Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction

Yue Pan Yves Kompis Luca Bartolomei Ruben Mascaro Cyrill Stachniss Margarita Chli

***Abstract*— Creating accurate maps of complex, unknown environments is of utmost importance for truly autonomous navigation robot. However, building these maps online is far from trivial, especially when dealing with large amounts of raw sensor readings on a computation and energy constrained mobile system, such as a small drone. While numerous approaches tackling this problem have emerged in recent years, the mapping accuracy is often sacrificed as systematic approximation errors are tolerated for efficiency's sake. Motivated by these challenges, we propose *Voxfield*, a mapping framework that can generate maps online with higher accuracy and lower computational burden than the state of the art. Built upon the novel formulation of non-projective truncated signed distance fields (TSDFs), our approach produces more accurate and complete maps, suitable for surface reconstruction. Additionally, it enables efficient generation of Euclidean signed distance fields (ESDFs), useful e.g., for path planning, that does not suffer from typical approximation errors. Through a series of experiments with public datasets, both real-world and synthetic, we demonstrate that our method beats the state of the art in map coverage, accuracy and computational time. Moreover, we show that Voxfield can be utilized as a back-end in recent multi-resolution mapping frameworks, producing high quality maps even in large-scale experiments. Finally, we validate our method by running it onboard a quadrotor, showing it can generate accurate ESDF maps usable for real-time path planning and obstacle avoidance.**

I. Introduction

Generating accurate maps in real-time is vital in order to enable robots to navigate autonomously, especially in unstructured, cluttered environments. While early approaches addressed this problem by constructing occupancy grids [1], where the space is discretized into voxels labeled as either occupied or free, the emergence of trajectory optimization-based path-planning methods [2] rendered these representations as not descriptive enough. For this type of path-planners, getting fast access to richer information about the environment, such as the distance between the robot and the closest nearby obstacle becomes fundamental. However, extracting this kind of information from sensor readings efficiently is far from trivial and becomes even more challenging when the mapping system runs on an embedded platform with limited computational capabilities.

Y. Pan, Y. Kompis, L. Bartolomei, R. Mascaro and M. Chli are with ETH Zurich, Switzerland, Email: {yuepan, ykompis, lbartolomei, rmascaro, chlim}@ethz.ch.

C. Stachniss is with the University of Bonn, Germany. E-mail: cyrill.stachniss@igg.uni-bonn.de.

This work has partially been supported by the Swiss National Science Foundation (SNSF, Agreement no. PP00P2183720), NCCR Robotics, NCCR Digital Fabrication, the HILTI group, the ETH Mobility Initiative and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob.

Video - https://youtu.be/g-mugSb6FJU

Code - https://github.com/VIS4ROB-lab/voxfield

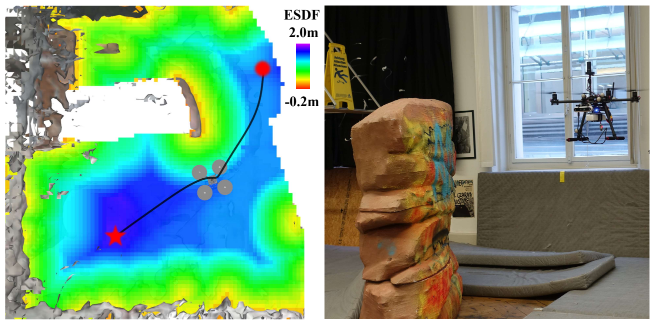


Fig. 1: Real-world mapping and path planning experiment. A Micro-Aerial Vehicle (MAV) is tasked to navigate towards a goal position, while avoiding the obstacle visible on the right. *Voxfield* generates the ESDF map visible on the left onboard the MAV in real time, which is used for planning. The red dot and the star are the starting and goal position, respectively.

Aiming to tackle these issues, in recent years numerous approaches offering a more comprehensive mapping of the environment have been proposed. In particular, methods that model the space as a discrete Euclidean signed distance field (ESDF), i.e. a regular 3D grid where each voxel stores the distance to the closest obstacle, have emerged as well-suited tools for path-planning purposes. Building ESDFs online, in real-time and in dynamically growing maps is, however, quite challenging, as this process is computationally intensive. Although approaches, such as Voxblox [3], FIESTA [4], and EDT [5] have succeeded in meeting the real-time requirement, their accuracy suffers from approximation errors, which are tolerated to ensure the efficiency of the underlying algorithms. This has crucial implications in path planning, where wrongly or poorly estimated distances might lead to obstacles being closer to the robot than reported by the map. These effects are aggravated at lower map resolutions, as the lack of detail can potentially lead to collisions between the robot and finer structures not being captured by the map. On the other hand, higher resolutions imply higher computational and memory costs, slowing down the mapping pipeline and rendering it unsuitable for real-time deployment. To achieve a better balance between efficiency and accuracy, multi-resolution panoptic mapping systems using semantic segmentation have recently been proposed [6], opening up exciting prospects for large-scale mapping use-cases. These methods can represent different parts of the environment with different voxel resolutions, capturing finer details only where needed (e.g. relatively small objects) and reducing the overall memory footprint. However, these multi-resolution approaches are generally built on top of well-established, single-resolution systems, therefore suffering from the same limitations and approximation errors.

Motivated by these challenges, in this work we propose a new mapping framework, dubbed *Voxfield*, that is able to tackle all the aforementioned issues by efficiently building accurate ESDFs from *non-projective* TSDFs. Given as input point clouds from RGB-D cameras or LiDARs, as well as the associated sensor poses, our system is able to reconstruct large-scale scenes with high accuracy in real time. We evaluate our method against the state of the art in synthetic and real-world datasets, demonstrating that, using non-projective TSDFs, our approach reaches better scene coverage, as well as higher ESDF accuracy with lower computational times. Moreover, we show that our framework can be used as the mapping back-end of an existing multi-resolution panoptic mapping pipeline [6], producing higher quality maps than the state of the art. Finally, in a real-world experiment (Fig. 1), we demonstrate that Voxfield runs onboard a Micro-Aerial Vehicle (MAV), generating ESDF maps that can be used by a path planner to perform obstacle avoidance in real time.

