Continuous Multilayer and Multilevel Shortest Path Vector Field Navigation on 3D Triangular Meshes for Mobile Robots

Sebastian Pütz, Thomas Wiemann, Malte Kleine Piening, and Joachim Hertzberg
Knowledge Based Systems, Institute of Computer Science, Osnabrück University, in collab. with
Plan-Based Robot Control, German Research Center for Artificial Intelligence, DFKI

***Abstract*— We present an optimal and efficient approach to compute continuous shortest path vector fields on arbitrarily shaped 3D triangular meshes for robot navigation in complex real-world outdoor environments. The continuity of the vector field allows to query the shortest distance, direction and geodesic path to the goal at any point within the mesh triangles, resulting in accurate paths. In order to avoid impassable areas, our wavefront propagation method runs on a modular extendable multilayer map architecture taking different geometric cost layers into account. We describe the mathematical foundation of the geodesic distances and continuous vector field computation and demonstrate the performance in real-world and multilevel environments on our campus with a tunnel, ramps and staircases, and in a difficult, steep forest area with a stone quarry. For reproducibility, we provide a ready-to-use ROS software stack as well as Gazebo simulations.**

I. Introduction

Robots operating outdoors in forests, agriculture, or rescue operations, etc. have to cope with variable and uneven terrain. In such applications it is important to determine the drivability of the terrain between the current pose and the given goal pose. The representation of 3D structures such as ramps, curved surfaces, stairways, bridges, underpasses, tunnels or walls that are not horizontally aligned with the robot opens up new challenges because they cannot simply be projected down to a 2D map. Decomposing the environment into multiple topologically connected 2D maps is often done, but shifts the problem from path planning in a complex 3D domain to disassembling the environment into a topological graph connecting the resulting 2D sub-maps. We aim for a general map representation and present an approach for robot navigation in complex environments on 2D-manifolds represented as 3D triangular meshes. We contribute the following aspects that work together as a complete applied system:

- An optimal, highly-efficient continuous vector field computation that defines the shortest geodesic path to a goal at any point on the 3D mesh surface.
- A modular, extendable multilayer mesh map to model the passability of complex outdoor environments through layers such as roughness, height differences, steepness, etc.
- A modular mesh navigation ROS software stack that integrates *Move Base Flex* (MBF) [1] and our layered mesh map, as well as our shortest path vector field planner and controller.
- 3D real-world datasets and Gazebo simulations to reproduce our experimental results.

The proposed method is based on the wavefront propagation principals of the *Fast Marching Method (FMM)* [2]. The computation of the vector field and path planning has an asymptotic runtime of $\mathcal{O}(n \log n)$, where n is the number of passable vertices in the mesh, with respect to the navigability defined by the layers. The resulting vector field can be accessed in a continuous fashion using barycentric coordinates and is not restricted to the topology of the mesh. Our ROS mesh navigation software stack allows to simply integrate the novel path planning and motion control into higher level robotics applications, e.g. by using the *SMACH* task level architecture [3], or behavior trees [4].

II. Related Work

During the last decades many path planning algorithms have been proposed which can roughly be categorized as *grid-based*, *sample-based* or *geodesic* path planning.

A. Path Planning in 2D and 3D Occupancy Gird Maps

Occupancy grid maps are established since decades [5]. Encoding occupancy probabilities is sufficient to support probabilistic localization and navigation methods [6], [7]. In 1993 and 1995 *D** and *Focused D** have been introduced by Stentz [8] as replanning methods distinguishing between known and unknown obstacles. *D** has been extended to *Focused D** by using a *focusing heuristic* in an *A** manner. In 2002, the *Lifelong Planning A* (LPA*)* and *D* Lite* replanning algorithms have been introduced by Koenig and Likhachev [9]. In addition to the typical *A** distance-to-goal cost estimate, *LPA** maintains a look-ahead value and consecutively propagates updates whenever these two values become inconsistent. *D* Lite* extends *LPA**, as it switches the start to the goal cell similar to *Focused D**.

However, the above algorithms are constrained to the grid topology, only allowing for 45° transitions to the eight direct cell neighbors. Shortest paths constrained to the edges of 8-neighbor transitions can be around 8% longer in 2D grid maps and constrained to the edges of 26-neighbor transitions in a 3D occupancy grid map can be around 13% longer than the actual shortest path in the represented environment [10]. Consequently, Ferguson et al. [11] introduced *Field D** in 2005. It uses linear interpolation, defines nodes at cell corners

The DFKI Niedersachsen Lab (DFKI NI) is sponsored by the Ministry of Science and Culture of Lower Saxony and the VolkswagenStiftung
{spuetz, twiemann, mklpiening, joachim.hertzberg}@uos.de

and takes varying cell costs into account. In 2007 and 2010 the any-angle algorithms, *Theta**, and *Lazy Theta** have been introduced by Nash et al. [12], [10]. Unlike the previous ones, both *Theta** algorithms perform line-of-sight checks in which nodes are also defined at cell corners. In 2013 *Anya* has been introduced by Harabor and Grastien [13] as an optimal and exact any-angle path planning method for simple occupancy grid maps.

