Joint 45th International Symposium on Robotics (ISR) and 8th German Conference on Robotics (ROBOTIK), Munich, June 2014.

Local Navigation in Rough Terrain using Omnidirectional Height

Max Schwarz, max.schwarz@uni-bonn.de
Sven Behnke, behnke@cs.uni-bonn.de

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

Abstract

Terrain perception is essential for navigation planning in rough terrain. In this paper, we propose to generate robot-centered 2D drivability maps from eight RGB-D sensors measuring the 3D geometry of the terrain 360° around the robot. From a 2.5D egocentric height map, we assess drivability based on local height differences on multiple scales. The maps are then used for local navigation planning and precise trajectory rollouts. We evaluated our approach during the DLR SpaceBot Cup competition, where our robot successfully navigated through a challenging arena, and in systematic lab experiments.

1 Introduction

Most mobile robots operate on flat surfaces, where obstacles can be perceived with horizontal laser-range finders. As soon as robots are required to operate in rough terrain, locomotion and navigation becomes rather difficult. In addition to absolute obstacles, which the robot must avoid at all costs, the ground is uneven and contains obstacles with gradual cost, which may be hard or risky to overcome, but can be traversed if necessary.

To find traversable and cost-efficient paths, the perception of the terrain between the robot and its navigation goal is essential. As the direct path might be blocked, an omnidirectional terrain perception is desirable. To this end, we equipped our robot, shown in Fig. 1, with eight RGB-D cameras for measuring 3D geometry and color in all directions around the robot simultaneously. The high data rate of the cameras constitutes a computational challenge, though.

In this paper, we propose efficient methods for assessing drivability based on the measured 3D terrain geometry. We aggregate omnidirectional depth measurements to robot-centric 2.5D omnidirectional height maps and compute navigation costs based on height differences on multiple scales. The resulting 2D local drivability map is used to plan cost-optimal paths to waypoints, which are provided by an allocentric terrain mapping and path planning method that relies on the measurements of the 3D laser scanner of our robot [10].

We evaluated the proposed local navigation approach in the DLR SpaceBot Cup—a robot competition hosted by the German Aerospace Center (DLR). We also conducted systematic experiments in our lab, in order to illustrate the properties of our approach.

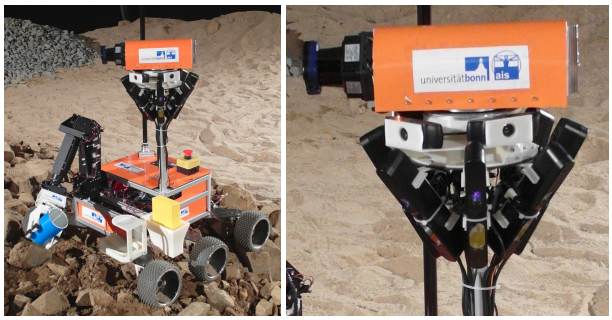


Figure 1: Explorer robot for mobile manipulation in rough terrain. The sensor head consists of a 3D laser scanner, eight RGB-D cameras, and three HD cameras.

2 Related Work

Judging traversability of terrain and avoiding obstacles with robots—especially planetary rovers—has been investigated before. Chhaniyara et al. [1] provide a detailed survey of different sensor types and soil characterization methods. Most commonly employed are LIDAR sensors, e.g. [2, 3], 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.
Further LIDAR-based approaches include Kelly et al. [6], who use a combination of LIDARs on the ground vehicle and an airborne LIDAR on a companion aerial vehicle. The acquired 3D point cloud data is aggregated into a 2.5D height map for navigation, and obstacle cell costs are computed proportional to the estimated local slope, similar to our approach. The additional viewpoint from the arial vehicle was found to be a great advan-

tage, especially in detecting negative obstacles (holes in the ground), which cannot be distinguished from steep slopes from afar. This allows the vehicle to plan through areas it would otherwise avoid because of missing measurements, but comes at the price of an increased system complexity.

Structured light has also been proposed as a terrain sensing method by Lu et al. [8], however, they use a single laser line and observe its distortion to measure the terrain geometry. Our sensor has comparable angular resolution (0.1° in horizontal direction), but can also measure 640 elevation angles (0.1° resolution) at the same time.