In short, the main contributions of this work are:

- an accurate TSDF integration method creating improved scene coverage based on non-projective distances,
- an ESDF update algorithm with sub-voxel accuracy and higher efficiency than the state of the art,
- an extensive evaluation of the accuracy and efficiency, the integration into a state-of-the-art multi-resolution panoptic mapping framework, as well as real-world path-planning experiments.

II. Related work

Robotic mapping, i.e. the problem of acquiring a spatial model of a robot's environment, has been a very active research area for the past decades. While different frameworks and underlying data structures have been proposed over the years, few of them have been proven capable of achieving high-fidelity 3D scene reconstructions, and at the same time being suitable for real-time path planning.

Dense mapping approaches, which discretize the space into a regularly sampled voxel grid that implicitly models 3D geometry, are arguably among the most mature, widely used techniques in the field. Contrary to point cloud, surfel or sparse feature representations traditionally employed for localization, dense approaches have the ability to represent continuous surfaces and distinguish between free and unknown space, which is usually a requirement for autonomous navigation. In this context, two of the most representative and successful map representations to date are occupancy grids [1] and TSDFs [7]. The former label voxels as either occupied or free, whereas the latter store at each voxel the signed distance to the nearest surface, up to a truncation radius around the surface boundary. Although both approaches allow for efficient 3D reconstruction into surface meshes, e.g. using the marching cubes algorithm [8], TSDFs can achieve sub-voxel resolutions, which makes them the preferred choice in cases where accurate surface reconstructions are needed. Given these benefits and their efficacy in fusing high-rate depth measurements while smoothing out sensor noise, as first shown by KinectFusion [7], TSDFs have increasingly become the representation of choice in several mapping frameworks. Two relevant examples are Chisel [9] and Voxblox [3], which are specifically designed for deployment on robots or devices with limited computing resources. More recently, TSDFs have continued being adopted as underlying representations in approaches targeting large-scale outdoor reconstruction [10] or exploring the incorporation of higher-level information, such as semantic or instance labels, into purely geometric maps [6], [11]–[14].

When it comes to path planning, however, ESDF maps are more informative than occupancy grids or TSDFs as they allow for efficient queries of distances and gradients to obstacles at any point in space. This is of particular interest in trajectory optimization-based planning algorithms, such as CHOMP [2], for which occupancy information alone is not sufficient. The main challenge arising from maintaining such a map representation during online operation is the need for updating the ESDF in an efficient and accurate manner. While different methods have been proposed to generate ESDFs in real-time on a dynamically growing map, just a few can run on embedded CPU-only platforms. For example, Voxblox [3] progressively builds an ESDF map from a TSDF that integrates the incoming sensor data. Other approaches, such as FIESTA [4] and EDT [5] propose specialized data structures for fast incremental ESDF updates from occupancy maps. The main drawback of all the aforementioned methods is that they can only achieve sub-optimal accuracy. In the case of Voxblox, errors come from two sources: first, its underlying TSDF representation employs projective distances, i.e. the distance along sensor rays to the measured surface, thus being prone to overestimate the actual Euclidean distance to the nearest surface; and second, it updates the ESDF according to quasi-Euclidean distances. FIESTA and EDT, on the other hand, compute the true Euclidean distance but suffer from discretization errors derived from the fact that this metric is computed between voxel centers.

Similarly to Voxblox [3], in this work we advocate for building ESDFs from an underlying TSDF-based map representation that is well-suited for surface reconstruction. However, we build our approach on top of a novel non-projective TSDF formulation that reaches better scene coverage and results in higher TSDF and ESDF accuracy when compared to previous approaches [3]–[5], while being computationally faster. In addition, we demonstrate that the presented framework can be seamlessly integrated into recently developed multi-resolution, panoptic mapping frameworks [6], effectively contributing to increase the quality of the generated maps. Finally, we show that Voxfield can run online onboard an MAV, generating maps that can be used for path planning.

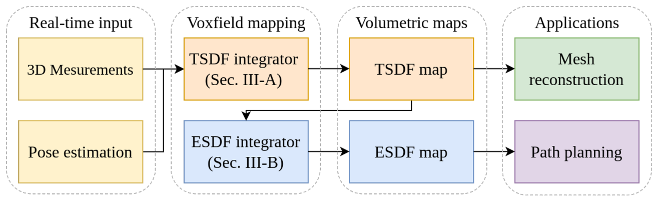


Fig. 2: Overview of the proposed Voxfield mapping framework. Sensor measurements with associated poses are integrated into a TSDF map, suitable e.g. for mesh reconstruction, and subsequently into a full ESDF map, used for path planning.

III. Methodology

Voxfield processes incoming 3D measurements with known associated poses, estimated e.g. by a Simultaneous Localization And Mapping (SLAM) system, and incrementally builds a dense voxel map that can be used for online path planning and 3D reconstruction. As shown in the system overview in Fig. 2, raw 3D measurements are first integrated into a non-projective TSDF map (Section III-A), which serves as a base for generating a full ESDF map (Section III-B). The system also outputs a 3D mesh of the scanned environment, which is generated from the TSDF using the marching cubes algorithm [8] as in [3].

A. Non-projective TSDF Mapping

The TSDF map representation employed by Voxfield consists of a set of spatially hashed voxels V_i with a common voxel size $\nu \in \mathbb{R}^+$. Every voxel V_i is identified by a global index $\mathbf{v}_i \in \mathbb{Z}^3$, from which the coordinates of its center can be retrieved, i.e. $\mathbf{x}_i = \nu\mathbf{v}_i \in \mathbb{R}^3$, and additionally stores a truncated signed distance D_i, a weight W_i representing the confidence in D_i, and the normalized gradient of the signed distance $\mathbf{g}_i \in \mathbb{R}^3$.

At every frame k, the proposed non-projective TSDF integration algorithm takes as input the sensor 3D point cloud $\{\mathbf{p}\}_k \in \mathbb{R}^3$ with point-wise normals $\{\mathbf{n}\}_k \in \mathbb{R}^3$, which can be efficiently estimated using the cross product over the depth map as described in [7], together with the sensor's pose $\mathbf{T}_{wk} \in \mathrm{SE}(3)$ in the world frame w.