In the context of ROS and corresponding map implementations, Fankhauser et al. introduced Grid Map [14] in 2016 as a multi-layered map representation using an efficient 2D ring buffer representing a 2.5D grid map with floating point precision. It addresses many of the limitations of the established ROS map representation `costmap_2d` [15]. Beside that, 2.5D-map-related environment analysis methods to represent digital elevation models (DEMs) have been successfully applied in rough terrain navigation [16]. Although they reflect the topography of a surface, DEMs are only able to encode one surface level with a fixed discrete resolution. Encoding free space between several levels and the levels itself is not possible. This limitation can be overcome by 3D occupancy maps using octrees or voxel hashing [17], [18] and triangular meshes with no fixed discrete resolution.

B. Geodesic Path Planning on Triangular Meshes

Recently, methods such as [19], [20], [21], [22] have been developed to compute triangular meshes from sensor data, including large-scale environments. Although such maps are now available, they are mainly used for simulation or rendering, but rarely for navigation.

Brandao et al. [23] present path planning or triangular meshes for legged robots in an industrial environment, which uses A* to compute a sequence of triangles between the start and goal triangle outperforming state-of-the-art sample-based planners. Similar to the topology restrictions in grid maps, simple graph based planning methods running on 3D triangular meshes are usually topologically restricted, too. Thus, Dijkstra running on triangular meshes can lead to sub-optimal paths with respect to the shape of the represented surface, as pointed out in [24]. Finding the actual topologically-unrestricted shortest path on surfaces represented by meshes – where paths cross triangles – is also known as computing geodesic distances or paths.

Modern path planning methods running on 3D triangular meshes have been advanced in the last decade. Yershov and LaValle [25] present a version of *FMM* and call it feedback planning, in which the wavefront propagation starts at the goal position and the distance field, approximated by linear interpolation within the triangles, defines the approximate shortest path from each position to the goal. Yershov and Frazzoli [26] build on top of the *Hamilton-Jacobi-Bellman* equations, *FMM*, and combine it with an adaptive mesh-refinement to improve the resolution of an initial simplicial mesh. Compared to many modern sam ple-based methods, it is also asymptotically optimal, which means it converges to the optimal solution, but it outperforms *RRT** [27] and *PRM** [28]. Xu et al. [29] present the *Fast Wavefront Propagation (FWP)* framework for several path planning algorithms to improve the performance of geodesic algorithms like the exact *Mitchell-Mount-Papadimitriou (MMP)* [30] algorithm, or the exact *Chen-Han (CH)* [31] algorithm. Their framework organizes windows with a bucket structure to process a large number of windows simultaneously as described in detail in [29]. Additionally, in [29], the asymptotic runtime, space and overhead complexities of the most popular algorithms are compared. *FMM* running on triangular meshes has an overhead of $\mathcal{O}(n)$, a space complexity of $\mathcal{O}(n)$ and time complexity of $\mathcal{O}(n \log n)$, and is thereby the fastest known solver with respect to the asymptotic runtime. Bhattacharya [32] presents *S** which computes shortest paths in configuration spaces of arbitrary topology, geometry, and dimension of a simplicial complex. It uses refinements to improve results and has an asymptotic runtime of $\mathcal{O}(n(\log n + d^4))$ where d is the average degree of a node. However, *S** has only been evaluated in 2D (and projected 3D to 2D), and *Theta** has a better runtime, see [32] for a comparison.

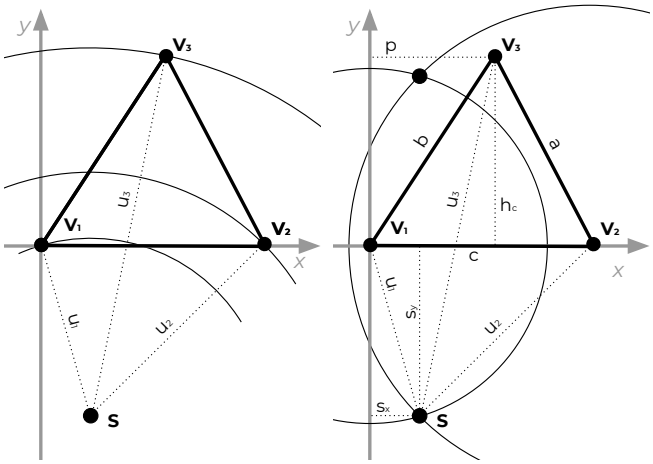
All the above methods were evaluated using artificial data in the respective publications. We close the gap and propose an optimal and fast method for shortest path vector field planning on arbitrary meshes to leverage robot navigation in complex real-world outdoor environments. Furthermore, our approach does not use refinements and does not run on 2D projections from a pre-processing step.

III. Wavefront Propagation

Inspired by physics, alternative approaches have been introduced. The Hamilton Jacobi partial differential equation is an alternative formulation of classical mechanics and equivalent to, e.g. Newton's laws of motion, Lagrangian mechanics and Hamiltonian mechanics. It reduces to the *Eikonal* equation describing physical waves if the formulations depend on the position state variable only, as described in [33]. The first two known algorithms that provide solutions for the *Eikonal* equation are *Tsitsiklis' algorithm* introduced in 1994 [34] and *Sethian's Fast Marching Method (FMM)* [2] introduced in 1996, solving the isotropic control problems using first-order semi-Langragian discretizations on Cartesian grids, as described in [35]. Many *FMM* variants have been introduced in the two last decades, e.g. for car-like robot-path-planning [36], and *Voronoi* partitioning [37]. The original *FMM* by Sethian [38], [2] was developed as a fast level set method computing a *cost-to-go* function in a way of a wavefront advancing outwards. We adapted *FMM* to compute a cost-to-go scalar field u for a single source and to simultaneously construct a corresponding vector field $\vec{d}$ during wavefront propagation.

