ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

PRS Photogrammetry Remote Sensing
Chair of Photogrammetry and Remote Sensing
Prof. Konrad Schindler

V4RL VISION FOR ROBOTICS LAB
Vision for Robotics Lab
Prof. Margarita Chli

Master Thesis

Mapping for online path planning and 3D reconstruction

Autumn Term 2021

Supervised by:
Yves Kompis
Luca Bartolomei
Ruben Mascaro

Author:
Yue Pan

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Mapping for online path planning and 3D reconstruction

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Yue	Pan

Student supervisor(s)

Yves	Kompis
Luca	Bartolomei
Ruben	Mascaro

Supervising professor(s)

Margarita	Chli
Konrad	Schindler

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zurich, 29th March 2022
Place and date

Yue Pan
Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Acknowledgements **iii**

Abstract **v**

Symbols **vii**

1 Introduction **1**

2 Related Work **3**
2.1 Dense volumetric mapping 3
2.2 Mapping for 3D reconstruction: TSDF map 3
2.3 Mapping for path planning: ESDF map 5

3 Methods **9**
3.1 Non-projective TSDF Mapping 10
3.2 Voxfield ESDF Mapping 11

4 Experiments **17**
4.1 TSDF Mapping Evaluation 17
4.2 ESDF Mapping Evaluation 19
4.3 Multi-resolution Mapping Evaluation 26
4.4 Real-world Path-Planning Experiment 29

5 Summary **31**
5.1 Conclusion 31
5.2 Future work 31

Bibliography **38**

A Experiment data sheet **39**

Acknowledgements

I would like to express my sincere gratitude to those who guided and supported me during this master thesis:

- Prof. Dr. Margarita Chli for giving me the opportunity to work on this project as well as providing me with precious feedback during the fortnightly meeting and before the paper submission. I learned a lot about how to give a clear talk and how to effectively present my work to the audience.
- Prof. Dr. Konrad Schindler for co-hosting this master thesis project and giving me valuable suggestions on the report writing.
- Yves Kompis, Luca Bartolomei and Ruben Mascaro for undertaking the supervision of this thesis. I am really grateful for their responsible and patient supervision and guidance throughout the project. It's not possible for me to finish this thesis and compile the results into a submitted paper without their recommendation of the latest related works, setting up of the real-world experiment and the suggestions for my presentations. The collaboration with them while writing the paper is a great experience in my master study. From this experience, I learned a lot from them about how to tell a story with a logical flow in academic writing.
- My friends Aihui Liu, Bingxin Ke, Jinyan Li, Ruobing Huang and Yihang She for the proofreading of my final presentation.

Abstract

Creating accurate maps of complex, unknown environments is of utmost importance for truly autonomous navigation of mobile robots. 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. This can be extremely problematic in path-planning applications, as such methods cannot always guarantee the safety of the robot. Motivated by these challenges, in this work 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 do 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 in real-time even on large-scale scenarios. Finally, we validate our method by running it onboard a quadrotor, showing it can efficiently generate accurate ESDF maps usable for real-time path planning and obstacle avoidance.
Video – `https://www.youtube.com/watch?v=QbH1aT3zAvs`
Code – `https://github.com/VIS4ROB-lab/voxfield`

Symbols

Symbols

$\mathbf{p}$	point
$\mathbf{x}$	voxel index
$\mathbf{s}$	sensor position
w	observation weight
τ	truncation distance
ϵ	drop-off distance
ψ	projective signed distance
m	sensor noise coefficient
d	non-projective signed distance
d_{max}	maximum ESDF integration distance
$\mathbf{n}$	surface normal vector
ν	voxel size
$\mathbf{g}$	signed distance gradient
D	Voxel TSDF value
D^E	Voxel ESDF value
W	Voxel TSDF weight (confidence)
θ	angle between the ray and signed distance graident vector
α	angle between surface normal vector and signed distance graident vector

Indices

i	voxel index
j	point (ray) index
k	frame index
u	pixel column index
v	pixel row index

Acronyms and Abbreviations

BFS	Breadth-First Search
DLL	Doubly-Linked List
ESDF	Euclidean Signed Distance Field

FIFO	First In, First Out
GPS	Global Positioning System
GNSS-INS	Global Navigation Satellite System and Inertial Navigation System
LiDAR	Light Detection And Ranging
MAV	Micro Aerial Vehicle
ROS	Robot Operating System
RTK	Real-time Kinematic
SLAM	Simultaneous Localization and Mapping
TSDF	Truncated Signed Distance Field
UAV	Unmanned Aerial Vehicle

Chapter 1

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 in the literature addressed this problem by constructing occupancy grids [1], where the space gets discretized into voxels labelled as either occupied or free, the emergence of trajectory optimization-based path-planning methods [2] rendered these representations 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.
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,

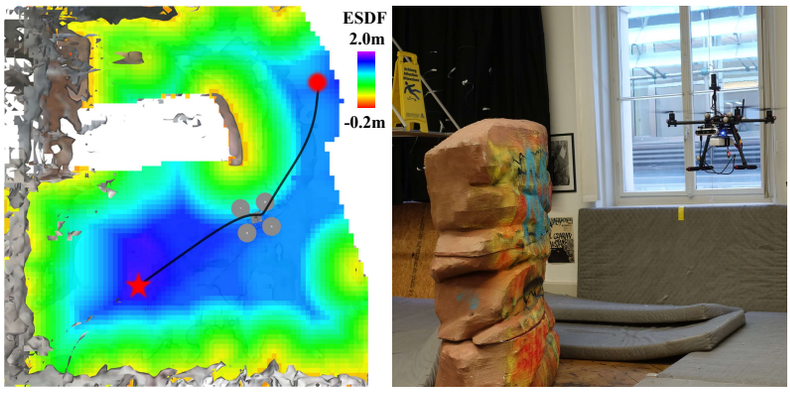


Figure 1.1: Real-world mapping and path planning experiment. A Micro-Aerial Vehicle (MAV) is tasked to navigate toward a goal position while avoiding the obstacle visible on the right. *Voxfield* generates the ESDF map visible on the left onboard the MAV in real-time, which is used for planning. The red dot and the red star is the starting and goal position, respectively.

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 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. 1.1), 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 producing improved scene coverage based on non-projective distances,
- an ESDF update algorithm with sub-voxel accuracy and better computational performance 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, and
- the release of Voxfield's source code[1]

[1] `https://github.com/VIS4ROB-lab/voxfield`

Chapter 2

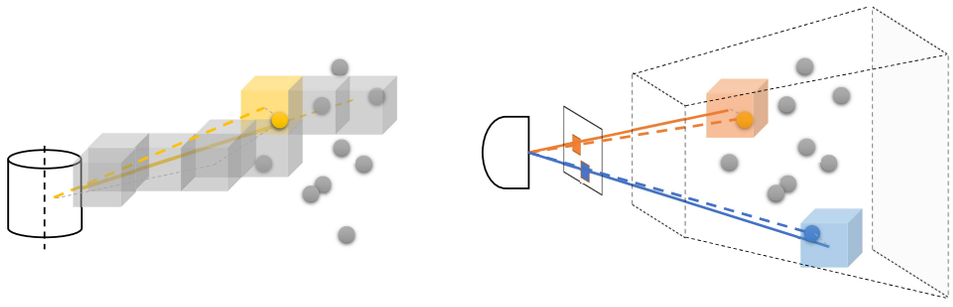
Related Work

2.1 Dense volumetric mapping

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 [7]. 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 [8].
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 the point cloud [9], surfel [10] or sparse feature landmark maps [11] 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 [12] and Signed Distance Fields (SDF) [13]. The SDF maps can be further categorized into Truncated Signed Distance Field (TSDF) and Euclidean Signed Distance Field (ESDF) depending on whether the signed distance is truncated close to the surface.
Occupancy grids represent the map as an evenly spaced random variable field. Each random variable is binary and represents the occupancy probability of the location it covers [14]. Such occupancy grids map can be integrated and updated using Bayes's law as [15]. To cope with the cubic memory cost for 3D grids (voxels), an occupancy grids-based mapping framework using the octree data structure named Octomap [1] is proposed. Octomap enables large-scale occupancy-based volumetric mapping for basic planning [16] and interaction as well as the coarse voxel-level 3D reconstruction [17]. However, the querying and indexing of leaf nodes in the Octomap are not efficient enough.

