The LRU Rover for Autonomous Planetary Exploration and its Success in the SpaceBotCamp Challenge

Martin J. Schuster[1], Christoph Brand[1], Sebastian G. Brunner[1], Peter Lehner[1], Josef Reill[1], Sebastian Riedel[1], Tim Bodenmüller[1], Kristin Bussmann[1], Stefan Büttner[1], Andreas Dömel[1], Werner Friedl[1], Iris Grixa[1], Matthias Hellerer[1], Heiko Hirschmüller[2], Michael Kassecker[1], Zoltán-Csaba Márton[1], Christian Nissler[1], Felix Ruess[2], Michael Suppa[2] and Armin Wedler[1]

***Abstract*— The task of planetary exploration poses many challenges for a robot system, from weight and size constraints to sensors and actuators suitable for extraterrestrial environment conditions. As there is a significant communication delay to other planets, the efficient operation of a robot system requires a high level of autonomy. In this work, we present the Light Weight Rover Unit (LRU), a small and agile rover prototype that we designed for the challenges of planetary exploration. Its locomotion system with individually steered wheels allows for high maneuverability in rough terrain and the application of stereo cameras as its main sensor ensures the applicability to space missions. We implemented software components for self-localization in GPS-denied environments, environment mapping, object search and localization and for the autonomous pickup and assembly of objects with its arm. Additional high-level mission control components facilitate both autonomous behavior and remote monitoring of the system state over a delayed communication link. We successfully demonstrated the autonomous capabilities of our LRU at the SpaceBotCamp challenge, a national robotics contest with focus on autonomous planetary exploration. A robot had to autonomously explore a moon-like rough-terrain environment, locate and collect two objects and assemble them after transport to a third object - which the LRU did on its first try, in half of the time and fully autonomous.**

I. Introduction

The Light Weight Rover Unit (LRU) [1] is particular suited for planetary exploration. This field of application challenges the design of a robot in many aspects. Economic transportation to the planet forces the rover to be light. After arriving at the planet's surface, all sensors and actuators need to work under the alien conditions. Even when faced with heavy communication delay and blackouts, the ground station team must be able to interact with the rover on a high level. As the delay renders teleoperation inefficient, the rover has to solve most tasks autonomously. It has to navigate unknown, rough terrain to explore the area and arrive at scientifically relevant locations. There, the rover has to manipulate the environment to take samples or to assemble technical equipment. We designed the LRU to cope with the challenges of planetary exploration. Its unique construction is

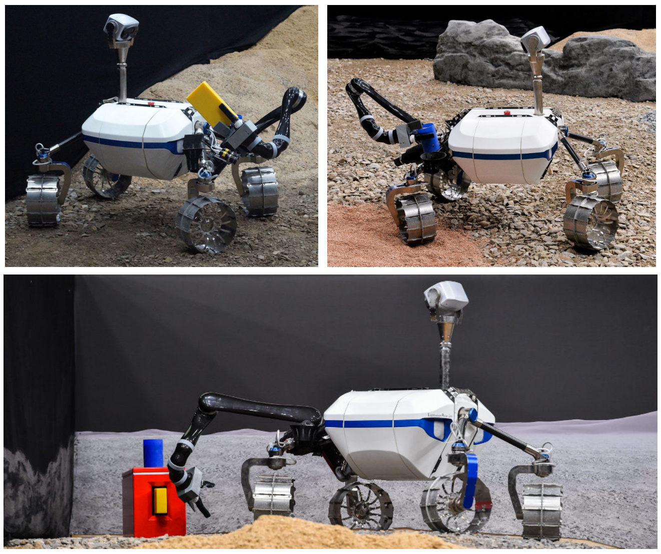

Fig. 1. The Light Weight Rover Unit (LRU) picks up the battery (yellow, top left) and the sample (blue container, top right) and assembles them at the base station (red, bottom) during the SpaceBotCamp challenge.

particularly light weight (approx. 40 kg) and thus economic to transport into space. The LRU only relies on sensor concepts (stereo cameras, inertial measurement unit) which work in alien conditions and are currently employed in space missions [2]. The LRU's locomotion system can drive over rough terrain and is highly maneuverable due to its four independent wheels, each of them having individual steering and driving motors. A force-controlled manipulator on the back of the rover picks up and assembles objects. The autonomy of the LRU stems from a variety of software components. We developed and integrated modules for on-board self-localization in GPS-denied environments, local and global mapping, fast obstacle avoidance, path planning, object detection and pose estimation, manipulation, inter-process communication, high-level task control as well as for a ground station mission control. We evaluated the LRU's capabilities in the official SpaceBotCamp challenge, which posed typical challenges of a planetary exploration mission. The robot had to find two known objects in an unknown moon-like rough terrain and assemble them at a base station. The rover had to fulfill these tasks in a single run and communication to the rover was heavily delayed as well as

[1] German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchner Str. 20, 82234 Wessling, Germany, {firstname.lastname}@dlr.de
[2] Roboception GmbH, Kaflerstr. 2, 81241 Munich, Germany, {firstname.lastname}@roboception.de

for most of the time unidirectional, restricting the ground station team to merely monitoring the system. Therefore the rover needs to solve the tasks autonomously, meaning that it has to cope with an unknown environment using its on-board sensor data only. In this work, we present the LRU and explain how it solved the challenge under all these constraints fully autonomously and successfully completed the mission in a single run in just thirty minutes, half of the given time.

