigation planning. (a) 2.5D height map generated by projecting the 3D map. (b) Calculated traversability costs for each cell. (c) Inflated costs used for A* path planning. The orange dot represents the current robot position, the blue square the target position. Yellow regions represent absolute obstacles, red regions indicate missing measurements.

$D_l(x, y)$:=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.

5.2 Path Planning

During the SpaceBot Camp, we used the standard ROS `navfn`[2] 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 values for the λ_l parameters were found empirically: $\lambda_1 = 2.2, \lambda_2 = 3.6, \lambda_3 = 2.5$.

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

[2] `http://wiki.ros.org/navfn`

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. Then for all other cells, we calculate local averages to produce costs that increase gradually close to obstacles:

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

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

Fig. 5 shows a planned path on the height map acquired during our mission at the SpaceBot Camp.

Local Trajectory Rollout The found global path needs to be executed by driving omnidirectionally on a local scale. To this end, we use the standard ROS `dwa_local_planner`[3] package, which is based on the Dynamic Window Approach [25]. The `dwa_local_planner` accounts for the robot foot print, so cost inflation is not needed.

During navigation, the global plan is updated every 4 s, while the local rollout is re-evaluated with 3 Hz to perform the necessary adaptions to robot movement and environment changes. We also added a simple recovery behavior that first warns the operator crew that the robot is stuck and then executes a fixed backward driving primitive after a timeout expires without operator intervention.

5.3 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

$$e_b = \frac{1}{1 + K_s \cdot K_p} \cdot w - \frac{K_{sz}}{1 + K_s \cdot K_p} \cdot z. \tag{5}$$

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 overcome inclines even greater than 20° without pitching over.

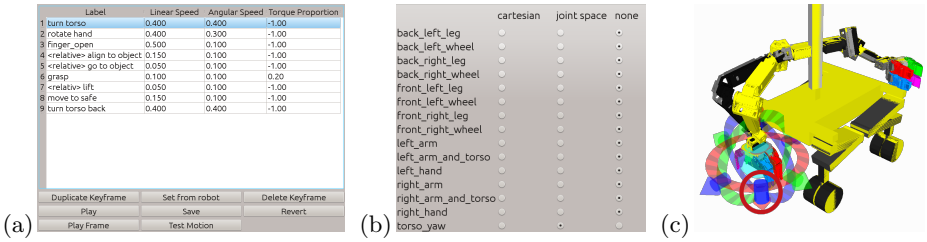

Fig. 6. Keyframe Editor GUI. (a) Motions are designed step by step and can be absolute or relative to perceived objects. (b) The user can select which joint groups are included in the currently edited keyframe and if interpolation between keyframes is Cartesian or joint space. (c) The real position of 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. A model of the cup is placed in front of the robot to assist designing relative motions.

Table 1. Custom Motions for the DLR SpaceBot Camp 2015.

Motion	Purpose	Reference type
scoop	fill scoop tool with soil sample	absolute
fill cup	pour soil into cup and discard tool	relative to cup
grasp cup right hand	grasp cup with right hand from above	relative to cup
grasp battery left hand	grasp battery with left hand from above	relative to battery
place cup	place cup on base station	relative to base
place battery	put battery into base station	relative to base
toggle switch	toggle switch on side of base station	relative to base
grasp abort {left,right}	move to initial position	absolute
reset {left,right} arm	move all arm joints in defined position	absolute
reset torso	move torso into initial position	absolute
cheer	cheer to the audience	absolute

6 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 6. 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 1.

[3] `http://wiki.ros.org/dwa_local_planner`

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 $T_{\text{perceived}}$:

$$T_{\text{relative}} = T^{(i)}_{\text{perceived}} \left(T_{\text{reference}}\right)^{-1} T^{(i)}_{\text{endeffector}}. \tag{6}$$

Fig. 7 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 Sec. 4, 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.

7 Evaluation

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[4]. 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 DLR SpaceBot Camp.

The SpaceBot Camp required participants to solve mapping, locomotion, and manipulation tasks in rough terrain. In detail, a cup and a battery had to be located and collected on the planetary surface. If possible, the robot had to take a soil sample at a specific location and fill the cup with it. Next, the robot had to carry both objects to a base station object. The cup had to be placed on a scale

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

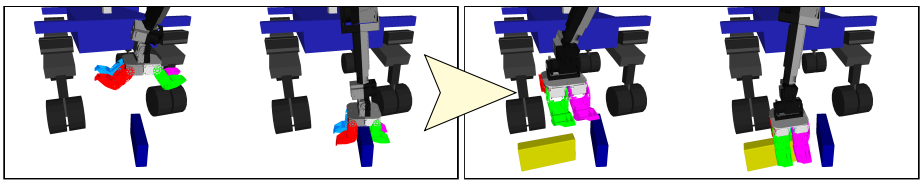

Fig. 7. Grasping objects using keyframe transformation. Left: The blue reference object is grasped as the primitive was designed in the Keyframe Editor. Right: The primitive is automatically adapted to the perceived pose of the yellow object.

located on the base station, and the battery had to be inserted into a slot on the side. By operating a switch on the other side, the base station was switched on. The participants were provided with a coarse map of the environment that had to be refined by the robot's mapping system. The communication link to the operator crew was severely constrained both in latency (2 s per direction) and in availability.