2.2 Mapping for 3D reconstruction: TSDF map

In contrast to the occupancy grids map, the TSDF map store at each voxel the signed distance to the nearest surface, up to a truncation radius around the surface boundary, as shown in Fig. 2.2a. TSDF map allows for efficient 3D reconstruction into surface meshes with sub-voxel resolutions, e.g. using the marching cubes algorithm [18], which makes them the preferred choice in cases where accurate surface reconstructions are needed. Besides, such mesh representation is easier for human beings to interpret and monitor. Another advantage of the TSDF map is that it can be efficiently integrated and updated from high-rate depth measurements while



(a) Raycasting (b) Projection mapping

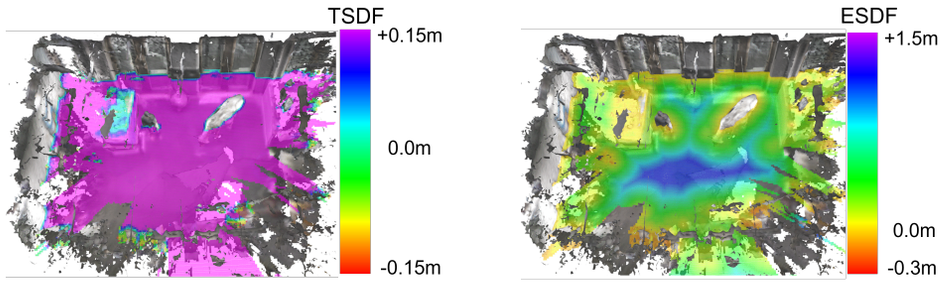
Figure 2.1: Diagram of the two basic TSDF integration methods: raycasting and projection mapping. For raycasting, a ray is cast from the sensor to each measured 3D point (in yellow). Each voxel (in yellow) along the ray would be updated. For projection mapping, each voxel (for example, the red and blue one) is projected to the image plane and updated according to the actual 3D measurement (the point in red and blue) at the projected pixel (in red and blue).

smoothing out sensor noise by taking the weighted average of the projective signed distance from the voxel to the surface along the measurement ray. This is conducted by either the raycasting or the projection mapping method from the 3D measurements. For raycasting integration, as shown in Fig. 2.1a, a ray is cast from the sensor to each 3D point and all the voxels along such ray would be updated. For projection mapping, as shown in Fig. 2.1b, each voxel in the field of view is projected to the image frame and compared with the actual depth measurement on the range image. Though always overestimating the actual value, the projective signed distance acts as a reasonable approximation when the voxel is close to the surface.

Using the aforementioned integration method, as first shown by KinectFusion [19], TSDFs have increasingly become the representation of choice in nowadays mapping frameworks [20, 21, 3]. However, KinectFusion is designed to run on GPUs in a small-scale indoor environment, thus not suitable for applications on mobile robots. Kintinuous [22] and RGB-D SLAM [23] are proposed to make KinectFusion be scalable to large-scale scenario applications. For deployment on robots or devices with limited computing resources (a single CPU), Chisel [21] and Voxblox [3] are proposed. They are built upon the voxel hashing [24] data structure for the consideration of map memory and querying efficiency. Voxblox decoupled the pose tracking [25] and mapping module of KinectFusion and only focus on the "mapping with known pose" task. In Voxblox, to improve the TSDF integration speed when using a large voxel size, a bundled or merged raycasting strategy is proposed. Other data structures such as VDB tree [26], octree [27] are also adopted for the TSDF mapping for better memory and time efficiency.

More recently, TSDFs have continued being adopted as underlying representations in approaches targeting large-scale outdoor reconstruction [28], global consistent reconstruction [29, 30, 31, 32], dynamic-resilient reconstruction [33] or exploring the incorporation of higher-level information, such as semantic or object instance labels, into purely geometric maps [34, 35, 36, 37, 38, 6, 39, 40]. Most of these works are built on top of the open-sourced implementation of Voxblox[1]. In other words, improving the mapping performance of Voxblox can also contribute to the performance-boosting of the aforementioned high-level mapping systems.

[1] `https://github.com/ethz-asl/voxblox`



(a) TSDF map (b) ESDF map

Figure 2.2: An example of the TSDF and ESDF map of an indoor scene. The voxel size is 5cm. For clarity, only a slice of the map is shown.

To deal with the trading-off map quality against efficiency with a fixed voxel, the multi-resolution TSDF mapping has been proposed. The adaptive voxel size can be determined according to the level of the measurement noise [41, 42] or the semantic label of the reconstructed object [6]. The noise-aware multi-resolution strategy requires a good sensor noise model and uncertainty estimation while the semantic-aware counterpart assigns small voxels to foreground small-scale instances to keep the details and assigns large voxels to the background to conduct a rough description according to the semantic prediction.
What's more, recently, with the development of deep learning, neural implicit representations [43, 44] have been used for 3D reconstruction. However, the incremental integration of such neural SDF remains a challenging topic. Up to now, the initial trials such as RoutedFusion [45], NeuralFusion [46], DI-Fusion [47] and NeuralBlox [48] have achieved better or on-par reconstruction quality compared with the traditional TSDF integration methods but still suffer from the scalability and the deployment on mobile robots with limited computational resources.

2.3 Mapping for path planning: ESDF map

When it comes to path planning, however, ESDF maps (Fig. 2.2b), which have no truncation of the signed distance, 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 STOMP [49], CHOMP [2], [50] and [51], for which occupancy information alone is not sufficient. ESDF maps enable fast collision checking and clearance cost calculation for these path planners.
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. Unlike TSDF maps, it's non-trivial to build an ESDF map efficiently directly from the sensor measurements. The projective signed distance along the measurement ray is no longer a bearable approximation of the Euclidean signed distance when the voxel is far away from the surface. An intermediate volumetric map representation such as the up-to-date occupancy grid map or the TSDF map is therefore needed for incremental ESDF mapping. Earlier methods such as Brushfire [52] can only handle the batch ESDF construction from a static occupancy grid map using graph search algorithms. Such batch mapping method has to update the whole distance field once there are some changes in the environment. Afterwards, various methods [53, 54, 55, 56, 57] have been proposed based on Brushfire to deal

with the incremental ESDF mapping in a dynamic environment. However, these methods are still not efficient enough for real-time performance onboard a mobile robot with limited computational resources.
Such incremental ESDF mapping algorithms typically consist of two key operations [5]: a *decrease* function used for updating the regions affected by newly inserted obstacles (the distance value would decrease) and an *increase* function used for clearing the regions affected by removed obstacles (the distance value would increase). In recent studies, by using specialized data structures for the *decrease* and *increase* update functions, the accuracy and efficiency of incremental ESDF mapping have been further improved. For example, as shown in Fig. 2.3, Voxblox [3] progressively builds an ESDF map from a projective-TSDF map that integrates the incoming sensor data. Instead of computing the distance to the nearest occupied voxel, Voxblox makes use of the distance stored in the TSDF map in a fixed band close to the surface. Based on the updated TSDF values, all the voxels are *increased* first and then *decreased* to reduce the bookkeeping effort.