II. Related Work

The experiences gathered with the successful and ongoing Mars exploration rover missions like MER [3] and MSL [2] as well as earlier considerations on rover autonomy [4] clarify requirements and space-suitable options regarding hardware as well as software components. Autonomous navigation solutions for unstructured and unknown environments taking robot safety, resource management (e.g. power consumption) and general robustness into account are available in many field robotic systems tested on earth [5], [6], [7], [8]. Similar to [5], we employ passive stereo cameras as a space-suitable sensor setup similar to current Mars rovers [3], [2]. Due to the sensor-specific noise characteristics, this however makes navigation and mapping more challenging compared to LIDAR-based systems [6], [7], [8], which allow high-precision measurements within a longer range of distances. In the light of unreliable and delayed communication channels, high-level autonomy, adaptation and failure recovery are desirable but pose significant challenges on the algorithmic and conceptual level. While mission-level reasoning and task planning allows for flexible online adaptation [7], we follow an approach based on modular but pre-defined control-flows. In our experience, this is more robust in case of well-defined tasks w.r.t. known objects, being typical for space missions. A predictable sequence of individual steps furthermore allows a high understandability of the robots behavior for the operating and monitoring crew.

III. System Overview

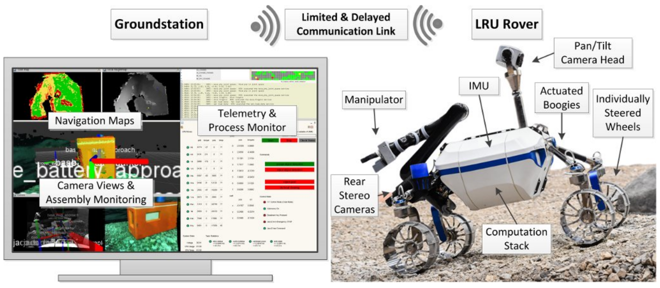
Groundstation
Limited & Delayed Communication Link
LRU Rover
Navigation Maps
Telemetry & Process Monitor
Camera Views & Assembly Monitoring
Pan/Tilt Camera Head
IMU
Manipulator
Actuated Boogies
Individually Steered Wheels
Rear Stereo Cameras
Computation Stack

Fig. 2. System overview

Our system setup, presented in Figure 2, consists of the LRU rover as well as a groundstation to monitor the robot's state and to provide remote access for shared autonomy approaches. In the SpaceBotCamp scenario, the communication link was artificially limited in bandwidth to 100 Mbit/s and delayed by two seconds in each direction, which approximates the delay between a ground station on earth and a rover

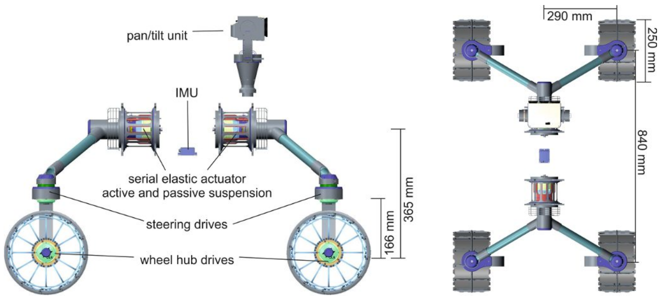

Fig. 3. LRU kinematic side and top view

on moon. In the following, we give details on the rover's hardware and present its software components in Section IV.

A. Kinematics

The LRU has a total length of 1090 mm and a total width of 730 mm. It is designed to drive in rough terrain at a maximum velocity of 1.1 m/s. In Figure 3, we give an overview over the locomotion subsystem. It consists of four individually powered and steered wheels, attached to two bogies that each make use of a serial elastic actuator (SEA). That way, the rover features active and passive suspension in both bogies. The active element allows controlling the bogie's rotational position to adjust the center of mass in order to distribute wheel load in rough terrain or improve stability on steep slopes. By use of an active-passive combination, even an active dynamic body damping is feasible. Also all components attached to the rover body benefit from that additional degree of freedom by increasing their workspace, e.g. by repositioning the attached camera beam for a better view of the environment. Each actuator in the SEA, the wheel hub and steering drives utilizes an ILM38 drive train that was designed for space applications, see Section III-B. Compared to other rover concepts, the LRU profits from its individually steered wheels, which allow for driving sideways and turning in place, thus increasing maneuverability. Furthermore, our four wheeled kinematic has advantages in terms of compactness and weight compared to six-wheeled rovers.

B. Locomotion

As for the kinematics, reliability and robustness in rough terrain are also most important for the locomotion sub-system (LSS). Due to their high peak torque and very high torque per volume and weight ratio, we use permanent magnet synchronous motors [9] in every rover joint. Two actuator unit sizes were developed to meet the different requirements. We employ the small ILM25 unit to move the pan/tilt unit and the bigger ILM38 unit for wheel traction and steering. The name indicates that it is an internal rotor motor with the number showing the diameter of the motor stator in millimeters. Originally the ILM38 unit (rated torque: 5 Nm), shown in Figure 4, has been developed for the mobile payload element rover prototype [10] and was developed further to the LRU actuator module. We employ it for traction as well as for steering and actuating the serial elastic module. To actuate the rover's pan/tilt unit [11], the concept was scaled to a smaller ILM25 module (rated torque: 2.4 Nm).

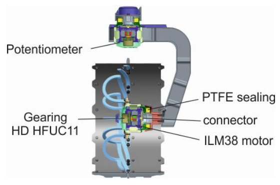

Fig. 4. Sectioned view of wheel hub and steering drive

This unit is actually used and validated in an ongoing DLR space mission called MASCOT [12], a contribution to the JAXA Hayabusa2 mission. The wheel design combines a flexible spring metal sheet running surface with a central rigid ring and thereby profits from both advantages of wide and narrow wheel types. The wheel-to-ground contact is focused on the rigid ring at hard and flat terrain, which leads to low rolling resistance. When driving in soft terrain, the flexible running surface ensures reduction of sinkage and, in combination with the grousers, gives the wheels maximum grip. We are able to adapt the stiffness of the elastic wheels by changing their number of spokes, allowing to absorb shocks directly in the wheels.