To update the volumetric TSDF map, we first cast a ray from the sensor origin $\mathbf{s}_k \in \mathbb{R}^3$ to every point $\mathbf{p}_j \in \{\mathbf{p}\}_k$ and compute the projective signed distance at every voxel in the map along this ray:

$$\psi_{ijk} = \mathrm{sign}\left((\mathbf{p}_j - \mathbf{s}_k) \cdot (\mathbf{p}_j - \mathbf{x}_i)\right) \|\mathbf{p}_j - \mathbf{x}_i\|. \tag{1}$$

The integration of a new measurement into the existing TSDF map is governed by a weighting function that, similarly to [3], we model as follows:

$$w_{ijk} = w_{jk}^{(\text{sensor})} w_{ijk}^{(\text{dropoff})}. \tag{2}$$

The first term, $w_{jk}^{(\text{sensor})}$, depends on the sensor's error model:

$$w_{jk}^{(\text{sensor})} = \frac{1}{\|\mathbf{p}_j - \mathbf{s}_k\|^m}, \tag{3}$$

with the exponent m representing the noise characteristics of the sensor. In our implementation, we use $m = 2$ or $m = 1$ for RGB-D and LiDAR input, respectively. The second term, $w_{ijk}^{(\text{dropoff})}$, represents a behind-the-surface drop-off that is used to improve the reconstruction quality of thin surfaces. Formally, it is defined as

$$w_{ijk}^{(\text{dropoff})} = \begin{cases} 1 & \psi_{ijk} \geq -\epsilon \\ \dfrac{\tau + \psi_{ijk}}{\tau - \epsilon} & -\tau < \psi_{ijk} < -\epsilon \\ 0 & \psi_{ijk} \leq -\tau \end{cases}, \tag{4}$$

where τ is the truncation distance and ϵ is the drop-off distance. A common choice for these parameters is $\tau = 3\nu$ and $\epsilon = \nu$.

In order to derive the non-projective signed distance d_{ijk} of voxel V_i from the previously computed signed distance ψ_{ijk}, we first update the voxel's gradient $\mathbf{g}_i$ according to the normal vector $\mathbf{n}_j$ at point $\mathbf{p}_j$:

$$\mathbf{g}_i \leftarrow \frac{\tilde{\mathbf{g}}_i}{\|\tilde{\mathbf{g}}_i\|}, \quad \text{with} \quad \tilde{\mathbf{g}}_i = \frac{W_i\mathbf{g}_i + w_{ijk}\mathbf{n}_j}{W_i + w_{ijk}}. \tag{5}$$

This results in a precise estimate of the gradient direction in the trivial case of reconstructing a flat surface, as seen in Fig. 3b. In the more general case of a curved surface, as shown in Fig. 3a, we rely on the movement of the sensor to estimate the gradient accurately over time, even with non-optimal normal estimations from each single measurement.

Given the estimated gradient, the non-projective signed distance d_{ijk} for voxel V_i from the measurement p_j at time k is then computed as:

$$d_{ijk} = \begin{cases} |\cos\theta|\, \psi_{ijk} & \text{if } \alpha = 0 \\ \left|\dfrac{(\cos\alpha - 1)\sin\theta}{\sin\alpha} + \cos\theta\right| \psi_{ijk} & \text{otherwise} \end{cases}, \tag{6}$$

where θ represents the angle between the ray and the gradient

$$\cos\theta = \frac{\mathbf{p}_j \cdot \mathbf{g}_i}{\|\mathbf{p}_j\|}, \tag{7}$$

and α represents the angle between the gradient and the current surface normal

$$\cos\alpha = \mathbf{n}_j \cdot \mathbf{g}_i. \tag{8}$$

The geometric relationship between the non-projective signed distance d_{ijk} and the projective signed distance ψ_{ijk} for flat and curved surfaces are shown in Fig. 3b and Fig. 3c, respectively.

Finally, the non-projective truncated signed distance D_i and the weight W_i can be updated as follows:

$$D_i \leftarrow \frac{W_i D_i + w_{ijk}\mathrm{trunc}\left(d_{ijk}, \tau\right)}{W_i + w_{ijk}}, \text{ and} \tag{9}$$

$$W_i \leftarrow \min\left(W_i + w_{ijk}, W_{\max}\right), \tag{10}$$

where the weight is bounded to $W_{\max}$ and the signed distance is truncated at τ

$$\mathrm{trunc}(d, \tau) = \begin{cases} \mathrm{sign}\,(d)\,\tau & \text{if } |d| > \tau \\ d & \text{otherwise} \end{cases}. \tag{11}$$

B. Voxfield ESDF Mapping

Voxfield's ESDF map shares the general characteristics of the TSDF map, such as the voxel size ν and the indices $\mathbf{v}_i$ of voxel V_i^E. Every ESDF voxel V_i^E stores the signed Euclidean distance to the closest obstacle D_i^E, the index of the closest occupied voxel $\mathbf{v}_i^{co}$, as well as a set of pointers to other voxels used for efficient map updates, explained below.

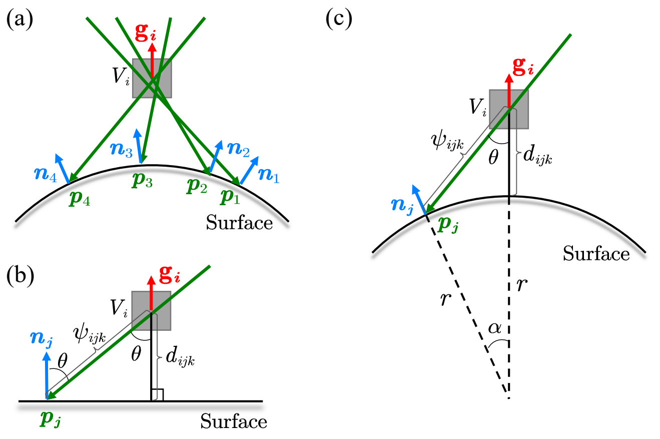


Fig. 3: Non-projective TSDF calculations for voxel V_i: the updated voxel is represented in grey, rays from raycasting in green, the surface normals in blue and the field gradient in red. (a) shows the integration of voxel V_i's TSDF-gradient by taking a weighted average of the normals on the surface point hit by the rays cast through V_i from different sensor positions. (b) and (c) show the geometric relationship between the projective distance ψ_{ijk} and the non-projective distance d_{ijk} under the assumption of a flat or curved surface, respectively. The radius of the curved surface in (c) is marked as r.

