Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-15 (2016) 212–217

3D Navigation Mesh Generation for Path Planning in Uneven Terrain

Sebastian Pütz * **Thomas Wiemann** ** **Jochen Sprickerhof** *** **Joachim Hertzberg** ****

* *Osnabrück University, Germany (spuetz@uni-osnabrueck.de)*
** *Osnabrück University, Germany (twiemann@uni-osnabrueck.de)*
*** *Osnabrück University, Germany (jspricke@uni-osnabrueck.de)*
**** *Osnabrück University and DFKI Robotics Innovation Center, Osnabrück Branch, Germany (joachim.hertzberg@uni-osnabrueck.de)*

Abstract: We present a 3D mesh surface navigation system for mobile robots. This system uses a 3D point cloud to reconstruct a triangle mesh of the environment in real time that is enriched with a graph structure to represent local connectivity. This *Navigation Mesh* is then analyzed for roughness and trafficability and used for online path planning. The presented approach is evaluated with a VolksBot XT platform in a real life outdoor environment.

© 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Robot Navigation, Path Planning, Surface Reconstruction, Mesh Generation

1. INTRODUCTION

In flat terrain most obstacles can be safely perceived by using horizontal 2D laser scanners. Compared to that, it is much more difficult to operate in rough terrain and to check the trafficability of uneven ground. Robots operating outdoors like in rescue operations or exploration of unknown environments need to cope with such uneven terrain and must still be able to find a path to their goal. Therefore, they have to classify if something is a potential obstacle or if they can pass it, e.g., by traversing over it. To reach the target, the trafficability of the ground between the current position and the goal pose is essential. Rough terrain involves certain risks, such as holes in the ground, bold peaks or sharp objects. There might be heavy obstacles that pose danger to the robot and that must be avoided in any case. But there might also be obstacles the robot can deal with at higher costs (energy, execution time) in order to reach the target.

We have developed a general mesh planner extending the ideas of Breitenmoser and Siegwart (2012) and Gingras et al. (2010). This paper presents our approach for local planning and robot control in the current field of view and demonstrates the usability on a mobile robot operating in a real world outdoor environment. Unlike 2D grid map approaches, this 3D planner works on triangle meshes instead of 2D grid map planes. Working with such 3D surfaces in navigation has the advantage of conformity of the navigation costs based on distances, height differences, roughness, forbidden areas and other criteria. Additionally, meshes represent 3D space directly and can easily be annotated with semantic labels as described in Deeken et al. (2014). Compared to 2.5D solutions, full 3D mesh surface navigation algorithms enable navigating through a multi level environment, such as multistory buildings or tunnel systems. In particular mesh navigation is appropriate for climbing robots or other robots with the ability to access walls or ceilings.

In this paper we present an open source, robot-independent 3D mesh navigation solution for robot navigation with high extensibility.

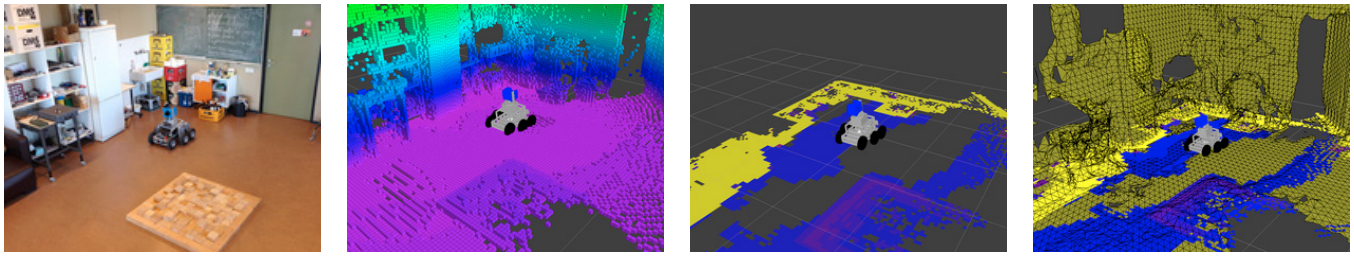
2. RELATED WORK

2.1 Octree Mapping and Digital Elevation Maps

Many navigation and path planning approaches use Octrees or similar representations that are created from 3D point cloud data. A popular implementation for it is the OctoMap package in ROS (Hornung et al., 2013). A planning system for Octrees is described in Morisset et al. (2009). It decomposes the 3D Octree map into several regions that are locally 2D. These regions can than be used like a ordinary grid map for path planning. Also, semantic classification of the represented surfaces can be integrated to speed up the computation. Fig. 1b shows an example of the Octree representation of a natural scene.