Kuthirummal et al. [7] present an interesting map representation of the environment based on a grid structure parallel to the horizontal plane (similar to our approach), but with height histograms for each cell in the grid. This allows the method to deal with overhanging structures (e.g. trees, caves, etc.) without any special consideration. After removing the detected overhanging structures, the maximum height observed in each cell is used for obstacle detection, which is the same strategy we use.

The mentioned terrain classification works [1, 8] all include more sophisticated terrain models not only based on terrain slope, but also on texture, slippage, and other features. The novelty of our work does not lie in traversability analysis, but in the type of sensor used. 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. The used cameras are consumer products (ASUS Xtion Pro Live) which are available at a fraction of the cost of 3D LIDAR systems. Stereo vision is a comparable concept, but has the disadvantage of being dependent on ground texture and lighting. Our sensor also produces RGB-D point clouds, which can also be used for other purposes like further terrain classification based on appearance, visual odometry, or even RGB-D SLAM.

A local navigation method alone does not enable the robot to truly navigate the terrain autonomously. A higher level, allocentric map and navigation skills are required. Schadler et al. [10] developed an allocentric mapping, real-time localization and path planning method based on measurements from a rotating 2D laser scanner. The method employs multi-resolution surfel representations of the environment which allow efficient registration of local maps and real-time 6D pose tracking with a particle filter observing individual laser scan lines. The rotating 2D laser scanner takes around 15 s to create a full omnidirectional map of the environment. While the range of our RGB-D sensors is not comparable to a laser scanner, the update rate is much faster, allowing fast reaction to vehicle motion and environment changes.

3 Omnidirectional Depth Sensor

Our 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. 3a).

3.1 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. 3b). 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.

3.2 Data Acquisition

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 bus. To overcome this limitation, we equipped the onboard PC with two PCI Express USB cards, which provide four USB 3.0 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 [9] 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].

4 Local Drivability Maps

A general overview of our data processing pipeline is given in Fig. 2. The input RGB-D pointclouds are projected on a gravity-aligned plane, locally differentiated and then used as cost maps for existing navigation components.

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

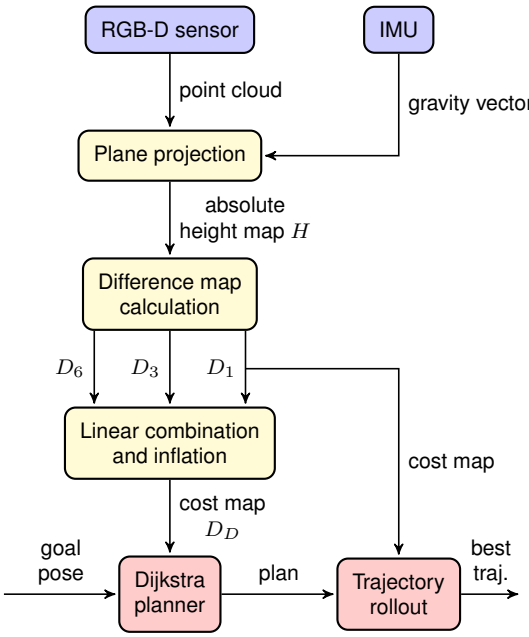


Figure 2: Data flow of our method. Sensors are colored blue, data filtering/processing modules yellow and navigation components in red.

4.1 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.

Because depth measurements contain noise depending on the type of ground observed, it is necessary to include a filtering step on the depth images. In this filtering step, outliers are detected with a median filter and subsequently eliminated. In our case, we use a window size of 10×10 pixels, which has proven to remove most noise in the resulting maps. The median filter is based on the optimization of Huang et al. [5] which uses a local histogram for median calculation in $O(k)$ per pixel (for a window size of $k\times k$ pixels). The filtering is executed in parallel for all eight input point clouds.

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 an IMU in the robot base.

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 resulting height map is two orders of magnitude smaller than the original point clouds of the eight cameras.