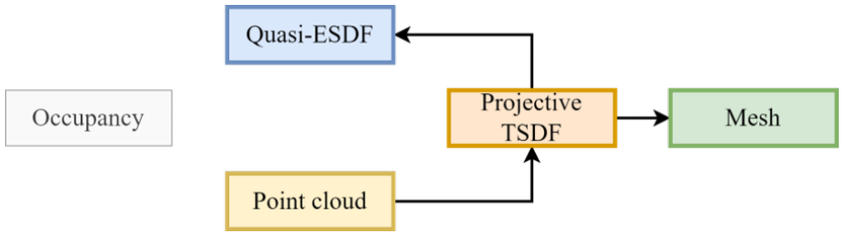


Figure 2.3: Voxblox mapping flow chart

Other recent approaches, such as FIESTA [4] and EDT [5] take advantage of specialized data structures for fast incremental ESDF updates from occupancy maps. As shown in Fig. 2.4, FIESTA actually builds an unsigned Euclidean distance field (EDF) incrementally from the occupancy map. By using the doubly linked lists data structure for the bookkeeping of each voxel's closest occupied voxel, FIESTA can achieve significant efficiency improvement on the *increasing* update than Voxblox. Similar to FIESTA, as shown in Fig. 2.5, EDT constructs ESDF from the occupancy map efficiently with the optimized *increasing* and *decreasing* update schedule priorities. Since the TSDF map is not constructed in both FIESTA and EDT, the 3D reconstruction quality would be sacrificed for the more accurate and efficient ESDF mapping.

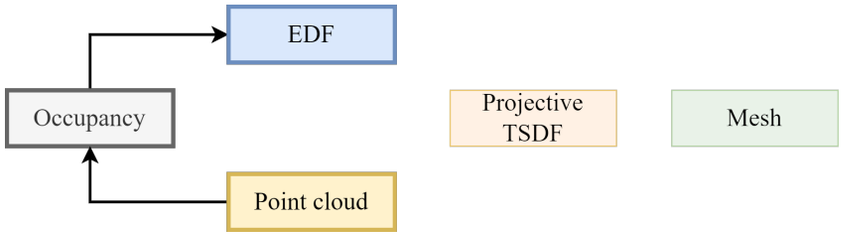


Figure 2.4: FIESTA mapping flow chart

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, which is the accumulated distance of the

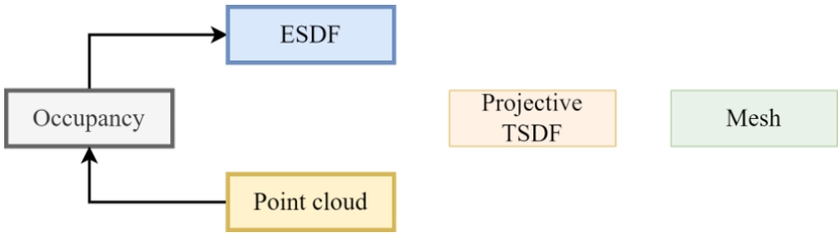


Figure 2.5: EDT mapping flow chart

polyline along the graph searching path. 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 thesis, we advocate for building ESDFs from an underlying TSDF-based map representation that is well-suited for fine-grained surface reconstruction, as shown in Fig. 2.6.

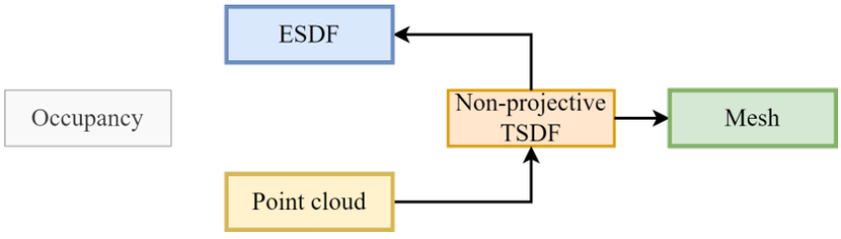


Figure 2.6: The proposed Voxfield mapping flow chart

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, 4, 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 increasing the quality of the generated maps. Finally, we show that Voxfield can also run online onboard an MAV, generating maps that can be directly used for path planning.

Chapter 3

Methods

Voxfield processes incoming 3D measurements with known associated poses, estimated e.g. by a Simultaneous Localization and Mapping (SLAM) system or a Global Navigation Satellite System and Inertial Navigation System (GNSS-INS) fusion, 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. 3.1, raw 3D measurements are first integrated into a non-projective TSDF map (Section 3.1), which serves as a base for generating a full ESDF map (Section 3.2). The system also outputs a 3D mesh of the scanned environment, which is generated from the TSDF using the marching cubes algorithm [18] as in [3].

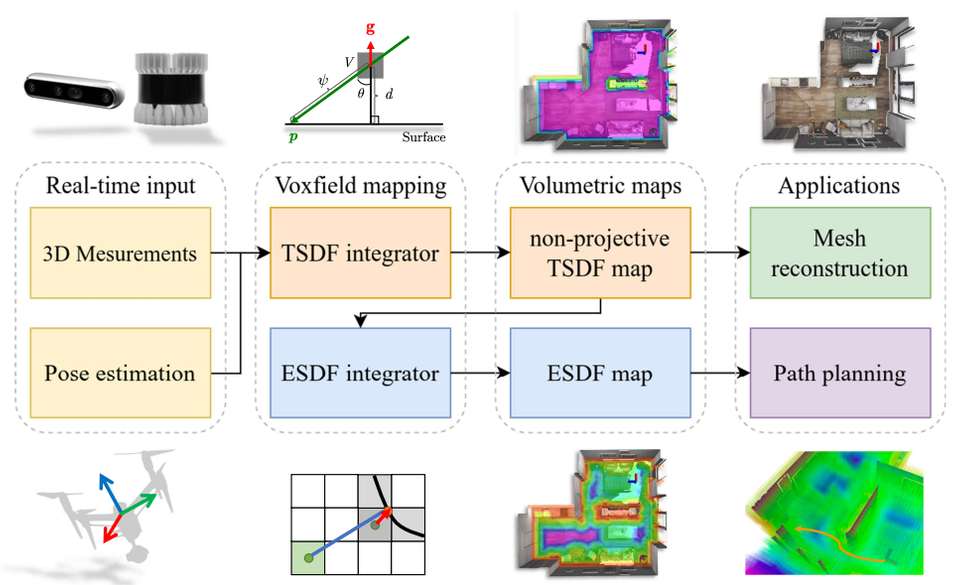


Figure 3.1: Overview of the proposed Voxfield mapping framework. Sensor measurements with associated poses are integrated into a TSDF map, suitable e.g. for mesh reconstruction, and subsequently into a full ESDF map, used for path planning.

3.1 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 $\mathbf{g}_i \in \mathbb{R}^3$, similar to [58].

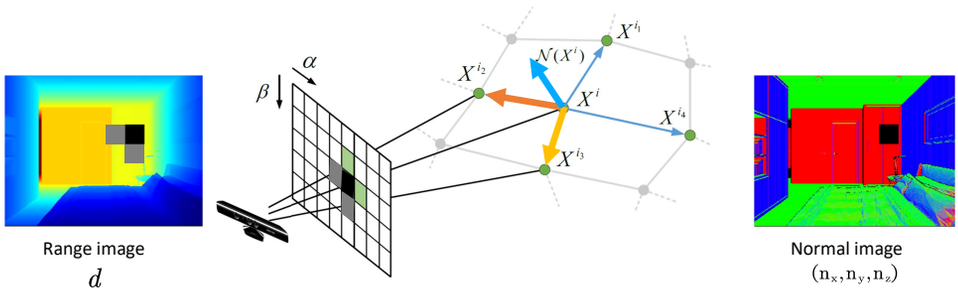


Figure 3.2: Efficient normal estimation using neighbouring measurements cross product from range image as the pre-processing step

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 [19] (shown in Fig. 3.2), together with the sensor's pose $\mathbf{T}_{wk} \in \mathrm{SE}(3)$ in the world frame w. The normal image $\mathcal{N}_k$ at frame k can be calculated from the vertex image $\mathcal{V}_k$ as:

$$\begin{aligned}\mathcal{N}_k(u,v) =& \mathrm{norm}(((\mathcal{V}_k(u+1,v) - \mathcal{V}_k(u,v)) \\ &\times (\mathcal{V}_k(u,v+1) - \mathcal{V}_k(u,v))),\end{aligned} \tag{3.1}$$

where the vertex image is generated from the range image pixel-wise and u, v represents the column and row index of the pixel.
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:

$$\psi_{ijk} = \mathrm{sign}\left((\mathbf{p}_j - \mathbf{s}_k) \cdot (\mathbf{p}_j - \mathbf{x}_i)\right) \|\mathbf{p}_j - \mathbf{x}_i\|. \tag{3.2}$$