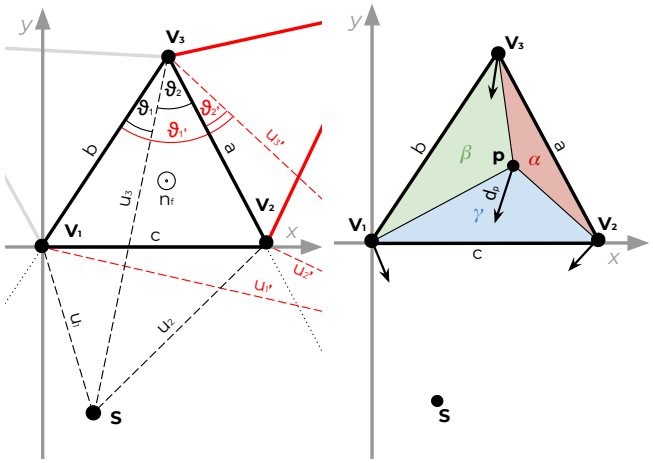
A. Fast Marching Method on Triangular Meshes

Using *FMM* for path planning, we model a wavefront starting at a single source s, as sketched in Fig 1a. Similar to replanning-algorithms or feedback-planning, we start the propagation at the goal pose, i.e. at s. A mesh consists of vertices, edges, and triangles, with $M = (V, E, F)$. *FMM*



(a) Wavefronts (b) Update Step

Fig. 1: The propagating wavefront arcs and the implicit unfolded source s (a). The distances u_1 and u_2 to s can be used as radii for intersecting arcs around v_1 and v_2 to compute u_3 (b).



(a) Direction Computation (b) Vector Interpolation

Fig. 2: The angle θ is used to compute the direction vector for v_3 (a). The direction vector $\vec{d}_p$ for a point p is the linear combination of the three directions and the barycentric coordinates (b).

uses three sets: The unprocessed set $V' \subseteq V$, the close set $Q \subseteq V$ and the fixed set $S \subseteq V$. Initially, V' contains all vertices which are labeled as passable. Q is implemented as a min-heap map sorting the vertices by the temporary distance estimate u and thereby allowing fast distance updates in logarithmic time and extracting the vertex with the smallest distance value in constant time. The values in the heap Q may decrease if a solution with a smaller distance is found. The min-heap lookup data structure Q limits the wavefront propagation to an asymptotic runtime of $\mathcal{O}(n \log n)$ where n is the number of vertices $|V'|$. Practically, Q contains far fewer vertices at runtime, as it only contains vertices associated with the current wavefront. S contains all vertices with the final and optimal shortest path distances to $s \in V'$.

Initialization: First, the triangle $\triangle f_s \in F$ which contains s – denoted as $|\triangle f_s| \ni s$ – is picked. Second, the three vertices v_i of $\triangle f_s$ are added to Q with the Euclidean distance: $\exists! \triangle f_s \in F$, $|\triangle f_s| \ni s$; $Q(v_i) \leftarrow ||s - v_i||_2 \forall v_i \in \triangle f_s$.

1.) The vertex $v \in Q$ with the smallest value in the close set Q is added to the fixed set S and erased from Q.

2.) Each adjacent neighbor in the fixed set $n \in \mathcal{N}_d(v) \cap S$ is connected to v through an edge $(v, n) \in E$. This edge corresponds to one or two triangles $\triangle f_i \in F(v, n) \subset F$. We pick $\triangle f_i = (v, n, v') \in F(v, n)$ with $v' \notin S$.

3.) Next, we compute the distance $u_{v'}$ of $v' \in |f_i|$, which is either unprocessed with $v' \in V'$ or in the close set Q and insert or update the sorted close set with $Q(v') \leftarrow u(v')$ if this decreases the distance value of v' in Q. We loop from step 2) until all neighbors n have been processed and then back to 1) until Q is empty.

Note, $\triangle vnv'$ corresponds to $\triangle v_1 v_2 v_3$ in the update step shown in Fig. 1b and described next.