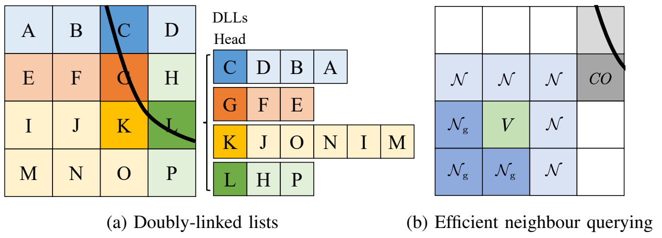


Fig. 4: Diagram of Voxfield's ESDF algorithm simplified in 2D: (a) the black line represents the surface, the saturated voxels (C, G, K, L) are occupied while the other voxels are free or behind the surface. Each of the doubly-linked lists has an occupied voxel as the head node. The other nodes in the list are the voxels where the head node is the closest occupied voxel. (b) the voxels in grey are occupied. We are querying the neighbouring voxels $\mathcal{N}$ of the voxel V. The subset of neighbouring voxels affected by a change of V, namely voxels $\mathcal{N}_g$, can be identified from the neighbourhood based on the direction of the closest occupied voxel CO of V.

The ESDF map is updated incrementally from the TSDF map, as listed in Algorithm 1. The core of the update step is a Breadth-First Search (BFS), starting from all occupied voxels and expanding into free space. This expansion continues until we hit the map boundary or the maximum ESDF integration distance d_{max}, updating the ESDF value of all voxels we traverse. However, an exhaustive search is inefficient, and does not exploit the incremental nature of map updates. Under the simplifying assumption of a noiseless sensor in a static environment, the number of updated voxels during BFS can be drastically reduced. In this case, where we only *add* obstacles and never *remove* them, any change to the value of an ESDF voxel will be a *decrease* of the Euclidean distance. We can exploit this fact, by only updating a voxel, if the newly calculated Euclidean distance is in fact lower than the currently stored distance (Line 30). By starting the BFS at newly occupied voxels (Line 8), instead of all occupied voxels, we effectively limit the scope of the BFS to regions of the map with actual changes in the ESDF.

This approach, e.g. used by [3] and [4], allows for real-time ESDF updates for sufficiently small voxel resolutions. Nonetheless, the efficiency of this step can be further improved by checking the position of the voxel subjected to update with respect to its closest occupied voxel (Line 29). As shown in Fig. 4b, it is sufficient to check the neighbors on the far side of the closest occupied voxel, while the majority of neighboring cells, lying between the current voxel and the closest occupied position, can be safely ignored.

The second fundamental step in the ESDF update is the removal of obstacles that are no longer present in the map, either due to sensor noise or a non-static environment. In this case, the affected voxels change their status from occupied to free. In contrast to the previous case, this update *increases* the value of affected voxels, as they are farther away from occupied positions. This case breaks the BFS strategy, since it can only decrease voxel values when updating the map incrementally. A simple solution is to reset all affected voxel values to d_{max} (Line 13). Naïvely, this can be done in an exhaustive manner as in [3]. Instead, we use an efficient bookkeeping strategy using Doubly-Linked Lists (DLLs) to deal with increased voxel values, similar to [4]. Every voxel in the list stores pointers to the previous and next element in the DLL, and every occupied voxel is the head of one DLL. As shown in Fig. 4a, the DLL stores all voxels that are to be *increased* when the head is no longer occupied. Using this data structure, we can efficiently traverse all voxels affected by a newly free voxel (Line 14), and reset their ESDF value to d_{max}. Following this strategy, the BFS step (Line 26) can properly update these voxels to their new, higher ESDF value.

Finally, when calculating a new ESDF value during the BFS update, we either copy the non-projective TSDF value from the source map, or we calculate the true Euclidean distance D_i^E from the voxel to the closest surface

$$D_i^E = \begin{cases} \text{sign}(D_i)\|\mathbf{x}_i - (\mathbf{x}_i^{co} - D_i^{co}\mathbf{g}_i^{co})\| & \text{if } |D_i| > \tau \\ D_i & \text{otherwise} \end{cases}, \tag{12}$$

where D_i is the truncated signed distance, and the superscript co refers to the closest occupied voxel. Voxfield makes use of the TSDF gradient and distance value to calculate the true distance from the voxel's center to the actual surface (Fig. 5a). Therefore, the distance values computed are more accurate than in other state-of-the-art approaches. In fact, while Voxblox approximates the ESDF value by a broken-line quasi-Euclidean sum of distances (Fig. 5c), FIESTA calculates the true Euclidean distance between the voxel and its closest occupied voxel's center (Fig. 5b).

IV. Experiments

The proposed Voxfield mapping framework is evaluated qualitatively and quantitatively on four publicly available datasets featuring synthetic and real-world scans of indoor and outdoor scenes. The main characteristics of these datasets, collected by either RGB-D cameras or LiDAR sensors, are summarized in Table I.

To further demonstrate its potential, we integrate Voxfield into a multi-resolution panoptic mapping system [6] and show its applicability in a large-scale mapping scenario,