Occupancy maps, like Octrees, represent obstacles in a binary way in form of a grid map. These maps only define obstacles, but not the trafficability of obstacles in a specific context. For example, an obstacle can provide drivable ground, if the robot is on top of it, but also an inaccessible ledge, if the robot is located at a lower level. Generally, obstacle detection depends on the obstacle's size and the distance to the obstacle, as evaluated in Schwarz and Behnke (2014). To deal with these shortcomings, elevation maps build on top of simple occupancy maps. Starting with height maps, where a height z is stored in the cells of the grid map with $z = f(x, y)$. These solutions are usually called 2.5D maps (cf. Fig. 1c). A special type of 2.5D maps are *Digital Elevation Maps* (DEM). These maps are used to compute local trafficability maps, e.g., by calculating the navigation costs based on the height differences on multiple scales (Schwarz and Behnke, 2014). However, these techniques are not able to handle multiple overlapping levels, such as tunnel systems or multistory

2405-8963 © 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.07.734



(a) Lab example scene with a step field.

(b) Octree representation of the scene.

(c) 2.5D Map (DEM) representation of the scene.

(d) Mesh representation of the scene.

Fig. 1. State of the art map representations to estimate trafficability in 3D in uneven terrain.

buildings and are not practicable for climbing robots. For evaluation we implemented the DEM trafficability estimation schemes presented in Schwarz and Behnke (2014) in a *costmap_2d* layer plugin for *move base* in *ROS*. An example DEM costmap is shown in Fig. 1d together with the reconstructed mesh for comparison.

2.2 Terrain Reconstruction

An alternative approach is described in Gingras et al. (2010) and Breitenmoser and Siegwart (2012). This method is able to handle multiple levels by reconstructing the rough terrain into a triangle mesh. Such triangle mesh representations are much more flexible and can be used for multiple purposes like environment representations in robotic simulators (Wiemann et al., 2013), visualization for human robot interaction and ground classification or path planning (Wiemann et al., 2010).

In such a representation, the alignment of surface parts or change between adjacent triangles can be evaluated for drivability estimation. This allows to determine navigation costs based on distances, height difference, roughness, forbidden areas and semantic information.

3. SYSTEM ARCHITECTURE

The components of our approach are shown in Fig. 2. The main input is a triangle mesh constructed with some kind of surface reconstruction procedure. In principle, arbitrary meshes from different sources like LVR (Wiemann et al., 2012) or PCL (Rusu and Cousins, 2011) could also be used. In this paper, we use the *Organized Fast Mesh* method to generate a triangle mesh surface representation from 3D laser scanner data at the current position of the robot in real time (Holz and Behnke, 2012). In this paper we do not present an approach for global mapping. However, our approach also works for meshes representing larger environment sections, coequal with some kind of a global map.

First, a so called *Navigation Mesh* is generated and analyzed to estimate distances, height differences, and roughness. If specific safety thresholds are violated, these areas will be marked as lethal obstacles, since the robot must avoid them. Areas around such lethal obstacles are inflated (marked as dangerous) and graph optimization algorithms are applied, e.g., *Graph Edge Region Growing* to implicitly smooth the discretized mesh path. Now it is possible to apply shortest path algorithms on the resulting optimized *Navigation Mesh* to compute paths taking the evaluated

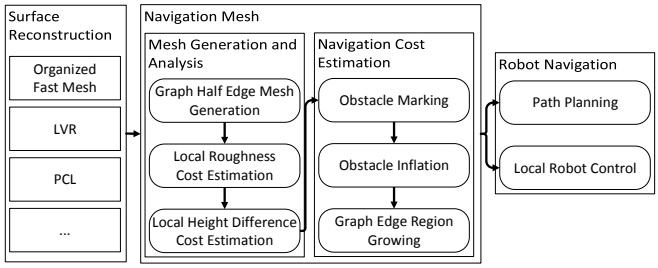


Fig. 2. Structure of our terrain classification approach. Triangle mesh representations are converted into a graph that is used for planning using navigation costs estimated for the represented terrain.

criteria into account. Besides global navigation, the *Navigation Mesh* can be used for a local control of the robot. Details on the single computation steps are presented in the following sections.