B. Single Source Update Step

Novotni et al. [39] introduce an update step for a single source, taking the plane defined by $\triangle v_1 v_2 v_3$ into account. We extend the single source update step to represent the correct distances as follows: Assuming the distances $u_1, u_2 \in \mathbb{R}$ of $v_1, v_2 \in S$ are already computed, a solution for the distance $u_3 \in \mathbb{R}$ of $v_3 \in Q \cup V'$ can be found by implicitly unfolding the source s onto the plane of $\triangle v_1 v_2 v_3$. The calculation of u_3 is sketched in Fig. 1b. Let $a = ||v_2 - v_3||_2$, $b = ||v_1 - v_3||_2$ and $c = ||v_1 - v_2||_2$. Let T be the triangle's coordinate system in $\mathbb{R}^2$ with v_1 at $(0, 0)$, v_2 on the x-axis with $(c, 0)$, and let v_3 have a positive y value with (p, h_c). We are now looking for the coordinates $(p, h_c) \in T$ and $(s_x, s_y) \in T$ of v_3 and the source s. The source coordinates (s_x, s_y) can be found by the intersection of the circles given by the radii u_1 and u_2 around v_1 and v_2 by solving $s_x^2 + s_y^2 = u_1^2$ and $(c - s_x)^2 + s_y^2 = u_2^2$ to the variables s_x and s_y, or by using the Pythagorean theorem. Note, the heights h_c and s_y could also be calculated by Heron's formula as an alternative formulation. Next, (p, h_c), the coordinate of v_3 in T, has to be calculated by solving $b^2 = h_c^2 + p^2$ and $a^2 = (c - p)^2 + h_c^2$. Since we are interested in the distance from the source s to v_3 in T, we choose the solution where $h_c \geq 0$ and $s_y \leq 0$. This results in a solution for (p, h_c) shown in Eq. 1 and Eq. 2 and in a solution for (s_x, s_y) shown in Eq. 3 and Eq. 4. Finally, u_3 is calculated as the distance of (p, h_c) to (s_x, s_y) according to Eq. 5. Note, in [39] (p, h_c) is not used in the update step. We added the missing part to the algorithm's update step which leads to a correct solution of the distance calculation for v_3. Mitchell et al. [30] showed in Lemma 3.3 that a planar unfolding of a geodesic path is a straight line segment. Since we are computing the straight line, i.e. Euclidean distance (and its angle θ) on the implicit unfolded triangle between (p, h_c) and (s_x, s_y), see Eq. 5, we assign the shortest path from s to each accessed vertex v'.

$$p = \frac{b^2 + c^2 - a^2}{2c} \quad (1) \qquad h_c = \sqrt{b^2 - p^2} \quad (2)$$

$$s_x = \frac{u_1^2 + c^2 - u_2^2}{2c} \quad (3) \qquad s_y = -\sqrt{u_1^2 - s_x^2} \quad (4)$$

$$u_3 = \sqrt{(p - s_x)^2 + (h_c - s_y)^2} \tag{5}$$

Every time u_3 decreases the distance, $u_3 < Q(v_3)$, the close set Q is updated. The wavefront moves forward monotonically due to the nature of *FMM* and the fulfilled triangle inequality, thus our approach can be used to generate a global vector field with s being the local and global minima.

IV. Vector Field Continuity

An agent would automatically use the shortest path from any point on the surface by following the vector field which corresponds to $\nabla u(x)$ if a path exists. The continuous vector field is computed during the update step described above and described in the following. A direction vector $\vec{d}_p$ at any query point $p \in \mathbb{R}^3$ projected onto the surface is pointing towards the goal pose s and can be calculated by a linear combination of the three direction vectors of the corresponding triangle after the wavefront propagation.

A. Direction Vectors

The computation of the direction vectors towards the goal pose is sketched in Fig. 2. During the update step described above, two angles $\theta_1 = \angle s v_3 v_1$, and $\theta_2 = \angle s v_3 v_2$ with respect to the sides a and b are computed by applying the law of cosines using the distances u_1, u_2 and the newly computed distance u_3, as well as the sides a, b, and c of $\triangle v_1 v_2 v_3$ as denoted in Eq. 6. The corresponding dashed triangles $\triangle u_3 b u_1$ and $\triangle u_2 a u_3$ are shown in Fig. 2a.

$$\theta_1 = \arccos\left(\frac{u_3^2 + b^2 - u_1^2}{2u_3 b}\right) \qquad \theta_2 = \arccos\left(\frac{u_3^2 + a^2 - u_2^2}{2u_3 a}\right) \tag{6}$$

To efficiently access vertex attributes, all vertices correspond to a unique index $\iota_i \in \mathbb{N}_\iota \subset \mathbb{N}$. The resulting direction angle θ of v_3 is set to θ_1 if $u_1 < u_2$ and to $-\theta_2$ otherwise and stored in a map $m_\theta : \mathbb{N}_\iota \to \mathbb{R}$. The negative $-\theta_2$ indicates the opposite direction of rotation. The respective predecessor $\rho \in V'$ is set to v_1 if $u_1 < u_2$, and to v_2 otherwise and is mapped by its index in $m_\rho : \mathbb{N}_\iota \to \mathbb{N}_\iota$.

Due to corner cases, e.g., obstacles, mesh borders, etc, we have to check $\theta_1 + \theta_2 \leq \angle v_2 v_3 v_1$. If this check is violated, $\overline{v_3 s}$ is not cutting $\triangle v_1 v_2 v_3$, as illustrated in Fig. 2 with the dashed red lines and $\theta_{1'}$, $\theta_{2'}$, and $u_{1'}$, $u_{2'}$, $u_{3'}$. This will result in updating $u_3 = u_1 + b$ and $\theta_1 = 0$ if $\theta_1 < \theta_2$, and $u_3 = u_2 + a$ and $\theta_2 = 0$ otherwise. Note that these values may change if shorter distance values for v_3 are found as long as $v_3 \notin S$.