Algorithm 1 Voxfield ESDF map update

```
Input: Updated TSDF map $M_T(k)$ with voxels $\{V_T\}$, previous ESDF map
    $M_E(k-1)$ and doubly-linked lists $\{dll(\mathbf{v})\}$
Output: Updated ESDF map $M_E(k)$ with voxels $\{V_E\}$
1:  for each $\mathbf{v}$ in GETUPDATEDVOXELINDICES($M_T$)
2:      if not $V_E(\mathbf{v}).occupied$ and ISOCCUPIED($V_T(\mathbf{v})$)
3:          insertList.push($\mathbf{v}$)
4:      if $V_E(\mathbf{v}).occupied$ and not ISOCCUPIED($V_T(\mathbf{v})$)
5:          deleteList.push($\mathbf{v}$)
6:      $V_E(\mathbf{v}).occupied \leftarrow$ ISOCCUPIED($V_T(\mathbf{v})$)
7:      $V_E(\mathbf{v}).fixed \leftarrow (V_T(\mathbf{v}).d \leq \tau)$
8:  for each $\mathbf{v}$ in insertList
9:      $V_E(\mathbf{v}).d \leftarrow 0$
10:     $V_E(\mathbf{v}).co \leftarrow \mathbf{v}$
11:     dll($\mathbf{v}$).insert($\mathbf{v}$)
12:     updateQueue.push($\mathbf{v}$, 0)
13: for each $\mathbf{v}$ in deleteList
14:     for each $\mathbf{u}$ in dll($\mathbf{v}$)
15:         dll($\mathbf{v}$).delete($\mathbf{u}$)
16:         $V_E(\mathbf{u}).d \leftarrow d_{max}$
17:         $V_E(\mathbf{u}).co \leftarrow$ null
18:         for each $\mathbf{n}$ in GETALLNEIGHBOURS($\mathbf{u}$)
19:             $\mathbf{m} \leftarrow V_E(\mathbf{n}).co$
20:             if $V_E(\mathbf{m}).occupied$ and DISTANCE($\mathbf{m}$, $\mathbf{u}$) $< V_E(\mathbf{u}).d$
21:                 $V_E(\mathbf{u}).d \leftarrow$ DISTANCE($\mathbf{m}$, $\mathbf{u}$)
22:                 $V_E(\mathbf{u}).co \leftarrow \mathbf{m}$
23:         if $V_E(\mathbf{u}).co \neq$ null
24:             dll($V_E(\mathbf{u}).co$).insert($\mathbf{u}$)
25:             updateQueue.push($\mathbf{u}$, $V_E(\mathbf{u}).d$)
26: while updateQueue $\neq \emptyset$
27:     $\mathbf{v} \leftarrow$ updateQueue.front()
28:     updateQueue.pop()
29:     for each $\mathbf{n}$ in GETAFFECTEDNEIGHBOURS($\mathbf{v}$, $V_E(\mathbf{v}).co$)
30:         if DISTANCE($V_E(\mathbf{v}).co$, $\mathbf{n}$) $< V_E(\mathbf{n}).d$
31:             dll($V_E(\mathbf{n}).co$).delete($\mathbf{n}$)
32:             $V_E(\mathbf{n}).d \leftarrow$ DISTANCE($V_E(\mathbf{v}).co$, $\mathbf{n}$)
33:             $V_E(\mathbf{n}).co \leftarrow V_E(\mathbf{v}).co$
34:             dll($V_E(\mathbf{n}).co$).insert($\mathbf{n}$)
35:             updateQueue.push($\mathbf{n}$, $V_E(\mathbf{n}).d$)
36:     if $V_E(\mathbf{v}).fixed$
37:         $V_E(\mathbf{v}).\tilde{d} \leftarrow V_T(\mathbf{v}).d$
38:     else
39:         $V_E(\mathbf{v}).\tilde{d} \leftarrow$ EXACTESDFVALUE($\mathbf{v}$, $V_E(\mathbf{v}).co$)
```

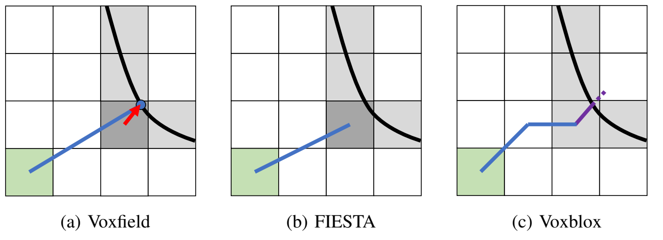


Fig. 5: ESDF calculation of different methods in the 2D case. The black line represents the surface, the voxels in grey are occupied, while those in white are free or behind the surface. The Euclidean distance is calculated for the voxel in green. The darker grey voxel is the green voxel's closest occupied voxel. The red arrow represents the gradient of the closest occupied voxel scaled by the TSDF value of the voxel. The ESDF is calculated as (a) the true Euclidean distance (in blue) between the voxel center and the surface point (in blue), (b) the true Euclidean distance (in blue) between voxel centers, (c) the quasi-Euclidean sum of broken lines (in blue) and the over-estimated projective TSDF value (in purple).

where the accuracy-efficiency trade-off plays a major role. Finally, we conduct a real-world experiment, where an MAV is tasked to perform online obstacle avoidance, proving the usefulness of the ESDF map produced by Voxfield on-the-fly.

A. TSDF Mapping Evaluation

We evaluate the quality of the proposed non-projective TSDF mapping approach based on three metrics that are calculated with respect to the ground-truth point cloud: the TSDF error, the reconstruction Chamfer distance, and the reconstruction coverage.

1) TSDF Error: For each point in the ground-truth point cloud, the TSDF error is regarded as the distance that results from projecting the point into the TSDF using trilinear interpolation. The overall TSDF error is, therefore, defined as the average error over the ground-truth point cloud.

2) Chamfer Distance: As a common metric for reconstruction quality, the Chamfer-L1 distance [17] is calculated between the vertices of the reconstructed mesh $\mathcal{M}$ and the ground-truth point cloud $\mathcal{G}$ as:

$$d_{\text{CD}}(\mathcal{M},\mathcal{G}) = \frac{1}{2|\mathcal{M}|}\sum_{\mathbf{m}\in\mathcal{M}}\min_{\mathbf{g}\in\mathcal{G}}\|\mathbf{m}-\mathbf{g}\| + \frac{1}{2|\mathcal{G}|}\sum_{\mathbf{g}\in\mathcal{G}}\min_{\mathbf{m}\in\mathcal{M}}\|\mathbf{g}-\mathbf{m}\| \,. \tag{13}$$

3) Reconstruction Coverage: The reconstruction coverage is defined as the ratio between the number of ground-truth points that do have a mesh vertex nearby ($\leq 2\nu$) and the total number of ground-truth points.

The evaluation results on all four datasets with different voxel size settings are shown in Fig. 6. We also report the performance scores obtained by running the same experiments with the projective-TSDF integrator proposed in Voxblox [3], which is regarded as the baseline for comparison. Our non-projective TSDF map representation achieves a lower TSDF error for all tested voxel sizes, with a more notable difference on the LiDAR datasets, where large incidence angles are more common. The mesh reconstruction quality, as indicated by the Chamfer distance, is on par with Voxblox. The reason behind this is that the marching cubes algorithm used for mesh reconstruction mainly depends on the ratio of TSDF values of neighboring voxels at zero-crossings of the field, which hardly changes between the compared TSDF integration approaches. Nonetheless, our method achieves higher coverage on all four datasets, as is also visible in the qualitative results shown in Fig. 7. This effect is especially noticeable at the boundaries of the scans and results from the fact that the projective TSDF mapping from [3] generally overestimates the signed distance value, leading to more truncated voxels than under the non-projective TSDF map built by Voxfield.