C. Manipulator

To extend the LRU's abilities for manipulating and fetching objects, a robotic arm needed to be attached to the system. As industrial developments and actual manipulators at our institute were too heavy and did not offer an appropriate workspace, we chose the JACO2 from Kinova. With a total weight of 5.7 kg, a single power supply of 24 V and a RS485 communication bus, the manipulator could be integrated at acceptable effort. The maximum load of 1.0 kg at its fully extended arm position is sufficient for grasping and manipulating our target objects. To enhance the grasping capabilities of the six degree-of-freedom manipulator, we optimized the gripper w.r.t. the shapes of the target objects. The final task in the SpaceBotCamp challenge was to fit one of the fetched objects into a base station. We chose to solve this by means of impedance control. Therefore we replaced the robot pose controller by our own control structure and only kept the Kinova motor controllers.

D. Sensor Setup

To ensure a robust and light weight sensor solution, we based our complete autonomous navigation, mapping and exploration on a single b/w stereo camera system. Its baseline of 9 cm results from the demand for precise close-range data for navigation. An additional center camera gives us color information for object detection. The cameras' fields of view are increased by using a pan/tilt mechanism that is able to pan the cameras $\pm180°$ and to tilt $\pm90°$ [11]. We employ an additional pair of stereo cameras at the back of the rover to get unobstructed, high-resolution vision data for object pose estimation and manipulation. With a baseline of 6 cm, we adjusted these cameras to the manipulator's workspace and use them only during object pickup and assembly. Furthermore, we use an IMU in the rover's body to improve its ego motion estimation through sensor fusion.

E. Computational Power

In order to build a truly autonomous system, we perform all required computation on board the LRU. We employ a standard industrial computer with an i7-3740QM CPU (2.70 GHz) and added a FPGA Spartan-6 Board. The utilization of terrestrial components for computation hardware allows faster research cycles on novel software concepts. We however decided to run the computationally intensive stereo processing on the FPGA. The corresponding VHDL code thus could also be transfered and implemented on space proof, i.e. radiation hardened, FPGAs in the future. We outsourced the control of the manipulator to an Atom processor board. This separation from the i/o-intensive image processing pipeline helped us to satisfy the controller's real-time requirements.

IV. Software Architecture

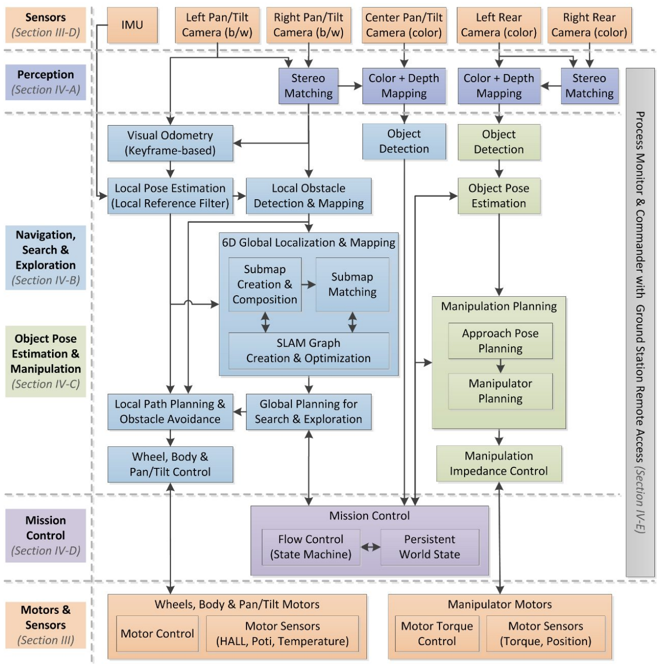
Sensors (Section III-D)
IMU
Left Pan/Tilt Camera (b/w)
Right Pan/Tilt Camera (b/w)
Center Pan/Tilt Camera (color)
Left Rear Camera (color)
Right Rear Camera (color)
Perception (Section IV-A)
Stereo Matching
Color + Depth Mapping
Color + Depth Mapping
Stereo Matching
Visual Odometry (Keyframe-based)
Object Detection
Object Detection
Local Pose Estimation (Local Reference Filter)
Local Obstacle Detection & Mapping
Object Pose Estimation
Navigation, Search & Exploration (Section IV-B)
6D Global Localization & Mapping
Submap Creation & Composition
Submap Matching
SLAM Graph Creation & Optimization
Object Pose Estimation & Manipulation (Section IV-C)
Manipulation Planning
Approach Pose Planning
Manipulator Planning
Local Path Planning & Obstacle Avoidance
Global Planning for Search & Exploration
Wheel, Body & Pan/Tilt Control
Manipulation Impedance Control
Mission Control (Section IV-D)
Mission Control
Flow Control (State Machine)
Persistent World State
Motors & Sensors (Section III)
Wheels, Body & Pan/Tilt Motors
Motor Control
Motor Sensors (HALL, Poti, Temperature)
Manipulator Motors
Motor Torque Control
Motor Sensors (Torque, Position)
Process Monitor & Commander with Ground Station Remote Access (Section IV-E)

Fig. 5. LRU software architecture: on-board key components and data flow

We present an overview of our software architecture in Figure 5. We established the data flow between our components via three different middlewares in order to satisfy their particular needs: *Links and Nodes* for real-time control and *SensorNet* for the distribution of high-bandwidth vision data, both being developed at our institute. In addition, we connect our higher-level software components via the widely used *Robot Operating System (ROS)*. On-board the LRU, we typically run more than 100 software processes in parallel, involving the execution of about 100 different libraries and components developed by more than 20 internal developers. In order to manage this complexity, we developed our own release and dependency management toolchain *RM Package Management (RMPM)* to track and deploy consistent software versions to the LRU as well as to mockup systems and simulations. In addition, we employ the process manager of

Links and Nodes to monitor process output, manage runtime dependencies and allow the compilation of mission settings by combining pre-defined modules and configurations.