After the wavefront propagation the respective side a or b is rotated around the triangle's normal $\vec{n}_f$ using Eq. 7. This results in the direction vector map $\vec{m}_d(\iota)$, where $V(\iota)$ is the vertex with the vertex index ι.

$$\vec{m}_d(\iota) = R(\vec{m}_{n_f}(\iota), m_\theta(\iota))(V(m_\rho(\iota)) - V(\iota)) \tag{7}$$

Finally, a linear combination of these direction vectors with barycentric coordinates allows to access the vector field in a continuous fashion.

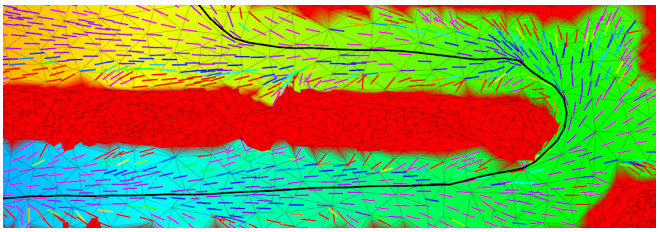


Fig. 3: Vector field, scalar field and the backtracked path. Cutout of the left ramp of the physics campus dataset (c.f. Fig. 6).

B. Barycentric Coordinates

To access attributes, e.g. directions, costs, distances, etc. for any point $p \in \mathbb{R}^3$ on the surface between vertices, we use a linear combination of barycentric coordinates. In our implementation, we use Heidrich's method [40] in order to efficiently compute projected barycentric coordinates of any query point p and $f = \triangle v_1 v_2 v_3$ as stated in Eq. 8. Let $\vec{u} = v_2 - v_1$, $\vec{v} = v_3 - v_1$, $\vec{w} = p - v_1$ and $\vec{n} = \vec{u} \times \vec{v}$.

$$\gamma = \frac{(\vec{u} \times \vec{w}) \cdot \vec{n}}{\vec{n}^2} \qquad \beta = \frac{(\vec{w} \times \vec{v}) \cdot \vec{n}}{\vec{n}^2} \qquad \alpha = 1 - \gamma - \beta \tag{8}$$

The projection of p onto the plane is then given by $p' = \alpha v_1 + \beta v_2 + \gamma v_3$. Moreover, this representations allows to easily check if p lies inside ($p \in |f|$), or outside ($p \notin |f|$) the triangle f. The point p is located inside f if $0 \leq \alpha \leq 1 \wedge 0 \leq \beta \leq 1 \wedge 0 \leq \gamma \leq 1$ is fulfilled and outside otherwise.

Finally, the direction vector $\vec{d}_p$ of p is a linear combination (Eq. 8) of 1.) the normalized direction vectors of the corresponding triangle vertices from Eq. 7, where ι_1, ι_2, and ι_3 are the respective indices of $\triangle f$, and 2.) the barycentric coordinates, as sketched in Fig 2b.

$$\vec{d}(p) = \alpha \cdot \hat{m}_d(\iota_1) + \beta \cdot \hat{m}_d(\iota_2) + \gamma \cdot \hat{m}_d(\iota_3) \tag{9}$$

In this way, the path planner and the controller can access the vector field at any point on the surface in a continuous fashion. An example of such a field is shown in Fig. 3, where obstacles are colored red. Next, we describe the integration into the ROS mesh navigation software stack.

V. Mesh Navigation

Our wavefront *Vector Field Planner (VFP)* implements the described method and computes a discretized geodesic path as shown in the experimental results beside the vector field $\vec{d}$ and the scalar distance field u. Incrementally moving p on $\triangle f$ in the direction of $\vec{d}_p$ until $p \notin |\triangle f|$ or until s is reached. Whenever the barycentric coordinates indicate that p is located inside a neighboring triangle, $\exists \triangle f_n \in \mathcal{N}_f(f)$ with $|\triangle f_n| \ni p$, then $\triangle f$ is updated to $\triangle f_n$. To move the robot along the shortest path vector field $\vec{d}$, we developed a *Vector Field Controller (VFC)* where p is associated with the current robot pose.

VFP and *VFC* are plugins for our `mesh_navigation`[1] stack integrating MBF[2] [1] and the layered mesh map. It allows to easily exchange, extend and configure planners, controllers and the underlying layered mesh map according

[1] `https://github.com/uos/mesh_navigation`
[2] `https://github.com/magazino/move_base_flex`

to robot abilities and the complexity of the terrain. Mesh layers to model *roughness*, *height differences*, *steepness*, *elevation*, as well as an *inflation* layer to inflate impalpable areas with the robot's diameter are provided, see [41] for layer definitions. Thus vertices that represent static obstacles by exceeding a robot specific threshold in the respective layer-metric are marked as *lethal* and not added to V'. To visualize the mesh, its layer attributes, and computed scalar fields we developed *RViz* plugins together with ROS mesh messages, bundled as `mesh_tools`[3], see [42].

In order to precisely localize a robot in uneven real-world environments on the surface and in real-time, we adapted the *LeGO Loam* [43] approach to use our mesh map for localization only. The last 20 laser sweeps combined in one cycle by the *LeGO loam* lidar odometry are aligned with the existing map by using a *point-to-plane ICP* [44]. Next, the transformation fusion method in [43] is used to combine our ICP-optimized pose and the lidar odometry to the current robot pose.