4. SURFACE RECONSTRUCTION

4.1 Organized Point Clouds

The organized fast meshes we use in this paper rely on organized point clouds. The data flow to generate them is displayed in Fig. 3. After removing outliers from a single 2D laser scan and applying a robot self filter, we assemble the single 2D laser scans to one organized point cloud. In such a representation, neighboring points are stored in a matrix structure resulting in a depth image. To generate this matrix, we exploit the fact that we know the polar angles ϕ and θ for every single point. Our rotation unit delivers for every single point the ϕ value. The θ value can be reconstructed from the order of points in each 2D scan taking the point count and the resolution into account. In summary: The Cartesian coordinates (x, y, z) of a scan point corresponding to the spherical coordinates ϕ and θ neighboring scan line distance values are stored in a matrix structure. This structural knowledge about the data allows us to connect neighboring points in a single 360 degree scan to construct the mesh. Unsensed or removed points are marked with *NaN* to maintain the matrix structure. Another advantage of the organized structure is that it enables a fast and robust computation of point normals by using integral image techniques such as *Average 3D Gradient*, *Average Depth Change* or *Covariance Matrix* as described in Holzer et al. (2012).

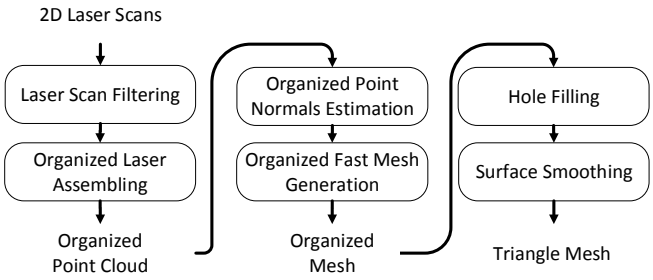


Fig. 3. Data flow for organized point cloud and *Organized Fast Mesh* generation.

4.2 Organized Fast Mesh

The used fast organized mesh generation is similar to the irregular triangular mesh (ITM) used in Gingras et al. (2010) and Breitenmoser and Siegwart (2012). The basic assumption is that ϕ and θ of neighboring scan lines represent points that are often neighbors in the real world. To address the problem of shadow edges - edges that violate continuity, e.g. connecting two objects, one in the front and one in the back - we implemented rules that filter these edges before insertion.

After the raw mesh is created, we fill up the robot's shadow in the input data. We assume that the robot stands on solid ground, since it successfully navigated to the current position. First, the shape of the robot shadow is extracted by finding the first transition from a *NaN* value to a valid one and following the contour line defined by this transition (*NaN* to valid) until the contour is closed. Once we found the robot shadow contour, we triangulate the 3D contour polygon and to fill it up, as shown in Fig. 6. The robot produces a shadow in the input scan, filled up with the red triangles in the reconstruction. The filled up mesh is then used as input for the computation of a *Navigation Mesh*.

5. NAVIGATION MESH

5.1 Graph Half Edge Mesh Generation

The basic idea behind the *Navigation Mesh* is to reduce the challenge of planning in uneven and rough terrain to a graph theoretical problem. We combined the well known Half Edge Mesh structure (Wiemann et al., 2012) with two different graphs, a vertex graph $\mathcal{G}_v$ and a triangle graph $\mathcal{G}_t$, resulting in a *Graph Half Edge Mesh* structure. The underlying triangle mesh is represented by a set of triangles $\mathcal{F}$ and a set of vertices or points $\mathcal{P}$ in 3D space. The vertex graph $\mathcal{G}_v = (V_v, E_v)$ connects all vertices, which are connected by triangle edges in the mesh with $E_v = \{e^{(i,j)} | \, i$ and $j \in f \in \mathcal{F}\}$, forming a network of all vertices in the triangle mesh. The set of graph nodes $\mathcal{V}_v$ corresponds to the set of vertices in the mesh. Similarly, the triangle graph $\mathcal{G}_t = (V_t, E_t)$ represents a network of mesh triangles with $deg(v) \leq 3 \, \forall \, v \in V_t$, since each triangle can have a maximum of three triangle neighbors.