B. ESDF Mapping Evaluation

The proposed ESDF mapping algorithm is evaluated in terms of both accuracy and efficiency using the ESDF error and ESDF update time, respectively.

TABLE I: Four public datasets used for the experiments

Dataset	Type	Sensor	Pose	Ground truth Point Cloud
Flat [6]	synthetic	RGB-D	Ground Truth	CAD model sampling
Cow&Lady [3]	real-world	RGB-D	Vicon	TPS-MS50 laser scan
MaiCity [15]	synthetic	LiDAR	Ground Truth	CAD model sampling
KITTI [16]	real-world	LiDAR	GNSS-INS	Scan accumulation by pose

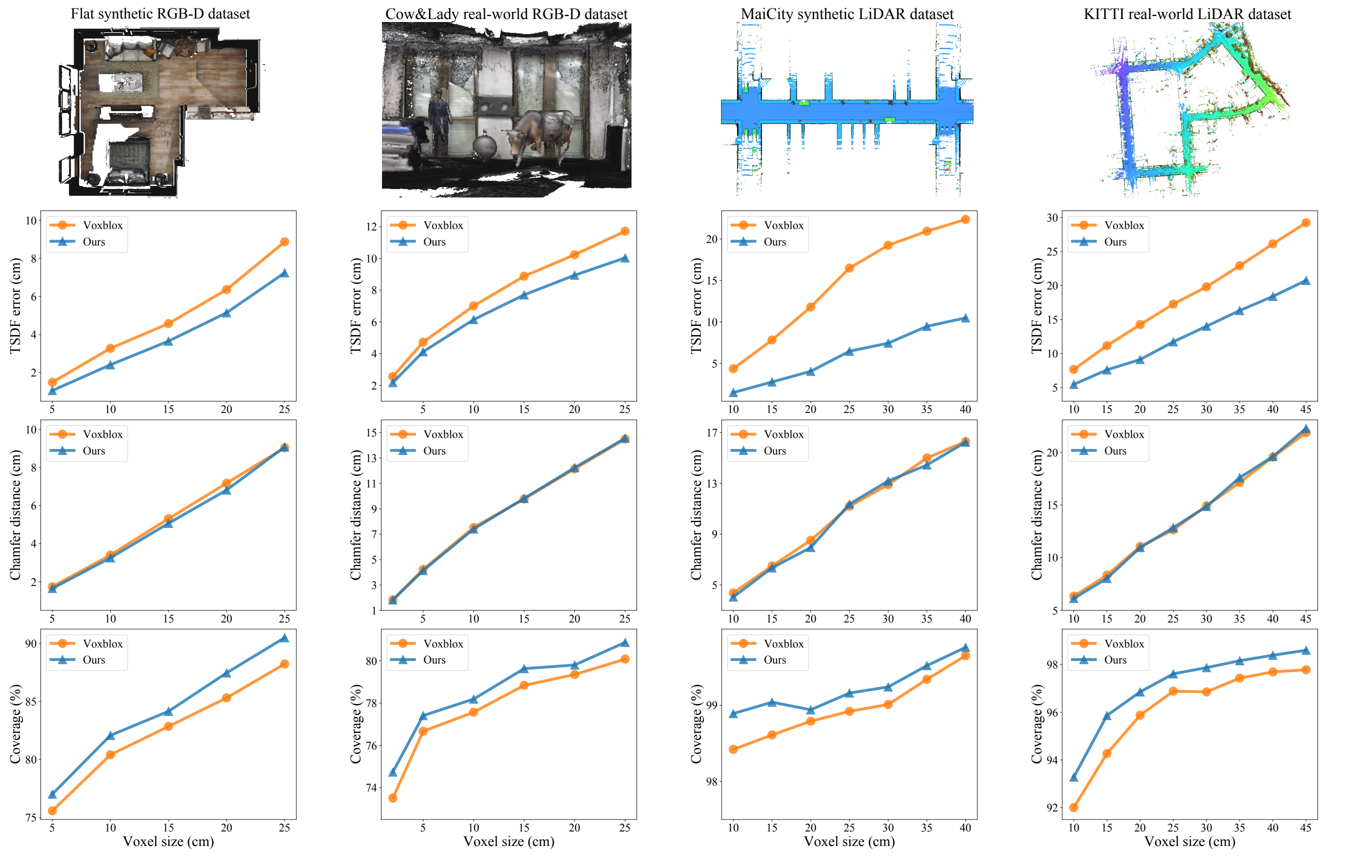
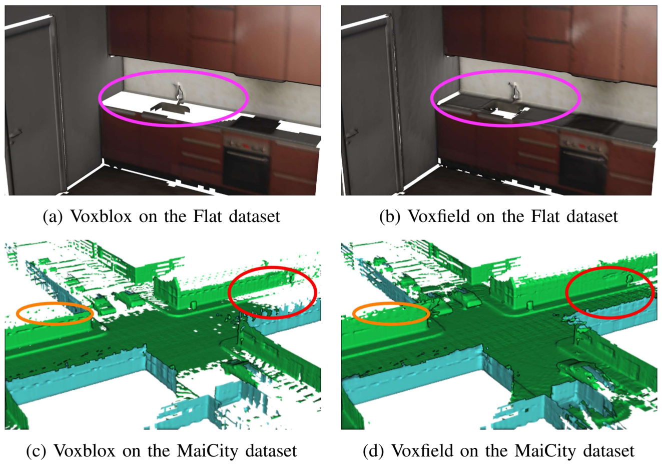


Fig. 6: A comparison of methods for TSDF mapping on each dataset from Table I with different voxel sizes. The proposed Voxfield is compared against Voxblox on the TSDF error, reconstruction Chamfer distance and the reconstruction coverage.



(a) Voxblox on the Flat dataset (b) Voxfield on the Flat dataset

(c) Voxblox on the MaiCity dataset (d) Voxfield on the MaiCity dataset

Fig. 7: Qualitative comparison of the 3D reconstruction on the Flat RGB-D dataset and the MaiCity LiDAR dataset. Voxfield has a more complete reconstruction, especially in the regions highlighted by the ellipsoids.