A. Perception

The LRU's perception of the environment is purely vision-based. For the pair of stereo cameras in the pan/tilt head, we perform dense stereo reconstruction through Semi-Global Matching (SGM) [13] running on an on-board Spartan 6 LX75 FPGA with a resolution of 1024×508 px at 14.6 Hz. The resulting depth data is used for stereo visual odometry, obstacle avoidance and 3D environment mapping. We chose b/w cameras for our navigation stereo setup as they have a shorter exposure time as well as a higher effective resolution, which is important for the accuracy of visual odometry estimation. Thus, we map the color information of our third pan/tilt camera onto the depth data to serve as input for our object detection. The rear-facing color stereo cameras are only triggered on-demand for close-range precise 6D object pose estimation. In order to achieve an accurate mapping of the color to depth data as well as to obtain reliable object localization for grasping, the transformations between the stereo and color camera as well as between the rear cameras and the gripper have to be determined. We automated this process, called *hand-eye calibration* [14], by attaching a calibration pattern to the LRU's gripper to move it to pre-defined poses within the cameras' fields of view.

B. Navigation, Search and Exploration

1) Self-Localization and Environment Mapping: For self-localization, we use the stereo data from the pan/tilt camera head to compute visual odometry [15]. For a robust, real-time local pose estimation, we fuse these estimates with IMU measurements in a local reference filter [16] that is realized as a keyframe-based Extended Kalman Filter (EKF) with time-delay compensation [17]. We designed our mapping framework to allow the LRU to operate in GPS-denied, previously unknown indoor as well as rough-terrain outdoor environments. It combines fast local mapping for obstacle avoidance with a submap-based online global mapping approach to create a consistent 3D environment model for search and exploration. As a first step, we perform a fast stereo-error adaptive obstacle and terrain classification on the depth images from the pan/tilt stereo cameras [18]. The resulting cost map is directly used for local path planning. In addition, we integrate the full 3D stereo data, including the obstacle classification results, into submaps by aggregating it along the trajectories estimated by our local reference filter. We always switch the filter's frame of reference into the origin of the current submap in order to maintain consistency and numerical stability within the filter as well as to allow for a more accurate integration of the filter's uncertainty estimates into our overlying SLAM system. For online global optimization of pose and map estimates, we add the submap origins as nodes to a SLAM graph, connect them via the filter estimates and then run the iSAM2 [19] incremental least-squares error minimization after each change of the graph.

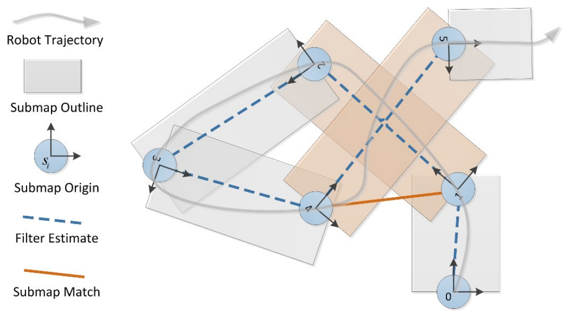

Fig. 6. Schematic of SLAM graph for our submap-based global mapping. The highlighted rectangle represent overlapping submaps that match, resulting in a loop closure constraint that is added to the graph.

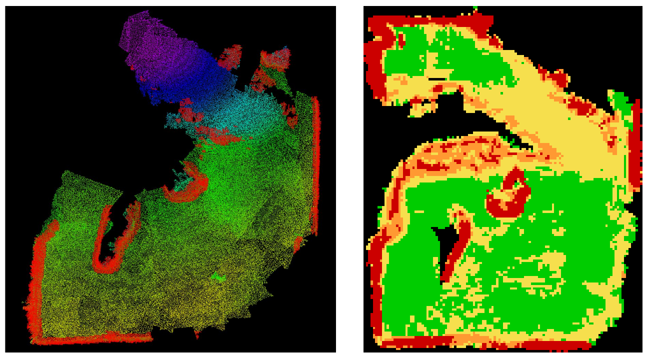

Fig. 7. Orthographic top-down views of 3D point cloud map (5 cm resolution, height colored, obstacles in red) at the middle of our SpaceBotCamp run (left) and 2.5D terrain classification map at the end of our run (right).

The combination of a local reference filter and incremental graph SLAM allows us to benefit from their particular advantages: The filter provides real-time, long-term stable state estimation for control and fast obstacle avoidance while the online graph optimization provides global pose and map estimates [20]. In order to generate loop closure constraints, we perform 6D map matching using the obstacle classifications and the 3D submap data to compute relative transformations between pairs of submaps [21]. Based on the estimated transform between two submaps we choose only potentially matching pairs for the registration instead of a brute force approach. We thereby rely on 3D geometric features as they are more robust to changing viewpoints and light conditions than 2D image features. After a final ICP refinement and outlier filtering, we integrate the resulting 6D transformation into the SLAM graph with respect to their estimated error, as sketched out in Figure 6. We present a visualization of our 3D point cloud map and our obstacle classification results in Figure 7.