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:

$$w_{ijk} = w_{jk}^{(\mathrm{sensor})} w_{ijk}^{(\mathrm{dropoff})}. \tag{3.3}$$

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

$$w_{jk}^{(\mathrm{sensor})} = \frac{1}{\|\mathbf{p}_j - \mathbf{s}_k\|^m}, \tag{3.4}$$

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, $w_{ijk}^{(\mathrm{dropoff})}$, represents a behind-the-surface drop-off that is used to

improve the reconstruction quality of thin surfaces. Formally, it is defined as

$$w_{ijk}^{(\text{dropoff})} = \begin{cases} 1 & \psi_{ijk} \geq -\epsilon \\ \dfrac{\tau + \psi_{ijk}}{\tau - \epsilon} & -\tau < \psi_{ijk} < -\epsilon \\ 0 & \psi_{ijk} \leq -\tau \end{cases}, \tag{3.5}$$

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 $\mathbf{p}_j$:

$$\mathbf{g}_i \leftarrow \frac{\tilde{\mathbf{g}}_i}{\|\tilde{\mathbf{g}}_i\|}, \quad \text{with} \quad \tilde{\mathbf{g}}_i = \frac{W_i \mathbf{g}_i + w_{ijk} \mathbf{n}_j}{W_i + w_{ijk}}. \tag{3.6}$$

This results in a precise estimate of the gradient direction in the trivial case of reconstructing a flat surface, as seen in Fig. 3.3b. In the more general case of a curved surface, as shown in Fig. 3.3a, we rely on the movement of the sensor to estimate the gradient accurately over time, even with non-optimal normal estimations from every 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:

$$d_{ijk} = \begin{cases} |\cos\theta| \, \psi_{ijk} & \text{if } \alpha = 0 \\ \left| \dfrac{(\cos\alpha - 1)\sin\theta}{\sin\alpha} + \cos\theta \right| \psi_{ijk} & \text{otherwise} \end{cases}, \tag{3.7}$$

where θ represents the angle between the ray and the gradient

$$\cos\theta = \frac{\mathbf{p}_j \cdot \mathbf{g}_i}{\|\mathbf{p}_j\|}, \tag{3.8}$$

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

$$\cos\alpha = \mathbf{n}_j \cdot \mathbf{g}_i. \tag{3.9}$$

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. 3.3b and Fig. 3.3c, respectively.
Finally, the non-projective truncated signed distance D_i and the weight W_i can be updated as follows:

$$D_i \leftarrow \frac{W_i D_i + w_{ijk} \text{trunc}\left(d_{ijk}, \tau\right)}{W_i + w_{ijk}}, \text{ and} \tag{3.10}$$

$$W_i \leftarrow \min\left(W_i + w_{ijk}, W_{\max}\right), \tag{3.11}$$

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

$$\text{trunc}(d, \tau) = \begin{cases} \text{sign}\left(d\right)\tau & \text{if } |d| > \tau \\ d & \text{otherwise} \end{cases}. \tag{3.12}$$

3.2 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

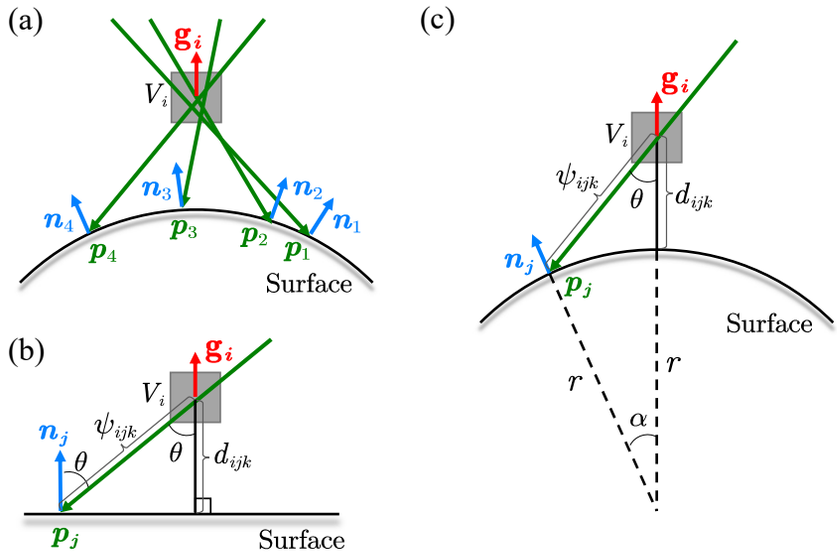
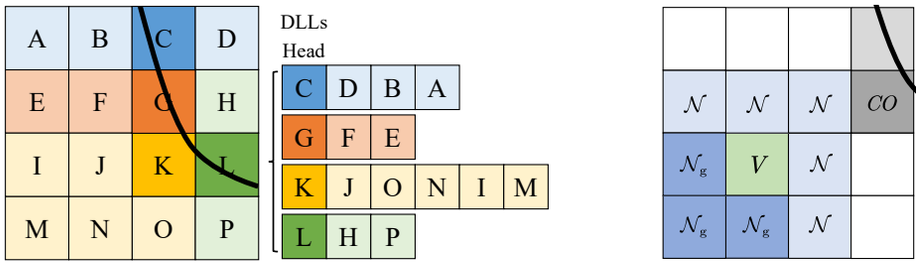


Figure 3.3: Non-projective TSDF calculations for voxel V_i: the updated voxel is represented in grey, rays from raycasting in green, the surface normals in blue and the field gradient in red. (a) shows the integration of voxel V_i's TSDF-gradient by taking a weighted average of the normals on the surface 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.

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.

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. Compared with using the "first-in, first-out" (FIFO) queue, using the bucketed priority queue with a priority of the absolute value of the voxel's ESDF to keep track of the updated voxels can boost the speed of the BFS. 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. 3.4b, it is sufficient to check the neighbours on the far side of the closest occupied voxel, while the majority of neighbouring cells, lying between the current voxel and the closest occupied position, can be safely ignored.



(a) Doubly-linked lists (b) Efficient neighbour querying

Figure 3.4: 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 second fundamental step in the ESDF update is the removal of obstacles that are no longer present on 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. 3.4a, 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 the default value d_{max} or the distance to the nearest voxel among their neighbours' closest occupied voxels (Line 18). Following this strategy, the BFS step (Line 26) can properly update all the voxels to their new 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 (Line 37) if the voxel is inside the truncation band, or we calculate the true Euclidean distance D_i^E from the voxel to the closest surface (Line 39) as:

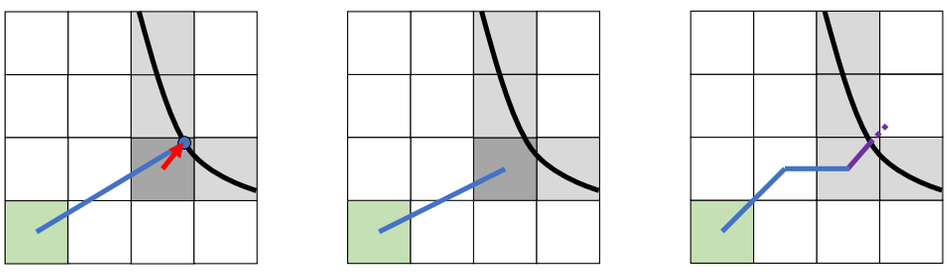
$$D_i^E = \begin{cases} \text{sign}(D_i)\|\mathbf{x}_i - (\mathbf{x}_i^{co} - D_i^{co}\mathbf{g}_i^{co})\| & \text{if } |D_i| > \tau \\ D_i & \text{otherwise} \end{cases}, \tag{3.13}$$

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. 3.5a). 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. 3.5c), FIESTA calculates the true Euclidean distance between the voxel and its closest occupied voxel's center (Fig. 3.5b).