7.1 Mapping and Self-localization

Consisting of different types of stones, sand, and soil, the planetary-like environment was specially challenging for the mapping system—causing slip in odometry and vibrations of robot and sensor.

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. 7.1. 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. 11,

Although our mapping system showed very robust and reliable performance in this environment, there was 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. The 3D scan distorted by the motion caused spurious measurements in the map. The operators decided to clear the SLAM 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.

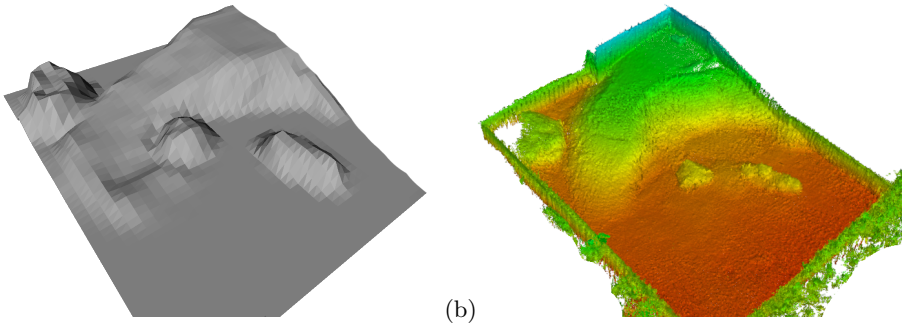

Fig. 8. Map refinement. (a) Provided coarse map of the SpaceBot Camp 2015 arena. (b) The resulting global map from data acquired during the competition.

7.2 Navigation System

While preparing for the SpaceBot Camp, we learned that our pitch stabilization control method works even under extreme conditions. Being able to reliably overcome obstacles 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. 10).

7.3 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. Thus, it could happen that Momaro was not able to push the battery entirely inside. 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 used two steps. First, the battery is pushed in as far as possible until the hand touches the base station. Then we release the clamped battery in the slot. Afterwards, we close the hand and push the battery inside with parts of the wrist and proximal fingers.

Overall, our straightforward keyframe adaption approach proved 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.

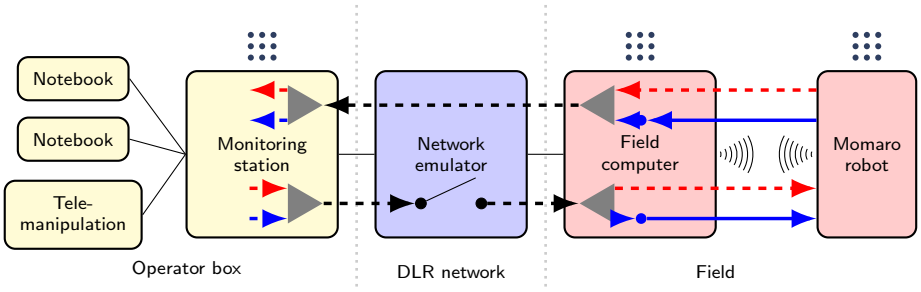

Fig. 9. Communication architecture. Operator components are shown in yellow, DLR-provided components in blue, field network components in red. Solid black lines represent physical network connections. Thick lines show the different data channels (dotted: UDP, solid: TCP). ⁘ indicates a ROS master. Streaming links are colored red, message links are shown in blue.

Table 2. 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 (incl. locomotion)	0:00	20:25	**20:25**

7.4 Full System Performance at DLR SpaceBot Camp 2015

Momaro solved all tasks of the SpaceBot Camp with supervised autonomy. Fig. 9 illustrates the communication architecture. Fig. 10 shows the operator interface used for mission planning. Our team was the only one to demonstrate all tasks including the optional taking of a soil sample. Fig. 11 gives an overview of the sequence of performed tasks. A video of our performance can be found online[5]. See Fig. 11 for detailed images of the subtasks. Timings are listed in Tab. 2.