VI. Experimental Results

We evaluated our *VFP* and *VFC* in various outdoor environments in simulation, as well as in the real world.

A. Evaluation Setup

We choose five challenging outdoor environments (c.f. Tab. I) to demonstrate the advantages of our mesh navigation stack. The evaluation environments were recorded with a high resolution laser scanner, the *Riegl VZ400i*, producing highly detailed and colored point clouds, which were aligned with *point-to-plane* ICP and a pose graph optimization by *Choi* et al. [45], [22]. The registered point clouds were reduced from several gigabytes to a manageable size using a voxel grid with $8\,\mathrm{cm}$ resolution. Next, the respective dataset was reconstructed from the set of aligned point clouds with a *Poisson* reconstruction [46] implemented in *Open 3D* [22] and finally reduced from several gigabytes to a manageable data size using *planar quadric edge collapse* [47]. Finally, the triangular mesh layers were computed as described in [41] and stored together with the mesh geometry in a compact *HDF5* map file according to [48]. Due to space constraints, only two datasets are shown in the figures in this paper. However, the other datasets (cf. Tab. I), corresponding planner, controller, and mesh layer configurations, as well as the respective start and goal poses are provided to reproduce the presented results[4].

The runtime and the path length of our *VFP* are compared to the exact *MMP* [30] shortest path planner and the topology-restricted *Dijkstra*, running on our 3D triangular mesh map with the same configurations. All mesh map layers are computed or loaded and combined dynamically when the map is initialized. Similar to layered grid maps, our layered mesh map defines occupied or obstacle regions using thresholds that define the so called *lethal* areas, which are marked in red in all corresponding figures. The distance field is encoded with rainbow colors in the respective figures.

[3] `https://github.com/uos/mesh_tools`
[4] `https://github.com/uos/pluto_robot`

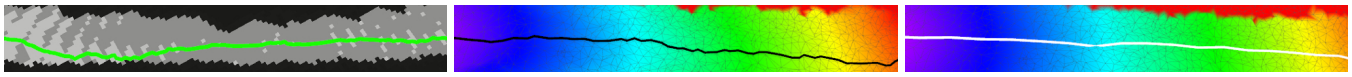
B. Evaluation Results

The *Stone Quarry Brockum* dataset is shown in Fig. 5, it was recorded in a forest in Brockum with an old stone quarry with multiple levels and overhanging structures like tree branches (cf. Fig. 5a). Although pathways are passable, the terrain here is challenging with many differently curved slopes and varying elevations. In the presented scene, the path forks around a hill structure with trees. A detailed cutout of the left pathway is shown in Fig. 4. Here, running Dijkstra on our mesh map (cf. Fig. 4b) already returns a smoother path than the standard ROS *Global Planner* running on the layered `costmap_2` with a 2.5D DEM layer (cf. Fig. 4a). To identify the characteristics shown in Fig. 5b and 5c, we cut out overhanging structures. Especially for the 2.5D approach, these mesh parts have to be filtered out to enable path planning at all in such highly curved and slope-varying terrains with overhanging structures, which is not necessary for planning in full 3D, as shown in the following. As shown in Fig. 5, out *VFP* computes smooth lines on the surface around static obstacles resulting in a geodesic path which is not restricted to the mesh topology. The shown *VFP* path is around $4\,\mathrm{m}$ shorter than the one computed with *Dijkstra* (cf. Fig. 5, Tab. I).

The *Physics Campus Westerberg* dataset shown in Fig. 6 is used to demonstrate path planning in multilevel large-scale urban outdoor environments. Here, we perform path planning from the upper to the lower level over a number of slightly inclined narrow ramps and finally through a tunnel. Fig. 3 shows a detailed cutout with the vector field and scalar field of the upper left ramp of the scene in Fig. 6. Again, our *VFP* computes a smooth geodesic path which is around $10\,\mathrm{m}$ shorter than the Dijkstra path, which is limited to the mesh topology. All path length and runtime results are presented in Tab. I with respect to the mesh size and dimensions. *VFP* results in smoother and shorter paths at the cost of around 1.8 times longer planning time, whereas the exact *MMP* is marginally shorter while requiring much more time.

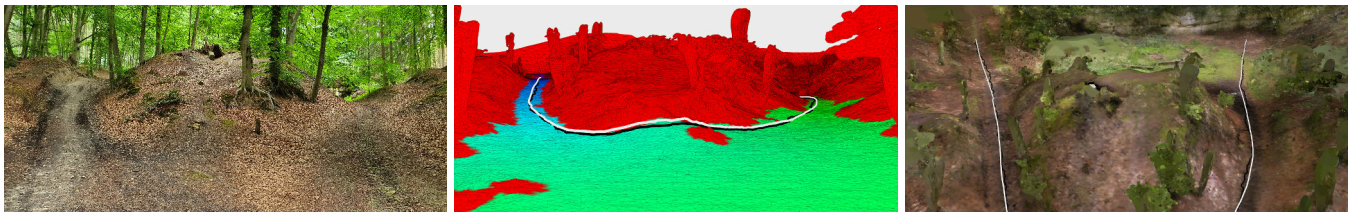
VII. Conclusion

The novel update step and integrated vector computation lead to an optimal goal vector field which can be accessed in a continuous fashion by using barycentric coordinates. The vector field $\vec{d}$ and distance field u define the shortest distance and direction to the goal at any accessible point on the surface and can be used for path planning and motion control. Our *VFP (Vector Field Planner)* implements the described method and computes an optimal and nearly exact shortest path to the goal on our modular 3D mesh map which models real world environments, while our *VFC (Vector Field Controller)* follows the computed vector field by querying the orientation to the goal and costs around the current pose. We provide a fully integrated but modular extendable system for ROS using *Move Base Flex* with our layered mesh map to load planner, controller and layer plugins, as requirements may change for differently environments or robot architectures. Cost layers are implicitly integrated into the distance field and vector field computation to avoid