Algorithm 1 Voxfield ESDF map update

Input: Updated TSDF map $M_T(k)$ with voxels $\{V_T\}$, previous ESDF map $M_E(k-1)$ with voxels $\{V_E\}$ and doubly-linked lists $\{dll(\mathbf{v})\}$
Output: Updated ESDF map $M_E(k)$ with voxels $\{V_E\}$

for each v **in** GetUpdatedVoxelIndices(M_T)
  if not $V_E(\mathbf{v}).occupied$ **and** IsOccupied($V_T(\mathbf{v})$)
    insertList.push($\mathbf{v}$)
  if $V_E(\mathbf{v}).occupied$ **and** not IsOccupied($V_T(\mathbf{v})$)
    deleteList.push($\mathbf{v}$)
  $V_E(\mathbf{v}).occupied \leftarrow$ IsOccupied($V_T(\mathbf{v})$)
  $V_E(\mathbf{v}).fixed \leftarrow (V_T(\mathbf{v}).d \leq \tau)$
for each v **in** *insertList*
  $V_E(\mathbf{v}).d \leftarrow 0$
  $V_E(\mathbf{v}).co \leftarrow \mathbf{v}$
  $dll(\mathbf{v})$.insert($\mathbf{v}$)
  updateQueue.push($\mathbf{v}$, 0)
for each v **in** *deleteList*
  for each u **in** $dll(\mathbf{v})$
    $dll(\mathbf{v})$.delete($\mathbf{u}$)
    $V_E(\mathbf{u}).d \leftarrow d_{max}$
    $V_E(\mathbf{u}).co \leftarrow$ null
    for each n **in** GetAllNeighbours($\mathbf{u}$)
      $\mathbf{m} \leftarrow V_E(\mathbf{n}).co$
      if $V_E(\mathbf{m}).occupied$ and Distance($\mathbf{m}$, $\mathbf{u}$) $< V_E(\mathbf{u}).d$
        $V_E(\mathbf{u}).d \leftarrow$ Distance($\mathbf{m}$, $\mathbf{u}$)
        $V_E(\mathbf{u}).co \leftarrow \mathbf{m}$
    if $V_E(\mathbf{u}).co \neq$ null
      $dll(V_E(\mathbf{u}).co)$.insert($\mathbf{u}$)
      updateQueue.push($\mathbf{u}$, $V_E(\mathbf{u}).d$)
while *updateQueue* $\neq \emptyset$
  $\mathbf{v} \leftarrow$ *updateQueue*.front()
  updateQueue.pop()
  for each n **in** GetAffectedNeighbours($\mathbf{v}$, $V_E(\mathbf{v}).co$)
    if Distance($V_E(\mathbf{v}).co$, $\mathbf{n}$) $< V_E(\mathbf{n}).d$
      $dll(V_E(\mathbf{n}).co)$.delete($\mathbf{n}$)
      $V_E(\mathbf{n}).d \leftarrow$ Distance($V_E(\mathbf{v}).co$, $\mathbf{n}$)
      $V_E(\mathbf{n}).co \leftarrow V_E(\mathbf{v}).co$
      $dll(V_E(\mathbf{n}).co)$.insert($\mathbf{n}$)
      updateQueue.push($\mathbf{n}$, $V_E(\mathbf{n}).d$)
  if $V_E(\mathbf{v}).fixed$
    $V_E(\mathbf{v}).\tilde{d} \leftarrow V_T(\mathbf{v}).d$
  else
    $V_E(\mathbf{v}).\tilde{d} \leftarrow$ ExactESDFValue($\mathbf{v}$, $V_E(\mathbf{v}).co$)



(a) Voxfield (b) FIESTA (c) Voxblox

Figure 3.5: 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).

Chapter 4

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, as shown in Fig. 4.1. The main characteristics of these datasets, collected by either RGB-D cameras or LiDAR sensors, are summarized in Table 4.1. For the synthetic datasets, the pose is given by the planned trajectory in simulation while the ground truth point cloud is downsampled from the CAD model used for simulation and rendering. For the Cow&Lady dataset, the ground truth pose is provided by a Vicon motion capture system and the ground truth point cloud is collected by a survey-level laser scanner with millimetre accuracy. For the KITTI dataset [59], a representative sequence seq.07 is adapted for the quantitative evaluation. For the test sequences (seq.11-21) in the KITTI dataset, where the ground truth pose is not provided, we take the pose estimated by MULLS [9], a state-of-the-art LiDAR SLAM system, as the ground truth. The ground truth point cloud for evaluation is also generated by accumulating the point cloud of each frame with the ground truth pose.
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 on the KITTI dataset, where the accuracy-efficiency trade-off plays a major role. These experiments on publicly available datasets are conducted on a PC equipped with an Intel Core i7-7700HQ@2.80GHz CPU for a fair comparison.
Finally, we conduct a real-world experiment, where an MAV is tasked to perform online obstacle avoidance, proving the usefulness of the ESDF map generated by Voxfield on-the-fly.

Table 4.1: Four public datasets used for the experiments

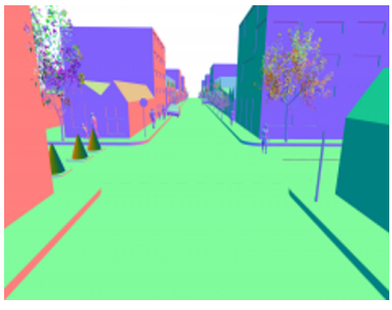
Dataset	Type	Sensor	Pose	Ground truth Point Cloud
Flat [6]	synthetic	RGB-D	Ground Truth	CAD model sampling
Cow&Lady [3]	real-world	RGB-D	Vicon	TPS-MS50 laser scan
MaiCity [60]	synthetic	LiDAR	Ground Truth	CAD model sampling
KITTI [59]	real-world	LiDAR	GNSS-INS	Scan accumulation by pose

4.1 TSDF Mapping Evaluation

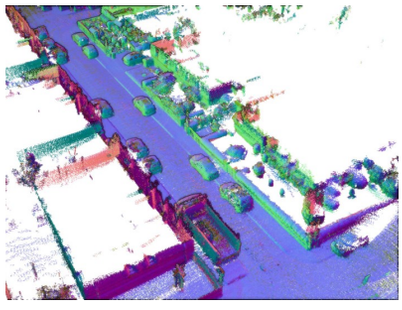
We evaluate the quality of the proposed non-projective TSDF mapping approach based on three metrics that are calculated with respect to the ground-truth point cloud: the TSDF error, the reconstruction Chamfer distance, and the reconstruction coverage.

(a) Flat synthetic RGB-D dataset

(b) Cow&Lady real-world RGB-D dataset



(c) MaiCity synthetic LiDAR dataset



(d) KITTI real-world LiDAR dataset

Figure 4.1: Four publicly available datasets used for the experiments

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.

Chamfer Distance

As a common metric for reconstruction quality, the Chamfer-L1 distance [43] is calculated between the vertices of the reconstructed mesh $\mathcal{M}$ and the ground-truth point cloud $\mathcal{G}$ as:

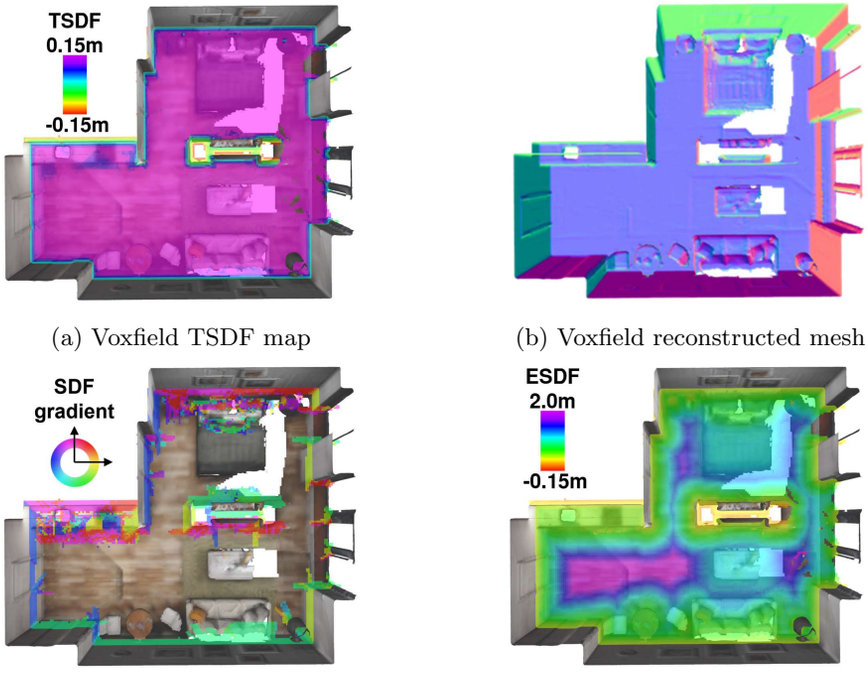
$$d_{\mathrm{CD}}\left(\mathcal{M}, \mathcal{G}\right) = \frac{1}{2\left|\mathcal{M}\right|} \sum_{\mathbf{m} \in \mathcal{M}} \min_{\mathbf{g} \in \mathcal{G}} \|\mathbf{m} - \mathbf{g}\| + \frac{1}{2\left|\mathcal{G}\right|} \sum_{\mathbf{g} \in \mathcal{G}} \min_{\mathbf{m} \in \mathcal{M}} \|\mathbf{g} - \mathbf{m}\| \ . \tag{4.1}$$

Chamfer distance is calculated by taking both the reconstructed model and the ground truth model as the reference, thus balancing both the reconstruction precision and recall.

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. For efficient evaluation, we build a kd-tree for both the mesh vertices and the ground truth point cloud for efficient nearest neighbour querying.

Our Voxfield's TSDF mapping evaluation is conducted on all four datasets with different voxel size settings with an interval of 5cm. Various voxel size settings are tested because we want to have an overall evaluation for both the fast but coarse and fine but slow volumetric mapping. We also report the performance scores



(a) Voxfield TSDF map

(b) Voxfield reconstructed mesh

(c) Voxfield signed distance gradient map

(d) Voxfield ESDF map

Figure 4.2: Voxfield mapping representations on the synthetic Flat dataset with a voxel size of 5 cm.

obtained by running the same experiments with the projective-TSDF integrator proposed in Voxblox [3], which is regarded as the baseline for comparison. In our Voxfield, the TSDF map (Fig. 4.2a) can be generated from a sequence of sensor measurements together with the signed distance gradient in each voxel (Fig. 4.2c) and further evolve to the reconstructed mesh (Fig. 4.2b) via the marching cubes algorithm. As shown in Fig. 4.3, our non-projective TSDF map representation achieves a lower TSDF error for all tested voxel sizes, with a more notable difference in the LiDAR datasets, where large incidence angles are more common. As shown in Fig. 4.4, the mesh reconstruction quality of Voxfield, 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 neighbouring voxels at zero-crossings of the field. Such distance ratio hardly changes by conducting the non-projective correction even though the absolute error of the signed distance of each voxel decreases. Nonetheless, as shown in Fig. 4.5 ,our Voxfield achieves higher reconstruction coverage on all four datasets, as is also visible in the qualitative comparisons shown in Fig. 4.6. The 3D reconstruction by Voxfield is more complete in the highlighted regions. This effect is especially noticeable at the boundaries of the scans and results from the fact that the projective TSDF mapping [3] generally overestimates the signed distance value, leading to more truncated voxels than under the non-projective TSDF map built by Voxfield.

4.2 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.

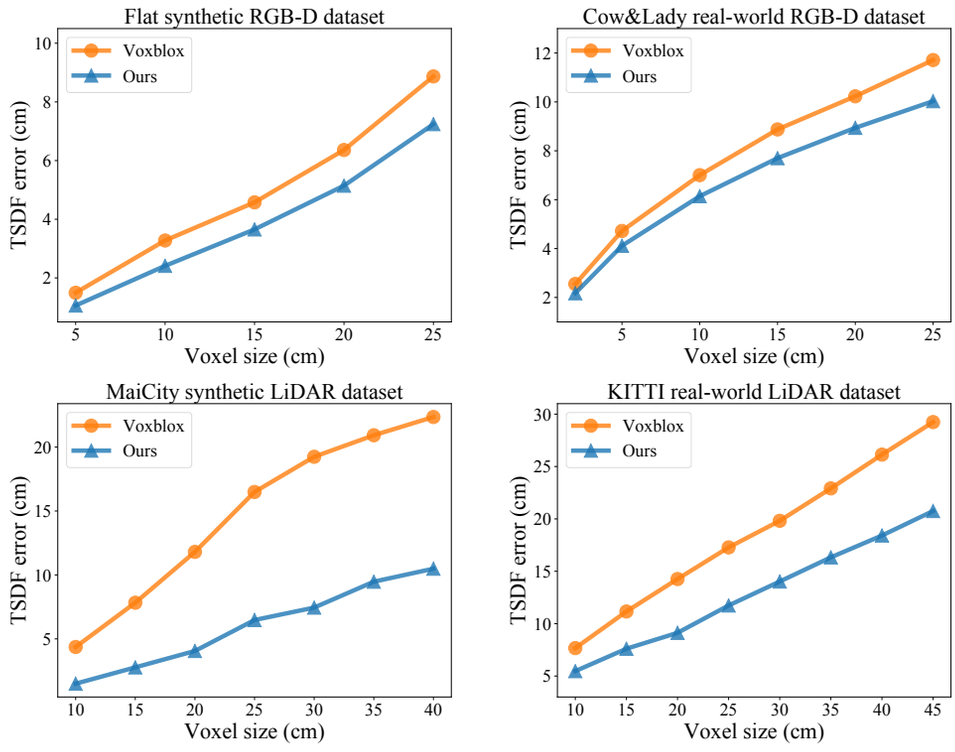


Figure 4.3: A comparison of methods for TSDF mapping on each dataset from Table 4.1 with different voxel sizes. The proposed Voxfield is compared against Voxblox on the TSDF error. A smaller TSDF error is favoured.

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.

ESDF Update Time

The ESDF update time is the average time consumed 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 so that they use the same TSDF integration front-end as Voxfield. In the cases of FIESTA and EDT, where the ESDF is built from an occupancy map, the occupancy status is inferred from the TSDF map.
As shown in Fig. 4.7, our approach outperforms the compared state-of-the-art methods on ESDF accuracy. Results demonstrate that, by calculating the true Euclidean distance based on a non-projective TSDF, Voxfield's ESDF error is cut in half compared to Voxblox. In addition, by eliminating the discretization error, Voxfield

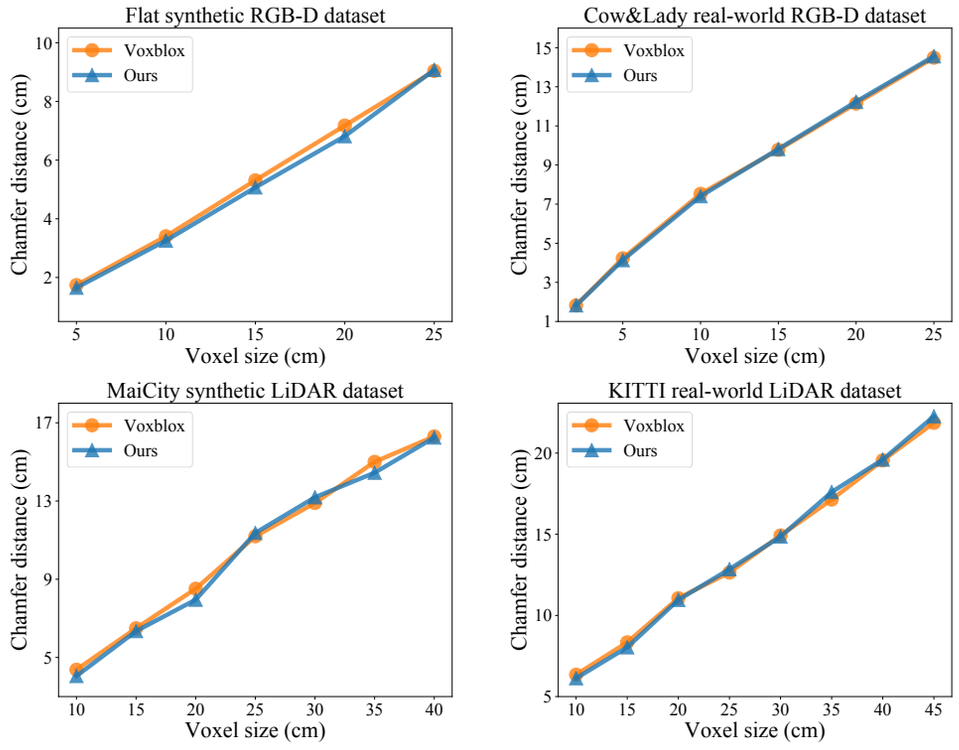


Figure 4.4: A comparison of methods for TSDF mapping on each dataset from Table 4.1 with different voxel sizes. The proposed Voxfield is compared against Voxblox on the reconstruction Chamfer-L1 distance. A smaller Chamfer distance is favoured.