Within the graph, we encode various characteristics of the surfaces into several layers of navigation costs for all edges that are fused into a single cost function $c^{(i,j)}$ for each edge $e^{(i,j)} \in E$ connecting the graph nodes i and j. The different cost functions are depending on values like the distance between two vertices or faces, the riskiness of accessing certain vertices or faces, and the local roughness assigned to vertices or faces. Graph edge costs can be estimated by taking distances or height differences into account. Graph node costs correspond to local roughness, local height difference and riskiness. Riskiness expresses forbidden areas in a gradual way. Additionally, each triangle and vertex can be associated with a normal vector that can be included in the computation of trafficability. In particular, these normal vectors (cf. Sec.4) are used to calculate the local roughness for each vertex considering their respective local network. After fusing these layers into a single cost function, we can apply standard graph search algorithms for planning. The different cost layers are described in detail in the following sub sections.

5.2 Lethal Obstacle Marking and Inflation

After the initial mesh is created, we mark inaccessible areas or positions on the surface. Since different robots are able to traverse different kinds of surfaces, the physical abilities of the used robot have to be considered when determining the costs within the mesh planning graph. We define all borders of the *Navigation Mesh* as lethal as the environment beyond the sensed area is unknown and hence potentially dangerous. Borders could result from shadows in the laser scans or by the limited range of the scanner. On the basis of the robot's abilities, high graph node costs are attributed to areas where obstacles are detected. First we calculate roughness and height differences on the surface. Then areas at heavy roughness and height difference costs are marked as lethal obstacles. Thus we are able to avoid dangerous areas like walls, crags, cliffs or ceilings implicitly.

After defining the lethal obstacles we inflate these forbidden areas so that we can treat the robot in navigation planning as a point. We developed a gradual inflation algorithm with an inner lethal inflation radius r_i and an outer maximum inflation radius r_o. In between the inner and outer radius a cosine function fades down the costs to zero continuously. The corresponding riskiness $r(v)$ for a graph node $v \in V$ is defined as shown in Eq. 1, where c_{max} is the total maximum cost.

$$r(v) = c_{max} \cdot \left(\cos \left(\frac{d_{lethal} - r_i}{r_o - r_i} \pi \right) + 1 \right) \qquad (1)$$

We use a wave front algorithm to calculate the smallest distance d_{lethal} to a lethal obstacle. For a simple optimization, the squared euclidean distance is used to differentiate between inside and outside of the inflation area. An example of a mesh with inflated regions around obstacles is shown in Fig. 4. The inflation radius to 0.3 m and the total inflation was set to a radius of 0.4 m, according to the specs of our robot.

5.3 Local Roughness Estimation

While we navigate in rough or uneven terrain, examination of roughness can be interesting for several reasons (cf.

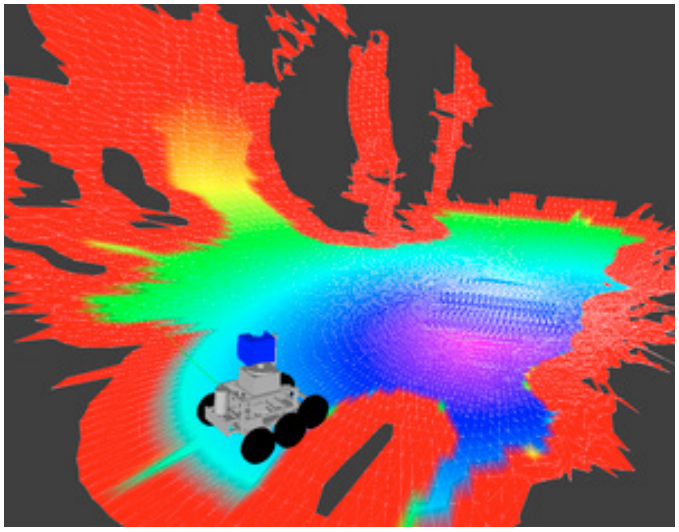


Fig. 4. 3D *Navigation Mesh* with a potential field to the goal, visualized in RViz. It shows a small section of the Botanical Garden of Osnabrück University with a cliff on the left and a planar region on the right. Lethal obstacles are red.