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 a 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. 11). Although the operators paused autonomous navigation

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

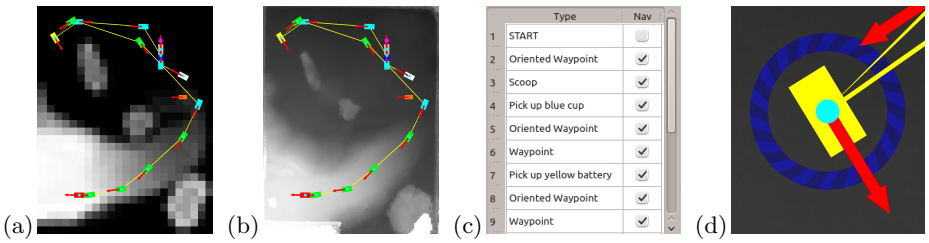

Fig. 10. Mission planning. (a) Mission plan on coarse height map provided by DLR. (b) Mission plan on detailed height map generated from the SLAM map. (c) List representation of the first eight poses. The "Nav" column can be used to disable navigation (e.g. start grasping an object immediately). (d) 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.

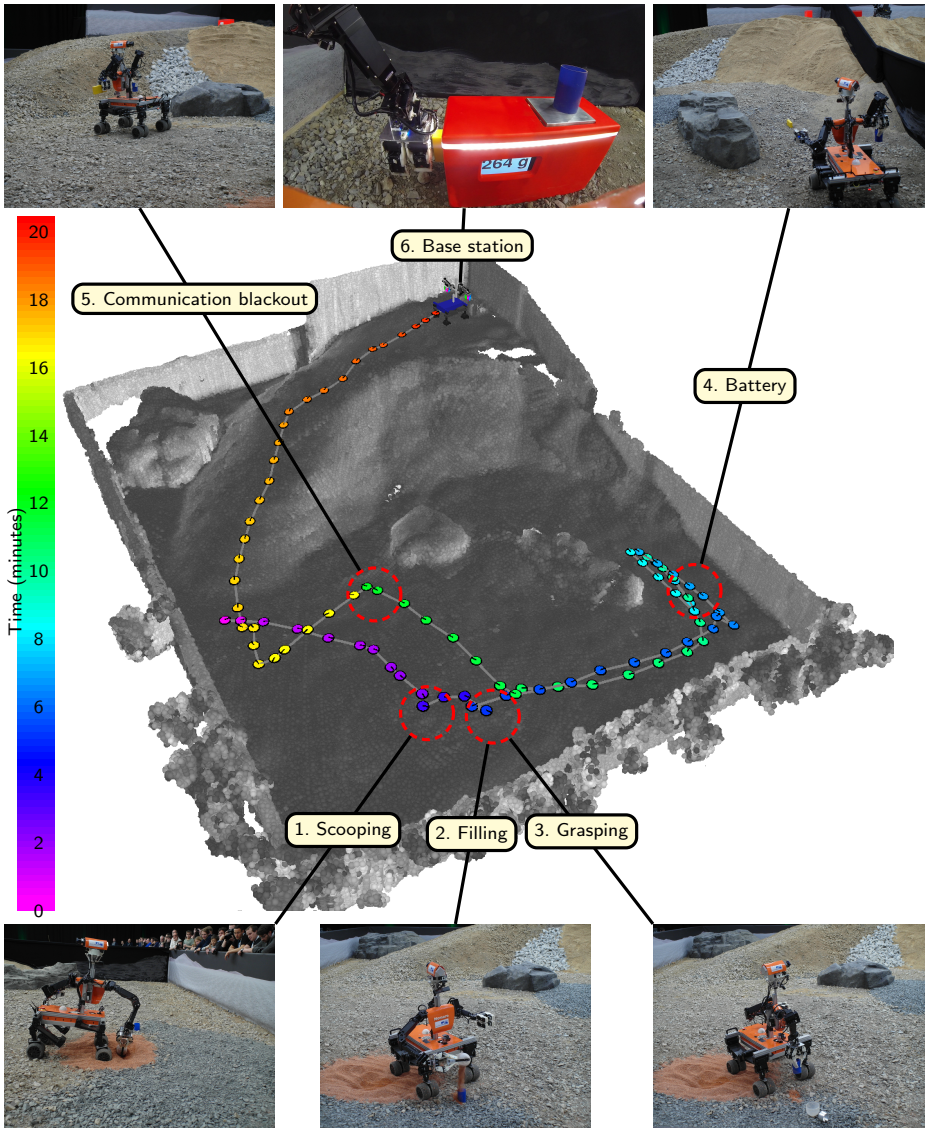

Fig. 11. Overview of the executed mission at SpaceBot Camp. The mission starts by scooping the soil sample, filling it into the cup and grasping the cup, then locating and grasping the battery pack. After waiting until the end of scheduled communication blackout, the mission is concluded by the base station assembly.

at one point on the slope to assess the situation, no intervention was necessary and navigation resumed immediately.