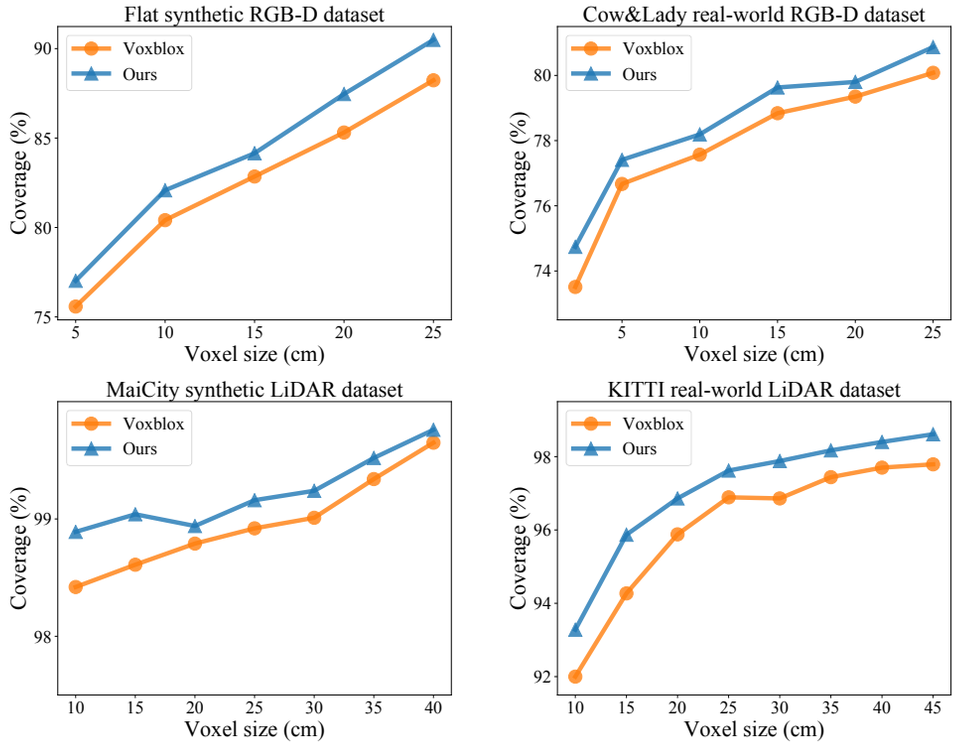
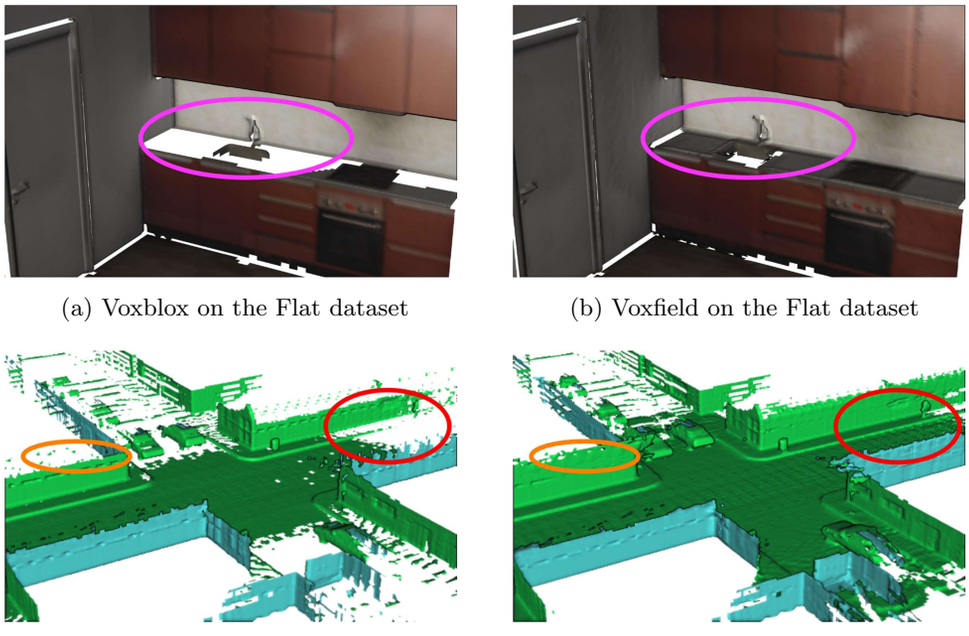


Figure 4.5: A comparison of methods for TSDF mapping on each dataset from Table 4.1 with different voxel sizes. The proposed Voxfield is compared against Voxblox on the reconstruction coverage. A larger coverage is favoured.



(a) Voxblox on the Flat dataset

(b) Voxfield on the Flat dataset

(c) Voxblox on the MaiCity dataset

(d) Voxfield on the MaiCity dataset

Figure 4.6: Qualitative comparison of the 3D reconstruction results 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.

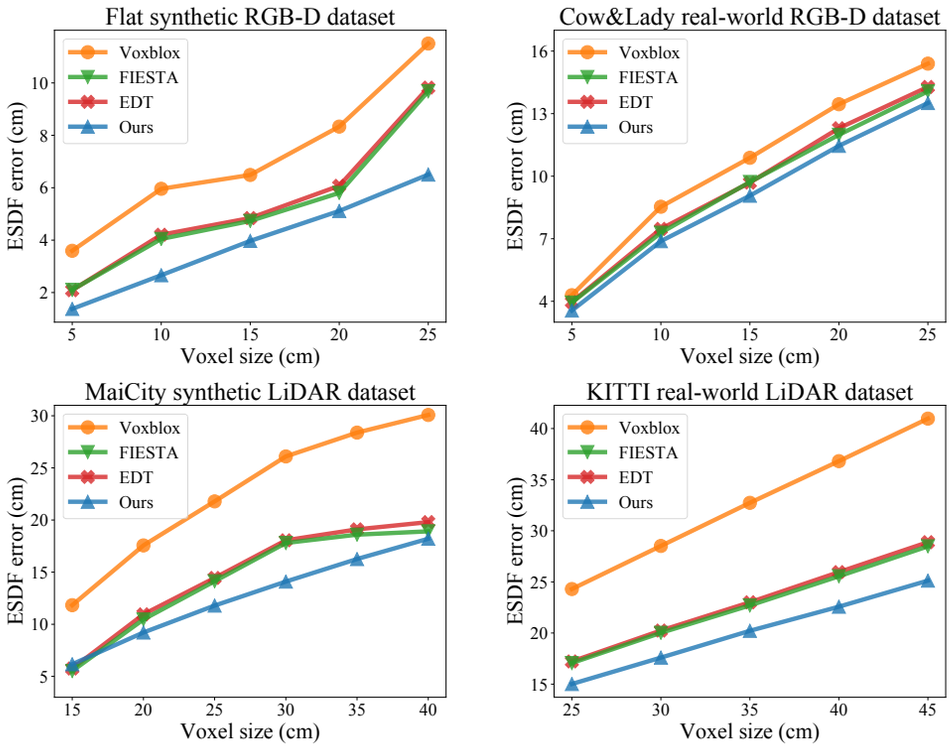


Figure 4.7: A comparison of methods for ESDF mapping on each dataset from Table 4.1 with different voxel sizes. The proposed Voxfield is compared against Voxblox, FIESTA and EDT on the ESDF error. A smaller ESDF error is favoured.