Fig. 5). It can be more convenient to prefer surfaces and areas with a higher continuity. This results in preferring paved roads or lanes which consequently results in faster locomotion. Roughness describes the continuity of the surface, independent of the alignment of the surface in world coordinates. That means that a higher local continuity of a local area is described by smaller roughness values for each vertex or triangle in the network. Therefore a wall can also exhibit continuity like the ground. In Eq. 3 we define the roughness $\rho(v)$ as the average angle of the fluctuation of the corresponding normals in a local neighborhood $\mathcal{N}_l(v)$ of the graph node $v \in V$. $\mathcal{N}_l(v)$ contains all graph nodes in a predefined local Euclidean range l, analogous to the height difference in the next section. Therefore the angle between the normals are calculated with $c_r^{(u,v)} = acos(\boldsymbol{n}(u) \cdot \boldsymbol{n}(v))$ for every edge $e^{(u,v)} \in E$ and a vertex average angle $\alpha_{av}(v)$ is defined as in Eq. 2, where $\mathcal{N}_d(v)$ are all direct neighbors of v.

$$\alpha_{av}(v) = \frac{\sum_{u \in \mathcal{N}_d(v)} c_r^{(u,v)}}{|\mathcal{N}_d(v)|} \tag{2}$$

$$\rho(v) = \frac{\alpha_{av}(v) + \sum_{u \in \mathcal{N}_l(v)} \alpha_{av}(u)}{|\mathcal{N}_l(v)| + 1} \tag{3}$$

5.4 Local Height Difference Cost Estimation

To enable navigation over staircases or other obstacles with a predefined maximum height, we use evaluate height differences, as proposed in Schwarz and Behnke (2014) for 2.5D digital elevation maps. They define *bumpiness* in a local environment for one cell. Analogously, we define bumpiness for one vertex v for a local vicinity $\mathcal{N}_l(v)$. Defining a direction $\boldsymbol{z}$ pointing upwards, we are able to calculate the maximum difference in a local field in the direction of $\boldsymbol{z}$ as shown in Eq. 4, where $\boldsymbol{p}(v)$ is the position vector corresponding to the graph node v. If $b(v)$ exceeds a

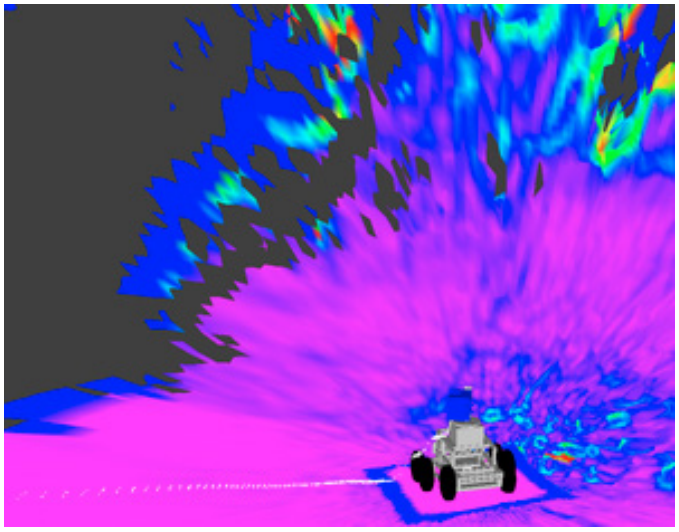


Fig. 5. Roughness of the environment. - Scene: A small section of the Botanical Garden of Osnabrück University.

specific threshold, we mark the node v as a lethal obstacle, because the robot is not able to handle the magnitude of height difference.

$$b(v) = \max_{u \in \mathcal{N}_l(v)} \left| \frac{(\boldsymbol{p}(v) - \boldsymbol{p}(u)) \cdot \boldsymbol{z}}{|\boldsymbol{z}|} \right| \tag{4}$$