8 Conclusion

In this article, we presented the mobile manipulation robot Momaro and its ground station. We detail the soft- and hardware architecture of the integrated robot system and motivate design choices. The feasibility, flexibility, usefulness, and robustness of our design have been demonstrated 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 built 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 realized 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 and object assembly using parametrized motion primitives.

References

1. Mehling, J., Strawser, P., Bridgwater, L., Verdeyen, W., Rovekamp, R.: Centaur: NASA's mobile humanoid designed for field work. In: Proc. of ICRA. (2007)
2. 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., Hirzinger, G.: Rollin' Justin-mobile platform with variable base. In: ICRA. (2009)
3. Stückler, J., Schwarz, M., Schadler, M., Topalidou-Kyniazopoulou, A., Behnke, S.: NimbRo Explorer: Semiautonomous exploration and mobile manipulation in rough terrain. Journal of Field Robotics (JFR) **33** (2016) 407–558
4. Semini, C., Tsagarakis, N., Guglielmino, E., Focchi, M., Cannella, F., Caldwell, D.: Design of HyQ–A hydraulically and electrically actuated quadruped robot. J. of Systems and Control Engineering **225** (2011) 831–849
5. Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., Abeles, P., Stephen, D., Mertins, N., Lesman, A., et al.: Team IHMC's lessons learned from the DARPA robotics challenge trials. J. o. Field Robotics **32** (2015) 192–208
6. Endo, G., Hirose, S.: Study on roller-walker (multi-mode steering control and self-contained locomotion). In: ICRA. (2000)

7. Halme, A., Leppänen, I., Suomela, J., Ylönen, S., Kettunen, I.: WorkPartner: Interactive human-like service robot for outdoor applications. Int. Journal of Robotics Research (IJRR) **22** (2003) 627–640
8. Kaupisch, T., Noelke, D., Arghir, A.: DLR spacebot cup — Germany's space robotics competition. In: Proc. of the Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA). (2015)
9. Heppner, G., Roennau, A., Oberländer, J., Klemm, S., Dillmann, R.: Laurope – six legged walking robot for planetary exploration participating in the SpaceBot Cup. In: WS on Advanced Space Technologies for Robotics and Automation. (2015)
10. Sünderhauf, N., Neubert, P., Truschzinski, M., Wunschel, D., Pöschmann, J., Lange, S., Protzel, P.: Phobos and Deimos on Mars–two autonomous robots for the DLR SpaceBot Cup. In: 12th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space-i-SAIRAS. (2014)
11. Schwendner, J., Roehr, T.M., Haase, S., Wirkus, M., Manz, M., Arnold, S., Machowinski, J.: 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. (2014)
12. Schwarz, M., Behnke, S.: Local navigation in rough terrain using omnidirectional height. In: Proc. of the German Conference on Robotics (ROBOTIK), VDE (2014)
13. Rodehutskors, T., Schwarz, M., Behnke, S.: Intuitive bimanual telemanipulation under communication restrictions by immersive 3d visualization and motion tracking. In: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids). (2015)
14. Schwarz, M., Rodehutskors, T., Schreiber, M., Behnke, S.: Hybrid driving-stepping locomotion with the wheeled-legged robot Momaro. In: ICRA. (2016)
15. Cheng, G., Zelinsky, A.: Supervised autonomy: A framework for human-robot systems development. Autonomous Robots **10** (2001) 251–266
16. Gillett, R., Greenspan, M., Hartman, L., Dupuis, E., Terzopoulos, D.: Remote operation with supervised autonomy (rosa). In: 6th Int. Conf. on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS). (2001)
17. Pedersen, L., Kortenkamp, D., Wettergreen, D., Nourbakhsh, I.: A survey of space robotics. In: Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space. (2003) 19–23
18. Droeschel, D., Stückler, J., Behnke, S.: Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner. In: ICRA. (2014) 5221–5226
19. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: an efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots **34** (2013) 189–206
20. Droeschel, D., Stückler, J., Behnke, S.: Local multi-resolution surfel grids for MAV motion estimation and 3D mapping. In: IAS. (2014)
21. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: A general framework for graph optimization. In: ICRA. (2011)
22. Huang, T., Yang, G., Tang, G.: A fast two-dimensional median filtering algorithm. IEEE Trans. Acoust., Speech, Signal Processing **27** (1979) 13–18
23. Holz, D., Holzer, S., Rusu, R.B., Behnke, S.: Real-time plane segmentation using RGB-D cameras. In: RoboCup 2011: Robot Soccer World Cup XV. (2012) 306–317
24. Segal, A., Haehnel, D., Thrun, S.: Generalized-ICP. In: Proc. of Robotics: Science and Systems. (2009)
25. Fox, D., Burgard, W., Thrun, S., et al.: The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine **4** (1997) 23–33