(a) *Global Planner* on 2.5D DEM layer (b) Dijkstra on 3D mesh map (c) *VFP* on 3D mesh map

Fig. 4: Qualitative comparison between (a) the standard 2D ROS *Global Planner* running on a 2.5D DEM `costmap_2d` layer, (b) Dijkstra, and (c) our *Vector Field Planner*, both running on our 3D mesh map shown with the respective scalar distance field u. Then shown environment part has an average elevation of $\sim 14°$, $\sim 2.8\,\mathrm{m}$, while it is highly curved.



(a) scene photo, front view (b) scalar field and path, front view (c) path on colored mesh, top view

Fig. 5: Path planning in a forest with an old quarry in the background. *VFP* computes an accurate and smooth geodetic path along the strongly horizontally curved and inclined forest paths. The path lead around a hill of several meters, from the lower part to the upper part of the quarry.

(a) satellite image (b) wire frame and path (c) distance field and path

Fig. 6: Path planning on the physics campus in Osnabrück from the upper to the lower level over a number of slightly inclined narrow ramps and through a tunnel. Again, our VFP computes a smooth geodesic path, whereas Dijkstra is limited to the mesh topology.

Dataset	Dimensions					Runtime [ms]			Path Length [m]		
	# Vertices	# Triangles	BB x, y, z [m]			VFP	Dijk.	MMP	VFP	Dijk.	MMP
Botanical Garden Osnabrück	719 080	1 430 188	39.05	49.25	6.67	108.1	51.4	1 731.8	26.34	27.47	26.09
Stone Quarry Brockum	992 879	1 904 178	100.58	100.58	23.94	831.0	451.8	7 364.6	110.26	114.67	109.54
Physics Campus Westerberg	719 080	1 617 772	166.02	83.61	26.33	342.9	176.2	1 839.5	200.57	210.15	199.03
Farmer's Pit Stemwede	401 036	794 509	122.23	104.57	14.84	56.2	36.1	1 348.3	54.04	56.02	52.93
Market Garden Ibbenbüren	1 361 308	2 656 283	174.33	149.61	24.58	1 211.0	695.4	6 316.0	95.71	98.62	94.46

TABLE I: Data set specifications and experimental results, where the runtime is compared to the path length for our VFP, Dijkstra and the exact MMP.

impassable areas in an efficient way. In contrast to graph based planning approaches, our algorithm is not restricted to the mesh topology, but has the same optimal asymptotic runtime complexity as *Dijkstra*.

Our algorithm is very fast and needs only around 1.8 times more time than *Dijkstra* to compute shorter and smoother paths. Moreover, the path length is negligibly longer than the path computed by the fastest known exact *MMP* algorithm, which computes the exact shortest geodesic path, However, *MMP* is much slower, it takes around 21.4 times more time that Dijkstra. For comparison, with respect to [32], *Basic S** needs around 3.5 times more time that Dijkstra. In the evaluation, we investigated the performance of our approach which is not bound to any level restrictions in five different sized representative reference datasets, in a forest with a stone quarry, a multilevel environment with a tunnel and ramps, a botanical garden, an agricultural field, and a marked garden. In the related video our *VFP* and *VFC* perform planning and motion control in the forest with a stone quarry in the back, while localization is done with an adapted *LeGO LOAM* [43] approach. Finally, our provided software stack leverages autonomous robot navigation in complex real-world outdoor environments and can be used to reproduce the presented results.

For future work, the wavefront propagation can be parallelized to further optimize the runtime [49]. However, even on the demonstrated large-scale terrains, planning is mostly performed in less than a second, which is sufficient for many real-world operations. Additionally, the goal vector field benefits replanning similar to the grid-based planners *D** or *LPA**. An efficient replanning extension which modifies the vector field to also avoid dynamic obstacles is underway.

REFERENCES

[1] S. Pütz, J. S. Simón, and J. Hertzberg, "Move Base Flex: A Highly Flexible Navigation Framework for Mobile Robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, software available at https://github.com/magazino/move_base_flex.

[2] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[3] J. Bohren and S. Cousins, "The SMACH high-level executive [ROS news]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[4] M. Colledanchise and P. Ögren, "Behavior Trees in Robotics and AI: An Introduction," *arXiv:1709.00084 [cs]*, Jul. 2018.

[5] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, Jun. 1987.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1999, pp. 343–349.

[8] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 3310–3317 vol.4.

[9] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, May 2002, pp. 968–975 vol.1.

[10] A. Nash, S. Koenig, and C. Tovey, "Lazy Theta*: Any-angle path planning and path length analysis in 3D," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*. Citeseer, 2010.

[11] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The Field D* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006.