6. ROBOT NAVIGATION

6.1 Global Path Planning

For global path planning, we have to define final edge costs based in distances, riskiness, local roughness and height difference. Depending on the weighting of the different cost functions, path planning prefers different paths with different characteristics. We determine the final edge costs $c^{(u,v)}$ for each edge $e^{(u,v)} \in E$ as described in Eq. 5. The coefficients k_ρ and k_b are used for weighting the local roughness ρ and the local height difference b. $d^{(u,v)}$ is the distance between the position vectors corresponding to u and v. Executing Dijkstra's shortest path algorithm or A* search on the graph G_t or G_v with edge costs $c^{(u,v)}$ results in a path with the smallest cost for the given coefficients, if a path can be found.

$$\begin{aligned} c^{(u,v)} = &((k_\rho \max(\rho(u), \rho(v))) + 1) \\ &\cdot ((k_b \max(b(u), b(v))) + 1) \\ &\cdot d^{(u,v)} \\ &+ |r(u) - r(v)| \end{aligned} \tag{5}$$

6.2 Local Robot Control

Following the idea of Breitenmoser and Siegwart (2012) it is easy to extract a vector field defined on the surface from our *Navigation Mesh*. After execution of the Dijkstra shortest path algorithm from the goal pose to the target pose, all smallest cost paths from every point to the target pose are available. For each node v, a predecessor $\Pi(v)$ is registered. At every point on the surface we can interpolate the next pose by evaluate the predecessors of

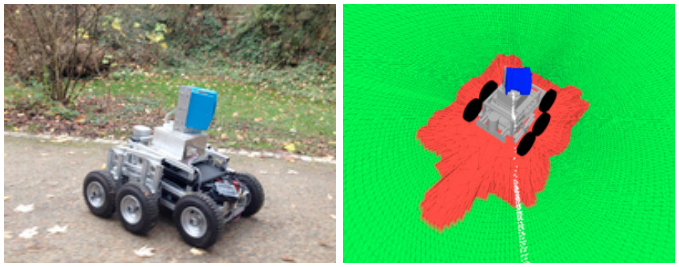


Fig. 6. (left) The robot *Pluto* based on the *VolksBot® XT*, equipped with a rotating 2D *Sick LMS 200*, a *Velodyne* 360 degree laser scanner and a *PhidgetSpatial 1044* inertial measurement unit. (right) *Organized Fast Mesh* displayed green; robot model of *Pluto*; filled up robot shadow displayed red; visualized in RViz.

the surrounding graph nodes. The resulting current path can be used to calculate velocity commands influenced by the costs of the path segments, the roughness and height differences. This vector field directly corresponds to the potential field as shown in Fig. 4.

7. EVALUATION

7.1 Robot Setup

We evaluated the trafficability estimation with a VolksBot XT that is equipped with a rotating 2D *Sick* laser scanner. The setup is shown in Fig. 6 (left). The aim is to navigate through rough terrain and to map its environment in a robust way. We implemented a mapping system which works in a stop-and-go scheme analogous to Schadler et al. (2014). In the stop-phase the robot assembles 360 degree 3D organized point clouds. These assembled clouds will be registered successively to one global point cloud. The organized point clouds enable a fast estimation of point normals (Holzer et al., 2012) and therefore a more robust registration result using point to plane ICP, taking the surface normals into account as shown in Segal et al. (2009).

7.2 Exemplary Path Planning Results

To demonstrate the ability of our robot to drive and plan in rough terrain, we tested the setup in the Botanical Garden of Osnabrück University. The garden is located in a former stone quarry and shows a variety of different surfaces at different levels of roughness ranging from road-like paved ways to small paths and cobblestone trails with steep stairs. To evaluate our approach, we chose three exemplary areas within the garden to show the benefits of our methods in different kinds of environments. Pictures of the evaluated scenes together with the computed roughness estimation, height difference map and fused values in the potential field approach are shown in Fig. 7. At each position, we took a 360 degree laser scan, computed the terrain classification and gave the robot a target goal within the scanned area to drive to. The goal poses are marked with a white arrow, the planned paths are rendered in black. The color coding shows the determined path costs for the different areas in a blue to red rainbow color gradient, i.e., blue and violet colors correspond to low values, red colored regions are areas with the highest path costs.