2) Search and Exploration: For planetary exploration, we consider previously unknown environments as well as scenarios where a low resolution map, e.g. from a satellite image, is available. As in real space missions, the rover should be able to execute predefined tasks fully autonomously, but also be able to support semi-autonomous operation as low-level remote control is either impossible or very inefficient due to the aforementioned communication delays. Starting with a rough map from a low resolution image, a waypoint mission can be planned, either in order to fully explore the whole

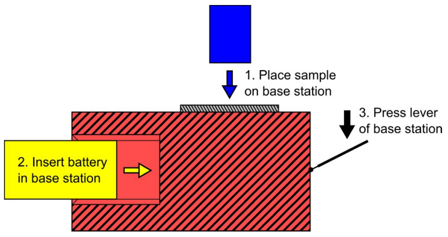
1. Place sample on base station
2. Insert battery in base station
3. Press lever of base station

Fig. 8. Assembly sketch of the base station of the SpaceBotCamp

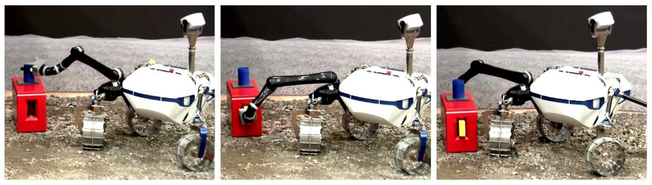

Fig. 9. Our successful base station assembly at the SpaceBotCamp

territory or to explicitly focus on areas of interest where relevant targets are likely to be found. In scenarios where no prior information is available, we employ a frontier based exploration algorithm in order to maximize coverage. It plans local goals online at the frontiers to unexplored territory with respect to the current global pose and map estimates.

In the SpaceBotCamp challenge, a rough map ($1\,\text{px} \mathrel{\widehat{=}} 0.5\,\text{m}$) was given, however we did not know the location of objects a priori. Thus, in order to find all targets, we needed to plan waypoints such that they cover the complete field. We therefore divided the area into grid cells sized according to the LRU's range of reliable perception. During the search and exploration in the challenge, we scanned the area on each waypoint with our pan/tilt sensor head, covering a full 360° angle of view around the rover. Unreachable waypoints are approached as close as possible and then skipped.

Detecting both near and far objects with a fixed lens stereo- and color setup poses considerable challenges to the algorithms. The further the objects are away, the less they reveal their characteristics due to the decreasing amount of pixels on the camera sensor. Thus segmentation by color and estimation of their extent looks more promising than trying to identify shape properties. We achieve a robust segmentation of unicolored objects in various light conditions through a learning-based classification approach, thereby taking HSV-color and brightness features as well as neighborhood relations into account. We combine the resulting segments with depth data for a cluster analysis. Since depth and angle w.r.t. to the camera are sufficient to estimate an object's position and size, we filter all clusters according to the object's expected dimensions in order to obtain a reliable and robust estimation of its location. We experimentally determined a reasonable maximum distance of $5\,\text{m}$ for object detection, taking into account robustness and the cost of acting on false positives. If an object is detected, the rover interrupts its current exploration state to pick up the object and continues with the next waypoint afterwards. We apply a fast, graph-based 2D path planner for the LRU to navigate autonomously between waypoints, taking the surrounding obstacles and local terrain classification maps into account.

C. Manipulation

To cope with the communication delay during remote planetary missions, the manipulation of probes and technical equipment must be semi-autonomous to the least. Round trip times from a few seconds (Moon) up to several minutes (Mars) degrade teleoperation techniques. The manipulation software of the LRU can autonomously pick and assemble prior known objects. An example is the assembly of the base station of the SpaceBotCamp, see Figures 8 and 9. The hardest subtask is the autonomous insertion of the battery. The rover first has to estimate the relative pose of the base station, then position the battery in front of the slot and finally insert it while actively controlling the contact.

1) Object Pose Estimation: First, the LRU has to estimate the relative pose of the base station with a precision below $1\,\text{cm}$. It therefore captures the target object with its rear stereo cameras and estimates the pose of the known object by segmenting the depth image based on the color of the object. The point cloud is then downsampled, smoothed, and surface normals are estimated using PCL. Finally, an extension of PCL's sample consensus module is used to fit the best matching cylinder or quad to the points, as described in [22]. The method is not only optimizing the inlier count probabilistically, as traditional RANSAC, but also minimizes the percentage of the shape's volume that is contradicted by depth measurements. Since we knew the sizes of the objects, those were added as constraints in the scene description language, and the remaining parameters were estimated.

A particular problem arose for the base station, which is rarely fully visible in a depth image. Thus the best matching model (quad) is likely to be too small. We therefore extended our method with the option to define a 3D model for the shapes that is then placed in all possible positions overlapping the fitted model (excluding those that violate any constraints in the scene description, e.g. models being upside down). The 3D model placements are then rendered and the resulting images compared with the camera image for an edge-distance based scoring.

2) Prepositioning the Manipulator: Second, the LRU has to preposition its hand in front of the slot with a maximum deviation of $1\,\text{cm}$. As the objects are known beforehand, we designed predefined configurations and frames for fast, robust and deterministic execution. The LRU first steers to a relative pose w.r.t. the base station. The manipulator then picks the battery from the transport basket and moves to an approach configuration. From there, the LRU computes a relative Cartesian path and moves the battery into the approach pose in front of the slot.

3) Impedance Controlled Contacts: Third, the manipulator has to control the contact to the base station. To control the stiffness of the manipulator, we replaced the commercial Kinova JACO2 API with our own controllers. We developed a communication layer based on an embedded microcontroller, which sends PWM commands directly to the motor controllers at the necessary control frequencies (approx. 700 Hz). Based on this interface, we implemented multiple control modes, including impedance control [23]. By controlling the stiffness of the manipulator, the LRU can robustly insert the battery into the slot of the base station.

D. *Task Control*

For programming complex autonomous tasks, we employ our own powerful visual programming tool *RAFCON* that is based on state machines. RAFCON can be split up in two parts: a core and a GUI. The GUI is used to visually program state machines that are then executed by the core on the target device. Our state machine concept is basically a finite state machine, combined with many concepts introduced by SyncCharts [24]. Thus, our state machine approach supports hierarchies and concurrencies. Logic connections defining the execution order are clearly separated from data connections managing the flow of data. Error and preemption handling concepts are tightly integrated. We plan a separate publication on RAFCON itself, as a more detailed description would lie beyond the scope of this paper.