4.2 Filling Gaps

Gaps in the height map (i.e. regions with NaN values) can occur due to several reasons. Obviously, occluded regions cannot be measured. Also, sensor resolution and range are limited and can result in empty map cells at high distance. Finally, the sensor might not be able to sense the material (e.g. transparent objects). Our general policy is to treat areas of unknown height as obstacles. However, in many cases, the gaps are small and can be filled in without risk. Larger gaps are problematic because they may contain dangerous negative obstacles.

To fill in small gaps, we calculate the closest non-NaN neighbors of each NaN cell in each of the four coordinate directions. NaN cells which have at least two non-NaN neighbors closer than a distance threshold δ away will be filled in. As the observation resolution linearly decreases with the distance from the robot, δ was chosen to be

$$\delta = \delta_0 + \Delta \cdot r$$

where r is the distance of the map cell to the robot. With our sensors, $\delta_0 = 4$ and $\Delta = 0.054$ (all distances in map cells) were found to perform well.

The value of the filled cell is calculated from an average of the available neighbors, weighted with the inverse of the neighbor distance.

For example, the occlusion of the large robot antenna visible in Fig. 3(a,b) is reliably filled in with this method.

4.3 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:

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

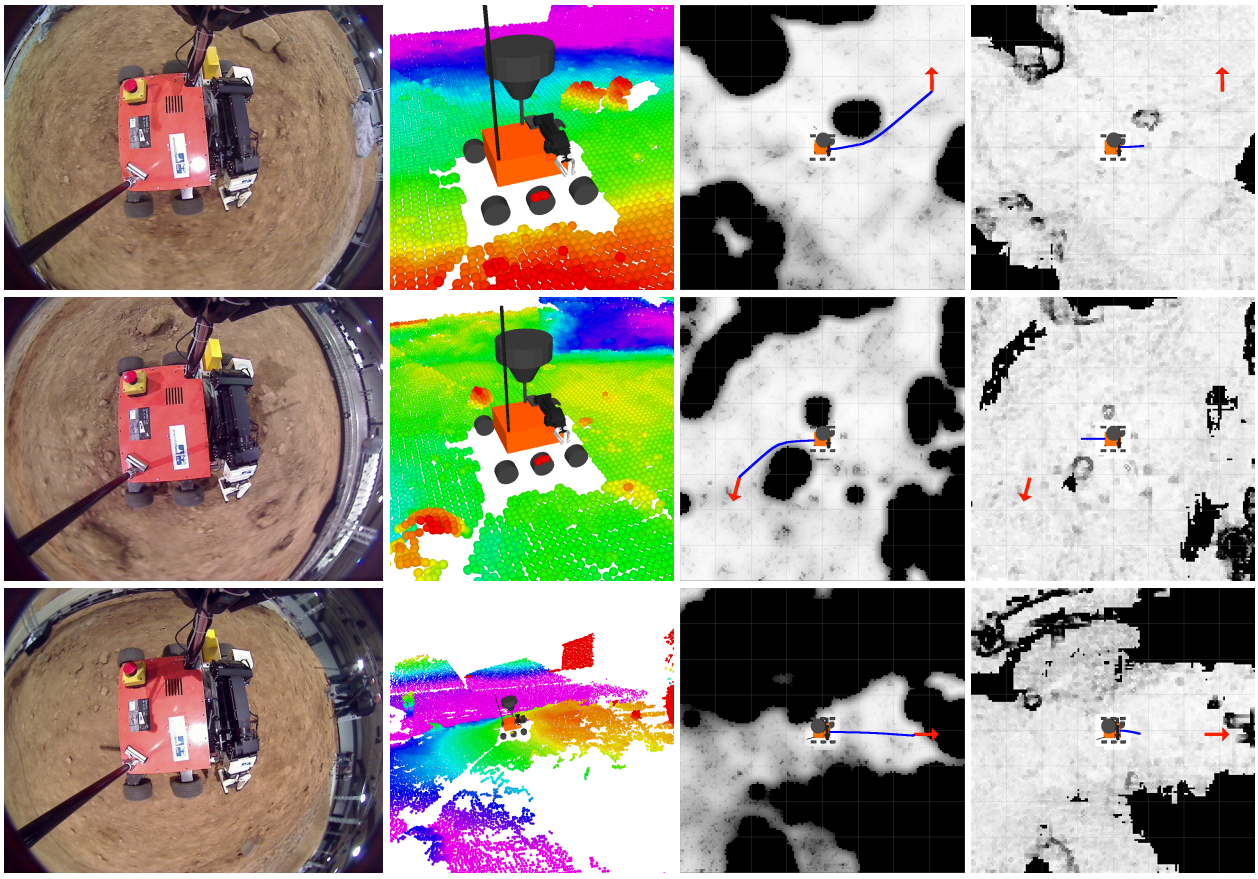
$H(u, v)$ values of NaN are ignored. If the center pixel $H(x, y)$ itself is not defined, or there are no other defined l-neighbors, we assign $D_l(x, y) :=$ NaN.