The first scene is a relatively smooth path with a steep cliff on the left side and a slope to the right. The second is small path with rough stones fallen down from a cliff covering the right side of the scene. The third example was taken on a crossroads of a larger path and a small path with a staircase. The chosen scenes show different characteristics. On the path, the scene is dominated by a large, sharply distinguished flat surface that is surrounded by undriveable areas. Fallen rocks form hazardous areas in the cliff scene that are not that well distinguished from the drivable areas. The slope towards the cliff raises relatively slow from the path towards the cliff. Similarly, the slope of the grass area raises smoothly from the path. In principle this region is drivable for our robot, but it should avoid the stairs in any case, as the height difference on the steps is too high.

7.3 Discussion

The results of the terrain classification shown in Fig. 7 nicely reflect the properties of the scanned environments. In the path scene, the driveable areas on the path are classified with low costs both in the roughness and height difference classification. The planned path is smooth and follows the middle of path. Since in this example both the height difference and roughness classification deliver similar drivability estimations, the resulting potential field also prefers the pathway. In the cliff scene the height difference and roughness metrics compute different costs. The height difference assigns relatively low costs to the slope towards the cliff, since the local height difference is low. However, the roughness estimation classifies this area as hazardous, due to the unevenness resulting from the fallen rocks. This example clearly demonstrates that a classification based on local height differences alone does not always deliver safe classification results. Consequently, using the fused information in the potential field, the planner prefers a route on the pathway. In the third example, both metrics punish the area with the steps and mark them dangerous.

The presented examples show that the presented metrics for the *Navigation Meshes* deliver coherent classifications results with respect the underlying classification heuristic. The example of the cliff data set shows that using a single metric may come to false drivability assumptions. However, the proposed fusion of both metrics in the potential field allows save navigation, if one the heuristics fails. That clearly demonstrates the strengths of the *Navigation Mesh* approach, since the underlying graph structure can encode different modalities (roughness, height difference, path length) in a single representation.

8. CONCLUSION AND FUTURE WORK

We introduced the *Organized Fast Mesh* as mesh generation method to compute a *Navigation Mesh* which can directly be used to perform path planning in the current field of view of the robot. In future research we will use the *Navigation Mesh* in a global mapping process to perform global path planning in a multilevel environment. The