For the SpaceBotCamp challenge, we built state machines with more than 700 states, 1200 transitions and up to 8 hierarchy levels with RAFCON, demonstrating the capability and scalability of the tool. The handling of huge state machines is possible due to the state machine editor GUI featuring an elaborate zooming concept. This allows the user to zoom and pan inside the state machine like in a digital map and thereby facilitates dynamic expansion or hiding of details of sub-states deeply nested in a hierarchy. Moreover it supports state re-usability as state machines, designed by different developers, can be linked into more complex state machines by so called library states. In scenarios like the SpacebotCamp challenge, state machine development is typically distributed among several developers. In our case, one cared for the states concerning computer vision, another prepared the navigation states and a third wrapped the manipulation functionalities of the robot into states. Thereby, we can provide all of the robot's functionalities in small state machines that we then combine to more complex ones. Furthermore, these single building blocks can be re-used in other scenarios, reducing development time. In Figure 10, we present a simplified hierarchical state machine for fetching an object. The ability to launch state machines at arbitrary states

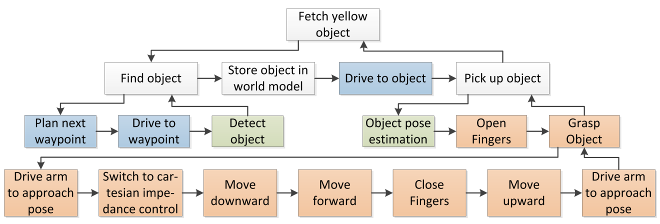

Fig. 10. A hierarchical state machine that combines different functionalities of the robot to the more abstract state "Fetch yellow object". Each of the state rows represent a different hierarchy level, the arrows illustrate the transitions. Branching transitions, e.g. for error recovery, have been omitted for clarity. States concerning navigation are marked blue, states dealing with object localization green and states concerning manipulation red.

boosts quick state machine development and makes powerful error handling and failure recovery possible. Especially in complex missions, in which a part of the task has already been fulfilled, dedicated entry points allow a full system recovery without performing redundant work.

E. *Ground Station Mission Control*

Being able to monitor and - if necessary - intervene with a robot's task execution is crucial for being able to operate a robot in unstructured and unknown environments. In contrast to many field robotics applications here on earth, the communication channel to a robot deployed on another planet will likely be low-bandwidth, severely delayed and unreliable with potentially extended periods of communication blackouts in one or both directions. Our approach for system and mission control is designed to take these constraints into account and allows to efficiently monitor autonomous task execution as well as to provide means for shared autonomy robot control when necessary.

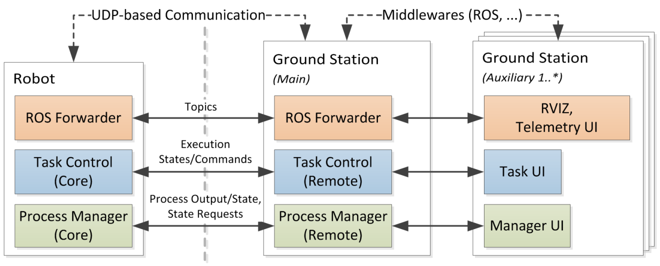

Fig. 11. Communication setup for remote monitoring and access

1) *Dealing with Unreliable Communication Channels:*

To avoid any side-effects of an unreliable communication channel on the robot's task execution, all communication between the robot and a human operator at the ground station (mission control) is based on non-acknowledged communication over UDP between otherwise completely decoupled networks. Not using any acknowledgment mechanism for communication fits well to many transmitted data streams of continuous nature (e.g. sensor data) and allows the robot to operate in scenarios where no uplink to the robot is available (the default in the SpaceBotCamp scenario).

For transmitting task-relevant information like sensor data, planned paths or detected objects, we use ROS in a multi-master setup with two ROS networks, one on the robot and one between all ground station computers. We developed an UDP forwarder to exchange any ROS topics in a network-transparent way between subscribers and publishers on both sides. High-bandwidth data like images and point clouds are transferred rate-limited and with reduced resolution. For specific ROS message types like images, the data is split and marshalled in a way that allows to reassemble a complete ROS message even if large parts of the message's packets were not received. We for example transmit image data chunk-wise and fill in missing chunks on the receiver side.

We monitor the execution status of ROS and non-ROS processes running on the robot through our process manager provided by our *Links and Nodes* middleware, thereby gaining access to all processes' console output. In addition, we get a clear picture if all required components operate normally by specifying run and restart dependencies as well as conditions for how a process normally behaves during start-up, execution and quitting (defined by regular expressions on console output). The console outputs and execution state (stopped, starting, started, ready, stopping, error, etc.) per process are forwarded to the ground station via UDP multicasts, allowing multiple operators and applications to receive and analyze this information at the same time. The process manager on the robot side also implements a UDP command protocol over which start/stop requests for any process can be triggered from the ground station. To reliably monitor the task execution and high-level system state, our task control software implements a UDP-based communication protocol (separate from our mechanism for ROS messages) with additional message bursts, sequence numbers and hash values to cope with packet loss and reordering during transmission of state information. Similar to the process manager, a UDP command protocol allows for example to pause and continue task execution.

2) Remote Access for Shared Autonomy: In case of (imminent) failure or observed abnormal behavior on system- or component-level, we use several mechanisms to restore normal operation conditions and resume autonomous task execution. On task-level, we are able to start, stop, pause and step through the active task execution model. In addition to the main task, we maintain a repository of standalone (sub)tasks, which can be triggered remotely and largely require no specific start conditions (e.g. *lookBackwards*, *moveArmToHomePosition*, *detectObjectA*). Usually they are part of the main tasks' execution flow and generate outputs (e.g. ROS tf frames), which can be used once the main task is resumed. On process-level, we are able to stop and restart processes and process groups on the robot. In addition, we prepared failure handling scripts, which can be triggered for example to re-initialize components or modify parameters.