[12] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *AAAI*, vol. 7, 2007, pp. 1177–1183.

[13] D. D. Harabor and A. Grastien, "An Optimal Any-Angle Pathfinding Algorithm." in *ICAPS*, 2013.

[14] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer International Publishing, 2016, pp. 99–120.

[15] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[16] M. Schwarz and S. Behnke, "Local Navigation in Rough Terrain using Omnidirectional Height," in *Proceedings of the International Symposium on Robotics (ISR)*, 2014.

[17] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, Nov. 2013.

[18] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[19] T. Wiemann, I. Mitschke, A. Mock, and J. Hertzberg, "Surface Reconstruction from Arbitrarily Large Point Clouds," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, Jan. 2018, pp. 278–281.

[20] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Real-time Large Scale Dense RGB-D SLAM with Volumetric Fusion," in *International Journal of Robotics Research Special Issue on Robot Vision*, 2014.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1366–1373.

[22] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847 [cs]*, Jan. 2018.

[23] M. Brandao, O. B. Aladag, and I. Havoutis, "GaitMesh: controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments," p. 8, 2019.

[24] D. Martínez, L. Velho, and P. C. Carvalho, "Computing geodesics on triangular meshes," *Computers & Graphics*, vol. 29, no. 5, pp. 667–675, 2005.

[25] D. S. Yershov and S. M. LaValle, "Simplicial dijkstra and A* algorithms for optimal feedback planning," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 3862–3867.

[26] D. S. Yershov and E. Frazzoli, "Asymptotically optimal feedback planning using a numerical Hamilton-Jacobi-Bellman solver and an adaptive mesh refinement," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 565–584, Apr. 2016.

[27] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," *arXiv:1005.0416 [cs]*, May 2010.

[28] ——, "Sampling-based algorithms for optimal motion planning:," *The International Journal of Robotics Research*, Jun. 2011.

[29] C. Xu, T. Y. Wang, Y.-J. Liu, L. Liu, and Y. He, "Fast Wavefront Propagation (FWP) for Computing Exact Geodesic Distances on Meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 7, pp. 822–834, Jul. 2015.

[30] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The Discrete Geodesic Problem," *SIAM J. Comput.*, vol. 16, no. 4, pp. 647–668, Aug. 1987.

[31] J. Chen and Y. Han, "Shortest paths on a polyhedron," in *Proceedings of the sixth annual symposium on Computational geometry*, ser. SCG '90. New York, NY, USA: Association for Computing Machinery, May 1990, pp. 360–369.

[32] S. Bhattacharya, "Towards optimal path computation in a simplicial complex," *The International Journal of Robotics Research*, vol. 38, no. 8, pp. 981–1009, Jul. 2019.

[33] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms," *SIAM Journal on Numerical Analysis*, vol. 41, no. 1, pp. 325–363, 2003.

[34] J. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," in *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 2, Dec. 1994, pp. 1368–1373 vol.2.

[35] J. V. G. González, "Fast Marching Methods in path and motion planning: improvements and high-level applications," PhD Thesis, Ph. D. Dissertation, Univ. Carlos III de Madrid, 2015.

[36] Q. H. Do, S. Mita, and K. Yoneda, "Narrow passage path planning using fast marching method and support vector machine," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, Jun. 2014, pp. 630–635.

[37] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 2376–2381.

[38] J. Sethian, "Fast Marching Methods," *SIAM Rev.*, vol. 41, no. 2, pp. 199–235, Jan. 1999.

[39] M. Novotni and R. Klein, "Computing Geodesic Distances on Triangular Meshes," *Journal of WSCG*, vol. 10, no. 1-2, pp. 341–347, 2002.

[40] W. Heidrich, "Computing the Barycentric Coordinates of a Projected Point," *Journal of Graphics Tools*, vol. 10, no. 3, pp. 9–12, Jan. 2005.

[41] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3D navigation mesh generation for path planning in uneven terrain," in *Proceedings. IFAC symposium on intelligent autonomous vehicles (IAV-2016), 9th, june 29-July 1, leipzig, germany*. IFAC, 2016.

[42] S. Pütz, T. Wiemann, and J. Hertzberg, "Tools for Visualizing, Annotating and Storing Triangle Meshes in ROS and RViz," in *Proc. 9th European Conference on Mobile Robotics, Prague, Czech Republic*. Prague, Czech Republic: IEEE, Sep. 2019.

[43] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 4758–4765.

[44] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[45] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, Jun. 2015, pp. 5556–5565.

[46] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[47] P. Chopra and J. Meyer, "Topology sensitive volume mesh simplification with planar quadric error metrics," in *IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2003), Benalmadena, Spain*, 2003, pp. 908–913.

[48] T. Wiemann, F. Igelbrink, S. Pütz, and J. Hertzberg, "A file structure and reference data set for high resolution hyperspectral 3D point clouds," in *Proceedings of the IFAC symposium on intelligent autonomous vehicles. IFAC symposium on intelligent autonomous vehicles (IAV-2019), july 3-5, gdansk, poland*. IFAC, 2019.

[49] O. Weber, Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Parallel algorithms for approximation of distance maps on parametric surfaces," *ACM Trans. Graph.*, vol. 27, no. 4, pp. 1–16, Oct. 2008.