Furthermore, we do not allow taking differences across camera boundaries. This prevents small transformation errors between the cameras from showing up in the drivability maps as obstacles. In the relevant ranges close to the robot, this additional constraint does not reduce information, because the field of views 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.

5 Navigation Planning

For path planning, the existing ROS solutions were employed. Our method creates custom obstacle maps for



(a) (b) (c) (d)

Figure 3: 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 people standing at the top of a large ramp.

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.

5.1 Trajectory Planning

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:

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

The coefficients λ_l were tuned manually and can be seen in Table 1.

Table 1: Coefficients for combining height differences.

l	1	3	6
λ_l	5.0	2.5	3.5

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:

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

$$D_D(x,y) := \max\left\{\tilde{D}(x,y), \sum_{(u,v)\in P(x,y)} \frac{\tilde{D}(u,v)}{|P(x,y)|}\right\}.$$

5.2 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.

5.3 Visual Odometry

Our local navigation method does not rely directly on accurate odometry information, as no state is kept across states and the input goal pose is updated with a high frequency by our global path planner [10]. We use odometry, however, when the operator is manually giving goal poses to the local navigation. This can be necessary when higher levels of autonomy are not available and the operator has to take a more direct role in controlling the robot. Odometry is used then to shift the given local goal pose as the robot moves.
The odometry measured from wheel movements can be very imprecise, especially in planetary missions with sand-like ground. To improve this, we first estimate movement for each camera with the well-known *Fovis* algorithm [4] from gray-scale and depth images. Individual camera movement, wheel odometry and IMU data are then combined into a fused odometry estimate.

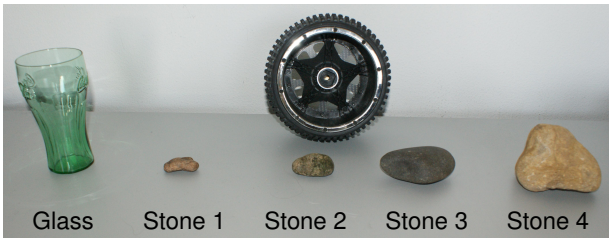
6 Evaluation

We evaluated our method during the DLR SpaceBot Cup competition, which focused on exploration and mobile manipulation in rough terrain. Communication between operators and the robot was impeded by an artificial delay, simulating an actual earth-moon connection. In order to complete the tasks in acceptable time, autonomous functions, such as local navigation and grasping, were required. Given local navigation commands by the operator, the system was able to navigate in all traversable parts of the arena. After the competition, we also recorded a full RGB-D dataset while steering the robot manually through the arena. Figure 3 shows calculated drivability maps and planned paths in three exemplary situations retrieved from the recording. It can be seen that the relevant obstacles are reliably perceived and obstacle-free paths are planned around them. Color information alone would clearly not by sufficient for navigation in the SpaceBot Cup environment—even the human operator cannot perceive all dangerous obstacles and slopes in the overhead camera image.
We also evaluated our proposed approach in our lab. Fig. 4 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.
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. 5a at varying distances to the robot on flat ground (carpet). In each situation, 120 frames of the local obstacle map D_1 were recorded. 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. 5b, the system can reliably detect obstacles which pose danger to the robot up to 3.5 m away.