1) ESDF Error: For each voxel in the ESDF map, the ground-truth signed distance is calculated as the Euclidean distance from the voxel's center to the nearest point in the ground-truth point cloud. The voxel's ESDF error is then defined as the difference between the ground-truth value and the mapped ESDF value, and the overall ESDF error is computed as the average of all ESDF voxel errors. Compared with the ESDF error metric in [4], which is defined with respect to the centers of the occupied voxels, the ESDF error metric used here is not affected by the grid discretization and is, therefore, more accurate.

2) ESDF Update Time: The ESDF update time is the average time used for one incremental update of the ESDF map, excluding any time for TSDF or occupancy mapping.

The proposed ESDF update algorithm is evaluated against Voxblox [3], FIESTA [4] and EDT [5] on all four datasets under various voxel size settings. To establish a fair comparison, we adapt these three state-of-the-art methods with the same TSDF integration front-end as Voxblox. In the cases of FIESTA and EDT, where the ESDF is built from an occupancy map, the occupancy status is inferred from the projective TSDF map.

As shown in Fig. 8, our approach outperforms the state-of-the-art methods on both ESDF accuracy and computation time. Results demonstrate that, by calculating the true Euclidean distance based on a non-projective TSDF, Voxfield's ESDF error is almost cut in half compared to Voxblox. In addition, Voxfield also improves the ESDF accuracy by more than 15% on average when compared to FIESTA and EDT, which are based on an occupancy map with true Euclidean distance calculations between voxel centers. A qualitative comparison of the ESDF mapping error obtained with the four evaluated methods is shown in Fig. 9.

Regarding the ESDF update efficiency, Voxfield runs about three times faster than Voxblox due to the usage of DLLs and is about 15% faster than FIESTA and EDT because of

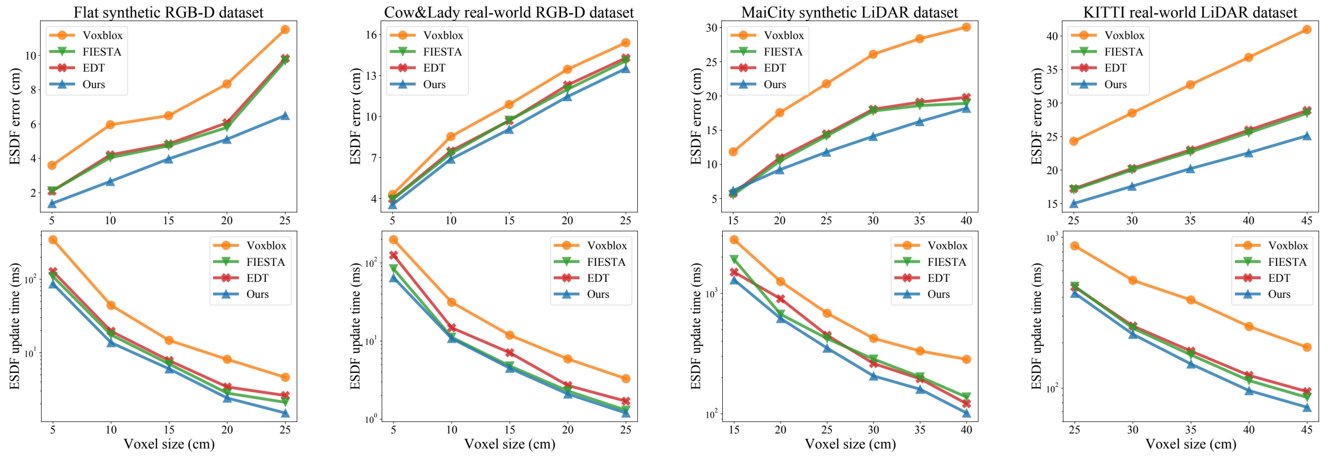
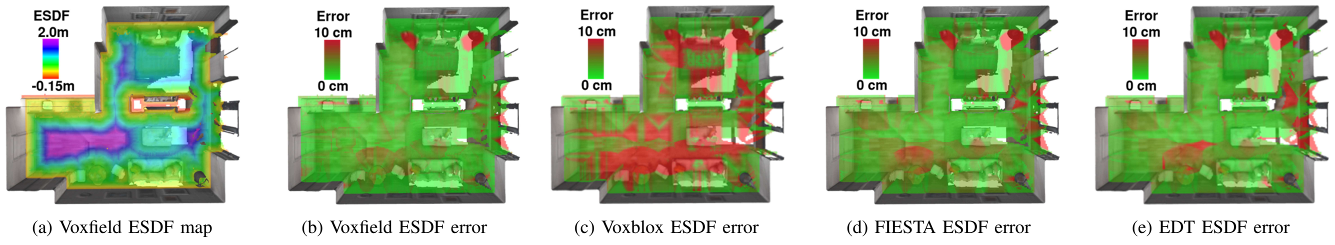


Fig. 8: A comparison of methods for ESDF mapping on each dataset from Table I with different voxel sizes. The proposed Voxfield is compared against Voxblox, FIESTA and EDT on the ESDF error and the ESDF update time.



(a) Voxfield ESDF map (b) Voxfield ESDF error (c) Voxblox ESDF error (d) FIESTA ESDF error (e) EDT ESDF error

Fig. 9: Qualitative results of ESDF mapping on the synthetic Flat dataset with a voxel size of 5 cm. Voxfield achieves the most accurate ESDF map.

the proposed directional neighbor search.

C. Multi-resolution Mapping Evaluation

Aiming at further exploiting the benefits of the proposed approach, we integrate Voxfield into a multi-resolution *panoptic* (i.e. containing instance- and semantic-level information) mapping framework [6], which we refer to as Panmap in this work. The system builds a global map composed of multiple TSDF submaps, each representing a single object instance with the voxel size determined by its semantic class. This allows certain object classes to be reconstructed at a small voxel size and with high accuracy, while using larger voxel sizes to efficiently map the background or less relevant classes. We replace the mapping back-end of [6] with Voxfield and complement the TSDF submaps with a free-space ESDF map for path planning. In addition, we extend the data pre-processing step, originally developed for RGB-D sensors, to enable the panoptic mapping pipeline to process LiDAR input.

We demonstrate the advantages of using Voxfield as a mapping back-end by conducting experiments on the KITTI dataset, where ground-truth 3D panoptic labels are available [18]. To test the system under a more realistic setup, we also take as input labels predicted by a neural network. For this purpose, we use RangeNet++ [19] combined with a depth-based clustering [20], which is light enough for real-time performance e.g. on a mobile GPU.