3) Mission Control Setup at Groundstation: Our mission control setup at the ground station, as used during the SpaceBotCamp challenge, includes three operators - *navigation*, (general) *system* and *manipulation operator* - and one supervising *operator coordinator*. The navigation operator monitors planning and execution of drive motions as well as robot localization and mapping through appropriate RVIZ visualizations for navigation frames, paths and the generated maps (obstacle map, height map, etc.). The system operator focuses on warning/error monitoring for all processes, (sub)system restarts, monitoring and, if necessary, control of high-level task execution. For this purpose, we use the ROS console together with custom remote UIs for process management and high-level task visualization/execution. The manipulation operator watches over correct manipulator motion planning & execution and object recognition. All operators have access to all basic robot telemetry information such as battery voltage, CPU usage and temperature, emergency status, joint position & torques, controller state and control rate statistics via a custom UI as well.

V. The SpaceBotCamp Challenge

The SpaceBotCamp[1] is a national robotics challenge organized by the DLR Space Administration. Similar to the DARPA robotics challenges in the US, its goal is to stimulate innovations, benchmark state-of-the-art technologies in the field of autonomous mobile robotics and kick-start their integration into working systems. It took place in November 2015 in Hürth, Germany with ten participating teams from universities and research institutes from all over Germany. We all had to face a challenging task focused on autonomous exploration and manipulation in an unstructured, previously unknown GPS-denied environment.

A. Scenario Description

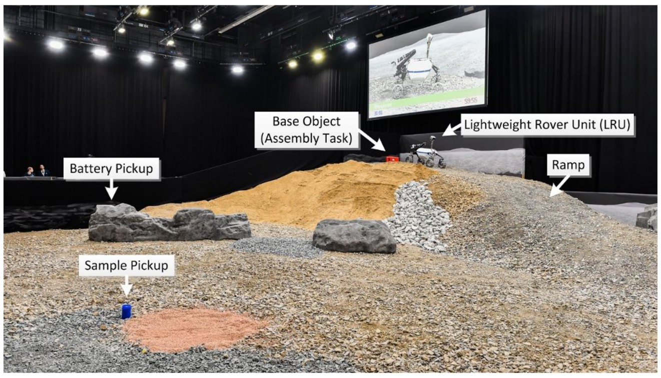

Fig. 12. An overview of the SpacebotCamp scenario. The competition field was 13 m x 18 m. The LRU started next to the red base station on top of a hill of approx. 2 m height. In the foreground, a blue container filled with a rock sample can be seen, while the yellow battery object is hidden by a large rock. The goal of the challenge was to explore and map the unknown terrain as well as to locate and collect the sample and the battery object for the assembly task at the base station.

An autonomous robot system, consisting of one or multiple robots with a total mass of less than 100 kg, had to autonomously explore and map an area modeled after a moon-like, rough-terrain planetary surface. Therein, it had to locate and collect both a blue container with a rock sample (approx. 500 g) as well as a yellow object representing a battery (approx. 800 g). Both objects had to be transported to a red base station and finally assembled, as sketched out in Figure 8. A high level of autonomy was necessary to fulfill the task since a delay of four seconds round trip time was artificially added to the communication in order to simulate the real delay between earth and moon. Furthermore, the uplink to the robotic system was completely blocked, except for up to three five-minute checkpoints. During these limited time frames, a ground station crew was allowed to send commands to the robotic system over a delayed channel. Apart from that, they could only passively monitor the robot. In addition, they were located in a separate room and thus without any visual contact to the competition field. After two days of preparation, each team had a single opportunity to solve the challenge within a sixty-minute time slot in front of a public audience.

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

B. Results and Discussion

According to the original rules of the challenge, we were the only team amongst the ten competitors to fulfill all mandatory tasks. Furthermore, we solved the tasks while facing all of the specified communication constraints from the very beginning of the mission. We accomplished this in just thirty minutes, half of the given time frame, and with full onboard autonomy. In contrast, many other teams softened the challenging communication restrictions in order to allow for mixed autonomy and teleoperation approaches. We only took a single one of the three allowed checkpoints to double-check the object localization for the base station as its precision is crucial for flawless assembly. We thereby solely sent four high-level commands to the rover during the whole mission, including the one-way delay of two seconds for all sent commands and received data. Thus we could demonstrate that we still had full high-level control of the system despite the delayed and constrained communication link between ground station and LRU. Finally, our system was the only robot which managed to climb and descend the steep crushed-stone ramp, shown in Figure 12, fully autonomously. We present a video of our run at https://youtu.be/wCTkSxcna8o

VI. Conclusion and Future Work

In this work, we have presented the Light Weight Rover Unit (LRU) as an agile rover system for autonomous planetary exploration. We provide an overview of our system architecture, detailing both the hardware and software components as well as our ground station setup for monitoring the robot's state and activities over a delayed and restricted communication link. We designed the LRU to operate at a high level of autonomy during rough-terrain navigation, search and exploration as well as object manipulation tasks. This is essential to conduct efficient space missions in the light of delayed communications to foreign planets. In addition, our task control framework allows interactions with the rover at a high level of abstraction in case shared autonomy is needed. The LRU faced the challenges posed by the SpaceBotCamp 2015 planetary exploration scenario with great success and allowed us to demonstrate a fully autonomous operation with very limited options for monitoring and remote access. For future work, we plan to extend the rover's object manipulation abilities by employing a more flexible grasp and assembly planning. In addition, we work on joint localization and mapping with multiple robots to improve efficiency through parallelization, robustness through redundancy and to benefit from complementary capabilities in heterogeneous robot teams.