Somewhat surprisingly, the system also detects the glass reliably for higher distances, even though the RGB-D sensor itself has problems with transparent objects. The reason for that is that even though the sensor cannot measure distance to the object, this creates a sizable gap through occlusion in the height map, which of course is not traversable and thus detected as an obstacle.

The processing pipeline from eight 240×320 RGB-D point clouds to the two drivability maps runs in around 50 ms on an Intel Core i7-4770K processor with 3.5 GHz. The method is open to further optimization, especially in the noise filtering step, which can take up to 10 ms.

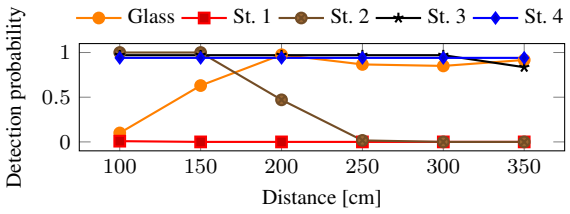
We also evaluated our custom camera driver. The standard ROS driver is not stable enough with eight cameras for direct comparison, so we can only provide absolute metrics for our driver. Capturing VGA color and depth images and converting them to RGB-D point clouds at 30 Hz is possible and creates a 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 %.



Object dimensions.

Object	Glass	Stone 1	Stone 2	Stone 3	Stone 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 size 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) Estimated detection probabilities at increasing distance.

Figure 5: Obstacle detection experiment.

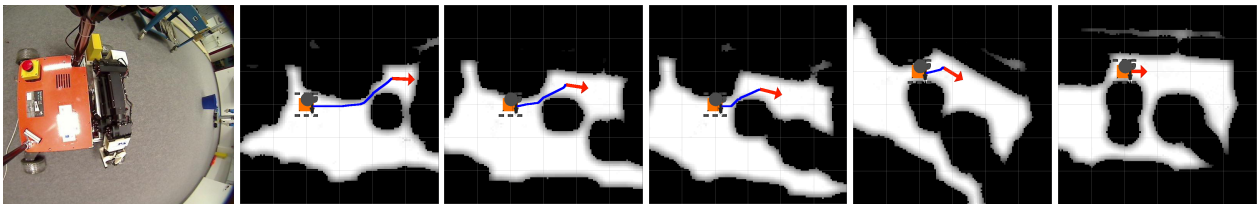


Figure 4: Lab driving experiment. The first image shows the initial situation observed from the robot overhead camera. Subsequent drivability maps show the planned path (blue line) towards the goal (red arrow).

7 Conclusion

We have shown that the proposed method allows the robot to navigate in previously unknown terrain using only locally observed depth data. The method was proven to be successful and reliable in the SpaceBot Cup competition.

This work is the first approach to omnidirectional terrain sensing with depth cameras to our knowledge. As explained above, this approach provides multiple advantages over other sensor types. The inherent challenge of our approach was the processing of depth data in acceptable time, which was only possible with the chosen 2D grid aggregation and the resulting applicability of efficient image processing methods instead of expensive true 3D processing. The developed local navigation method is an important step towards fully autonomous operation in complex environments, which is our long-term goal.

The dependence on the existing ROS navigation planners has proven to be a hindrance, though. These planners were designed for navigation in 2D laser obstacle maps and have to be modified to accept gradual costs. In the future, a custom planner with inherent knowledge about the multiple D_l maps might be able to achieve even better results and performance.

A more efficient kernel-space driver for the ASUS Xtion cameras is also under development. We expect a substantial increase in performance by avoiding unnecessary copies, buffering and vectorization of decompression and conversion routines.

The sensor itself opens up new possibilities in terrain sensing and navigation. The direct availability of color information makes the sensor applicable to existing terrain classification methods.

References

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

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

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

[4] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

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

[6] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006.

[7] S. Kuthirummal, A. Das, and S. Samarasekera. A graph traversal based algorithm for obstacle detection using lidar or stereo. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3874–3880. IEEE, 2011.

[8] L. Lu, C. Ordonez, E. Collins, E. Coyle, and D. Palejiya. Terrain surface classification with a control mode update rule using a 2d laser stripe-based structured light sensor. *Robotics and Autonomous Systems*, 59(11):954–965, 2011.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 2009.

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