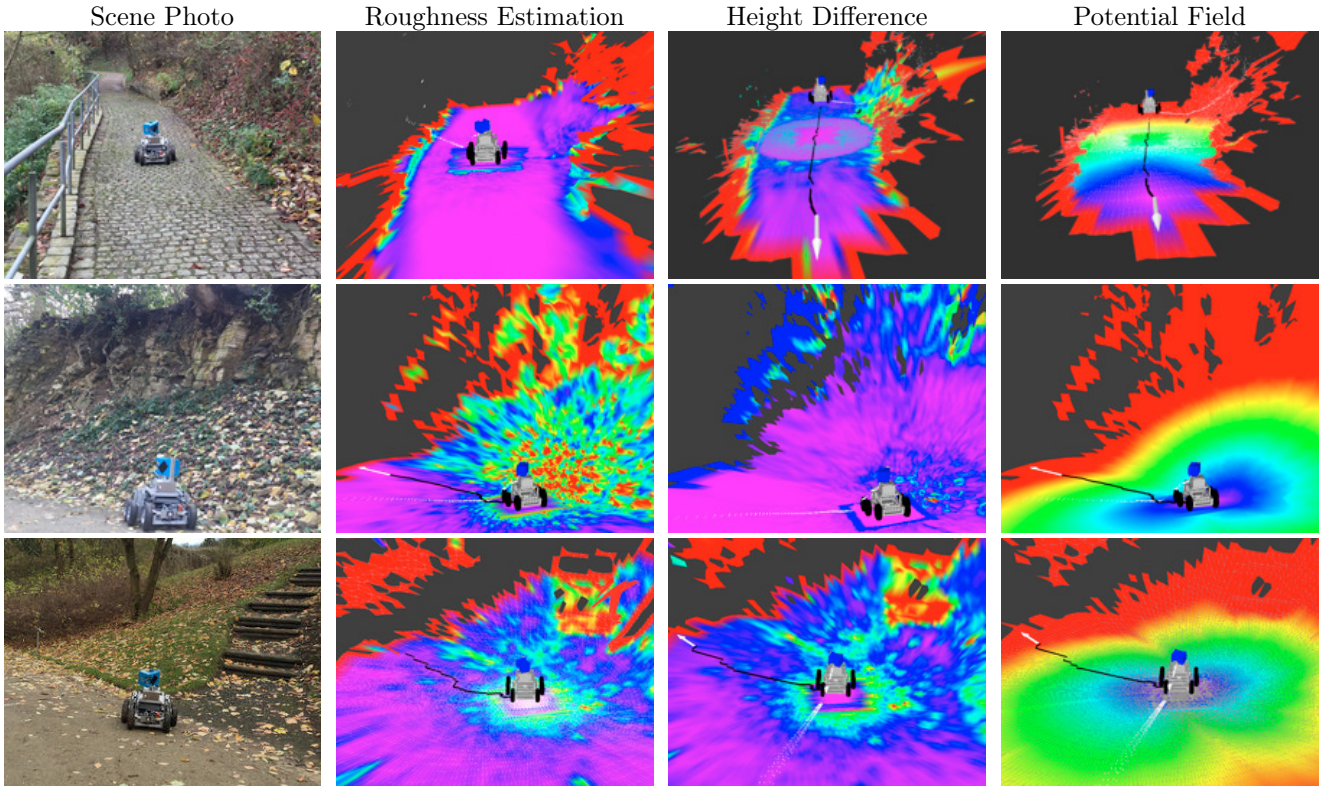


Fig. 7. Evaluation examples of the *Navigation Mesh* and its metrics. The left column shows a picture of the scene, the right three columns present the classification results.

Navigation Mesh is used to evaluate different drivability measures in a single data structure for path planning and local navigation. We evaluated the proposed method on real life application examples and demonstrated that our approach delivers reliable terrain classification and plans safe paths. In future work, we will integrate path optimization methods to create smoother trajectories. Additionally, the system has to be evaluated in varying environments and different use cases with different robots. However, the presented results show that the proposed mixture of triangle mesh and graph representation looks promising for future work in the area of drivability estimation and path planning. We plan to make the software available as open source for the robotic's community whilst adapting and extending it. Further tests will be done on other robot platforms to prove the benefits of the presented methods.

REFERENCES

Breitenmoser, A. and Siegwart, R. (2012). Surface reconstruction and path planning for industrial inspection with a climbing robot. In *Proc. CARPI 2012*.

Deeken, H., Pütz, S., Wiemann, T., Lingemann, K., and Hertzberg, J. (2014). Integrating semantic information in navigational planning. In *Proc. ISR/Robotik 2014*. München.

Gingras, D., Lamarchemarche, T., Bedwani, J., and Dupuis, E. (2010). Rough terrain reconstruction for rover motion planning. In *Proc. CRV 2010*, 191–198.

Holz, D. and Behnke, S. (2012). Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Proc. IAS-12*.

Holzer, S., Rusu, R.B., Dixon, M., Gedikli, S., and Navab, N. (2012). Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Proc. IROS 2012*, 2684–2689. IEEE.

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

Morisset, B., Rusu, R.B., Sundaresan, A., Hauser, K.K., Agrawal, M., Latombe, J.C., and Beetz, M. (2009). Leaving flatland: Toward real-time 3d navigation. In *Proc. ICRA 2009*. IEEE.

Rusu, R.B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *Proc. ICRA 2011*. IEEE.

Schadler, M., Stückler, J., and Behnke, S. (2014). Rough terrain 3d mapping and navigation using a continuously rotating 2d laser scanner. *KI*, 28(2), 93–99.

Schwarz, M. and Behnke, S. (2014). Local navigation in rough terrain using omnidirectional height. In *Proc. ISR/Robotik 2014*.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Proc. RSS 2009*. Seattle, USA.

Wiemann, T., Lingemann, K., and Hertzberg, J. (2013). Automatic map creation for environment modelling in robotic simulators. In *Proc. ECMS 2013*. Älesund.

Wiemann, T., Lingemann, K., Nüchter, A., and Hertzberg, J. (2012). A toolkit for automatic generation of polygonal maps - las vegas reconstruction. In *Proc. Robotik 2012*, 446–451. VDE Verlag, München.

Wiemann, T., Nüchter, A., Lingemann, K., Stiene, S., and Hertzberg, J. (2010). Automatic construction of polygonal maps from point cloud data. In *Proc. SSRR 2010*. Bremen.