Acknowledgment

We thank the members of the Mobile Robots Group at DLR-RMC, especially Florian Schmidt and Philipp Lutz, for their assistance. This work was supported by the Helmholtz Association, project alliance ROBEX, under contract number HA-304.

References

[1] A. Wedler, B. Rebele, J. Reill, M. Suppa, H. Hirschmüller, C. Brand, M. Schuster, B. Vodermayer, H. Gmeiner, A. Maier, B. Willberg, K. Bussmann, F. Wappler, and M. Hellerer, "LRU - Lightweight Rover Unit," in *ASTRA*, 2015.

[2] J. P. Grotzinger *et al.*, "Mars Science Laboratory Mission and Science Investigation," *Space Science Reviews*, vol. 170, no. 1, pp. 5–56, 2012.

[3] M. Maimone, A. Johnson, Y. Cheng, R. Willson, e. M. H. Matthies, Larry", and O. Khatib, *Experimental Robotics IX*. Springer Berlin Heidelberg, 2006, ch. Autonomous Navigation Results from the Mars Exploration Rover (MER) Mission, pp. 3–13.

[4] R. Washington, K. Golden, J. Bresina, D. Smith, C. Anderson, and T. Smith, "Autonomous Rovers for Mars Exploration," in *IEEE Aerospace Conference*, vol. 1, 1999.

[5] D. Wettergreen, M. Wagner, D. Jonak, V. Baskaran, M. Deans, S. Heys, D. Pane, T. Smith, J. Teza, D. R. Thompson, *et al.*, "Long-Distance Autonomous Survey and Mapping in the Robotic Investigation of Life in the Atacama Desert," in *iSAIRAS*, 2008.

[6] J. Stückler, M. Schwarz, and M. Schadler, "NimbRo Explorer: Semi-Autonomous Exploration and Mobile Manipulation in Rough Terrain," *JFR*, 2015.

[7] M. Eich, R. Hartanto, S. Kasperski, S. Natarajan, and J. Wollenberg, "Towards Coordinated Multirobot Missions for Lunar Sample Collection in an Unknown Environment," *JFR*, vol. 31, no. 1, 2014.

[8] J. Schwendner, T. Röhr, S. Haase, M. Wirkus, M. Manz, S. Arnold, and J. Machowinski, "The Artemis Rover as an Example for Model Based Engineering in Space Robotics," in *ICRA Workshop*, 2014.

[9] A. Wedler, M. Chalon, K. Landzettel, M. Görner, E. Krämer, R. Gruber, A. Beyer, H.-J. Sedlmayr, B. Willberg, B. Wieland, J. Reill, M. Schedl, A. Albu-Schäffer, and G. Hirzinger, "DLR's Dynamic Actuator Modules for Robotic Space Applications," in *Aerospace Mechanisms Symposium*, no. 41, 2012.

[10] R. Haarmann, Q. Mühlbauer, L. Richter, S. Klinkner, C. Lee, C. Wagner, R. Jaumann, A. Koncz, H. Michaelis, J. Schwendner, H. Hirschmüller, and A. Wedler, "Mobile Payload Element (MPE): Concept study for a sample fetching rover for the ESA Lunar Lander Mission," in *I-SAIRAS*, 2012.

[11] A. Wedler, A. Maier, J. Reill, C. Brand, H. Hirschmüller, M. Suppa, A. Beyer, and R. Haarmann, "Pan/Tilt-Unit as a Perception Module for Extra-Terrestrial Vehicle and Landing Systems," in *ASTRA*, 2013.

[12] J. Reill, H. Sedlmayr, P. Neugebauer, M. Maier, E. Krämer, and R. Lichtenheldt, "MASCOT – Asteroid Lander with Innovative Mobility Mechanism," in *ASTRA*, 2015.

[13] H. Hirschmüller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IPAMI*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[14] K. H. Strobl and G. Hirzinger, "Optimal Hand-Eye Calibration," in *IROS*, 2006.

[15] H. Hirschmüller, P. Innocent, and J. Garibaldi, "Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics," *ICARCV*, vol. 2, pp. 1099–1104, 2002.

[16] K. Schmid, F. Ruess, and D. Burschka, "Local Reference Filter for Life-Long Vision Aided Inertial Navigation," in *FUSION*, 2014.

[17] K. Schmid, F. Ruess, M. Suppa, and D. Burschka, "State Estimation for highly dynamic flying Systems using Key Frame Odometry with varying Time Delays," in *IROS*, 2012.

[18] C. Brand, M. J. Schuster, H. Hirschmüller, and M. Suppa, "Stereo-Vision Based Obstacle Mapping for Indoor/Outdoor SLAM," in *IROS*, 2014.

[19] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2 : Incremental Smoothing and Mapping Using the Bayes Tree," *IJRR*, vol. 31, pp. 217–236, 2012.

[20] M. J. Schuster, C. Brand, H. Hirschmüller, and M. Suppa, "Multi-Robot 6D Graph SLAM Connecting Decoupled Local Reference Filters," in *IROS*, 2015.

[21] C. Brand, M. J. Schuster, H. Hirschmüller, and M. Suppa, "Submap Matching for Stereo-Vision Based Indoor/Outdoor SLAM," in *IROS*, 2015.

[22] S. Büttner, Z.-C. Márton, and K. Hertkorn, "Automatic scene parsing for generic object descriptions using shape primitives," *Robotics and Autonomous Systems*, vol. 76, pp. 93–112, 2016.

[23] A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *IJRR*, vol. 26, no. 1, pp. 23–39, 2007.

[24] C. André, "SyncCharts: A Visual Representation of Reactive Behaviors," *TR 95-52, Université de Nice-Sophia Antipolis*, 1995.