We compare against the original multi-resolution Panmap [6], as well as the single-resolution mapping systems C-blox [21] and Voxblox [3]. Fig. 10 shows the reconstruction Chamfer distance versus integration time per frame, excluding ESDF mapping, under varying voxel sizes. Our multi-resolution Panmap with Voxfield as the back-end, even with non-optimal neural network predictions, can achieve better accuracy using less computational time than the single-resolution mapping systems Voxblox and C-blox. The original Panmap implementation is slightly faster for a given set of voxel sizes, since it does not calculate normals nor performs the non-projective distance correction, but our implementation achieves better reconstruction accuracy and coverage, as shown in Fig. 11.

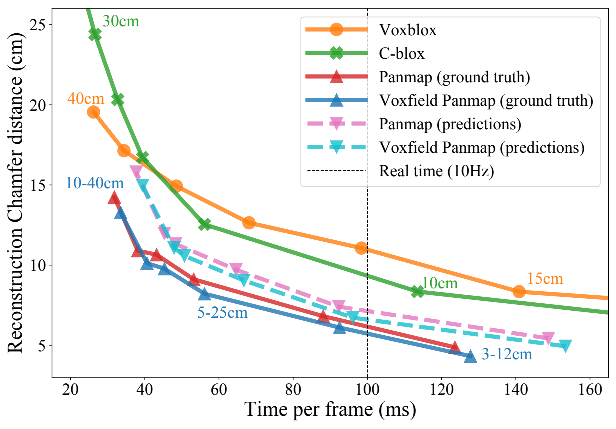
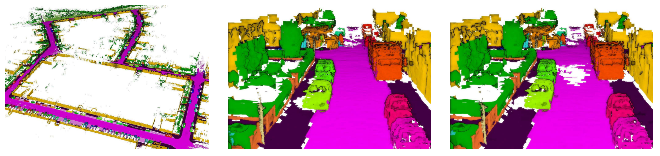


Fig. 10: Reconstruction error (Chamfer distance) versus update time for different voxel sizes on the KITTI dataset. Each marker represents the result of one run. For the single-resolution mapping systems Voxblox and C-blox, we test voxel sizes ranging from 5 cm to 40 cm with an interval of 5 cm. For the multi-resolution Panmap with ground-truth labels or neural network label predictions, with or without Voxfield as its back-end, we test the same six sets of voxel sizes from 3 cm–12 cm up to 10 cm–40 cm. The bottom left corner on the plot means an overall better performance.



(a) Large-scale reconstruction by Voxfield Panmap (b) A close-up of Voxfield Panmap's reconstruction (c) A close-up of Panmap's reconstruction

Fig. 11: Qualitative results of the multi-resolution panoptic mapping reconstruction on the KITTI dataset. Ground-truth semantic labels and voxel sizes in the range of $5\,\mathrm{cm}$–$25\,\mathrm{cm}$ are used.

D. Real-world Path-Planning Experiment

To verify Voxfield's usefulness as a mapping module for real-world path-planning applications, we conduct an experiment with an MAV equipped with an Intel Core i5-1145G7 CPU and a RealSense D455 depth sensor. State estimation [22], TSDF and ESDF mapping with Voxfield and path planning [23] all run onboard the MAV. As shown in Fig. 1, the robot is able to plan a path and safely navigate to a goal position inside a previously unknown environment, with the ESDF mapping conducted on-the-fly. Voxfield spends $1.9\,\mathrm{ms}$ and $22.7\,\mathrm{ms}$ on average for each ESDF map update with a $20\,\mathrm{cm}$ and $10\,\mathrm{cm}$ voxel size, respectively, indicating that it can achieve real-time performance onboard the MAV.

V. Conclusion

In this work, we present Voxfield, a fast and accurate mapping framework for online 3D reconstruction and path planning. Thanks to the use of non-projective TSDFs, the proposed approach can create accurate reconstructions with higher scene coverage than state-of-the-art methods. Our efficient, gradient-guided ESDF update method allows robots to calculate ESDF values with sub-voxel accuracy, while reducing the overall computational burden. In our evaluation, we show that Voxfield's non-projective TSDF can reduce the average error by 32% compared to the state of the art, while at the same time increasing coverage by 1% on average. Similarly, the ESDF accuracy is improved on average by 24% while reducing the integration time by 42% on average. Furthermore, we incorporate Voxfield into a multi-resolution panoptic mapping framework, improving coverage and reconstruction quality over the state of the art in large-scale scenes. Finally, in a real-world experiment, we demonstrate that Voxfield can create accurate maps in real time when running onboard an MAV, enabling the platform to perform online obstacle avoidance.

Future work will explore extending Voxfield with a pose-graph optimization back-end to be resilient to noisy input poses and to improve the fidelity of the mapping process.

References

[1] 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, 2013.

[2] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, 2013.

[3] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board MAV planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[4] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[5] D. Zhu, C. Wang, W. Wang, R. Garg, S. Scherer, and M. Q.-H. Meng, "VDB-EDT: An Efficient Euclidean Distance Transform Algorithm Based on VDB Data Structure," *arXiv*, 2021.

[6] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena, "Panoptic Multi-TSDFs: a Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[7] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[8] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM Siggraph Computer Graphics*, vol. 21, no. 4, 1987.

[9] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao, "Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields," in *Robotics: Science and Systems (RSS)*, 2015.

[10] T. Kühner and J. Kümmerle, "Large-scale volumetric scene reconstruction using LiDAR," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[11] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3D object discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, 2019.

[12] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "PanopticFusion: Online volumetric semantic mapping at the level of stuff and things," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[13] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," *The International Journal of Robotics Research*, vol. 40, no. 12-14, 2021.

[14] R. M. Palliser, L. Teixeira, and M. Chli, "Volumetric Instance-Level Semantic Mapping via Multi-View 2D-to-3D Label Diffusion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 2022.

[15] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for LiDAR odometry and mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of lidar sequences," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[19] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[20] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[21] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent TSDF-based dense mapping approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[22] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, 2018.

[23] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Safe Local Exploration for Replanning in Cluttered Unknown Environments for Micro-Aerial Vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, 2018.