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 on the flat dataset is shown in Fig. 4.9. It is shown that Voxfield has the smallest error using either the occupied voxel center-based metric used in [4] or the ground truth surface-based metric used in this thesis. It can be also noticed that Voxfield almost has no error when using the metric in [4].

Regarding the ESDF update efficiency, as shown in Fig. 4.8, our Voxfield runs about three times faster than Voxblox due to the usage of doubly linked lists data structure for bookkeeping and is more than 10% faster than FIESTA and 15% faster than EDT thanks to the proposed efficient directional neighbor search.

In summary, the characteristic of Voxfield and three other ESDF mapping algorithms are compared in Table 4.2.

Table 4.2: Comparison of different incremental ESDF mapping algorithms

Method	From	To	Discretization	Accuracy	Efficiency
Voxblox [3]	proj-TSDF	quasi ESDF	no	low	slow
FIESTA [4]	Occupancy	true EDF	yes	high	fast
EDT [5]	Occupancy	true ESDF	yes	high	fast
Voxfield	TSDF	true ESDF	no	highest	fastest

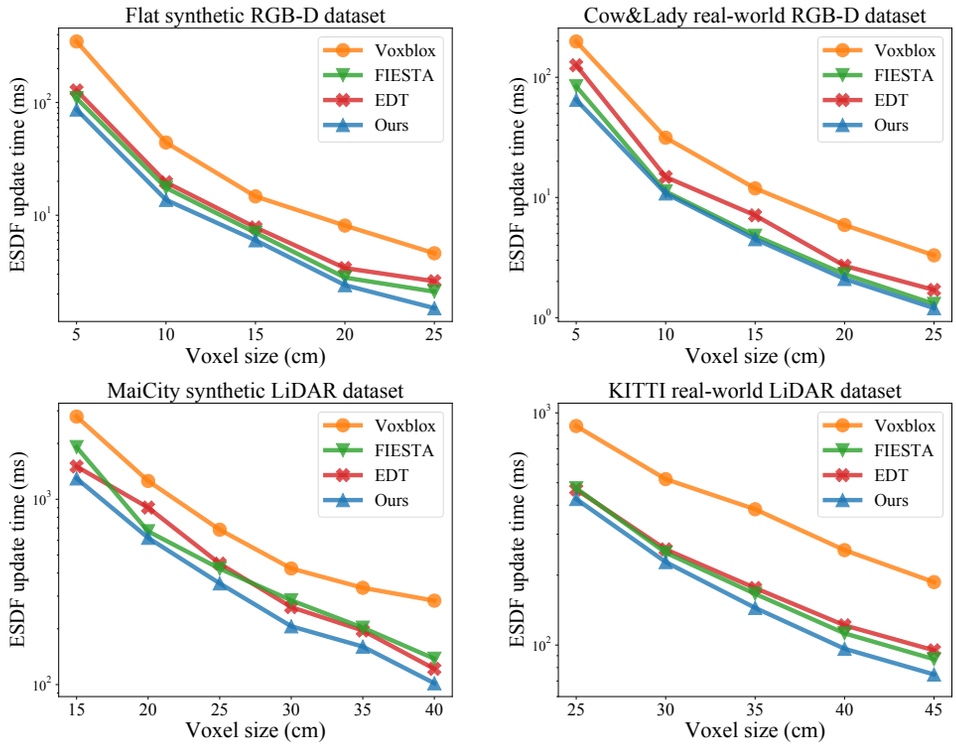
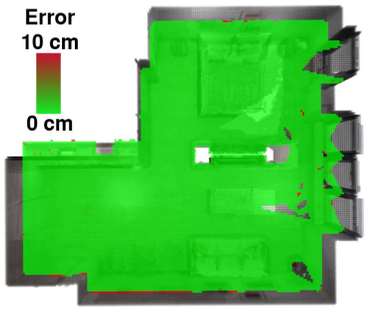
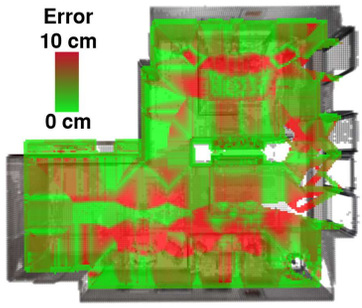


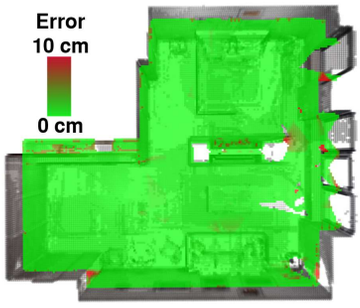
Figure 4.8: A comparison of methods for ESDF mapping on each dataset from Table 4.1 with different voxel sizes. The proposed Voxfield is compared against Voxblox, FIESTA and EDT on the ESDF update time. The smaller update time is favoured.



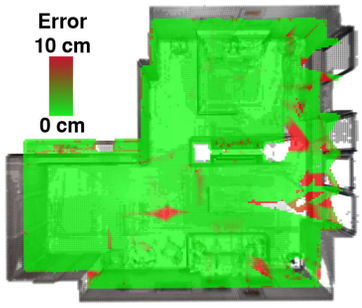
(a) Voxfield ESDF error (reference: occupied voxel center points)



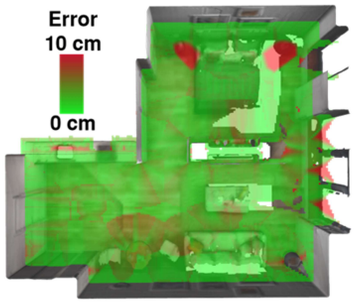
(b) Voxblox ESDF error (reference: occupied voxel center points)



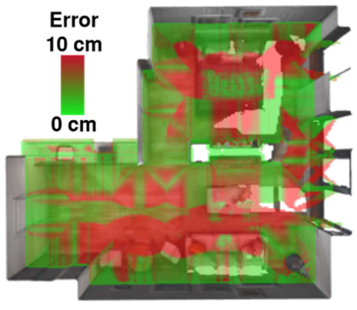
(c) FIESTA ESDF error (reference: occupied voxel center points)



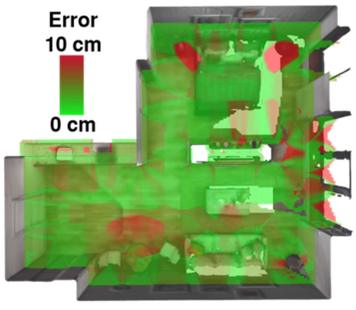
(d) EDT ESDF error (reference: occupied voxel center points)



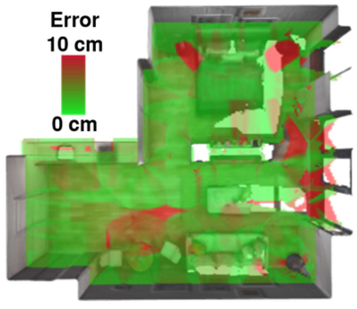
(e) Voxfield ESDF error (reference: ground truth point cloud)



(f) Voxblox ESDF error (reference: ground truth point cloud)



(g) FIESTA ESDF error (reference: ground truth point cloud)



(h) EDT ESDF error (reference: ground truth point cloud)

Figure 4.9: Qualitative comparison of ESDF mapping accuracy on the synthetic Flat dataset with a voxel size of 5 cm. ESDF error calculated with regards to the occupied voxel center (the metric used in [4] and the ground truth surface (the metric used in this thesis) are both shown.

4.3 Multi-resolution Mapping Evaluation

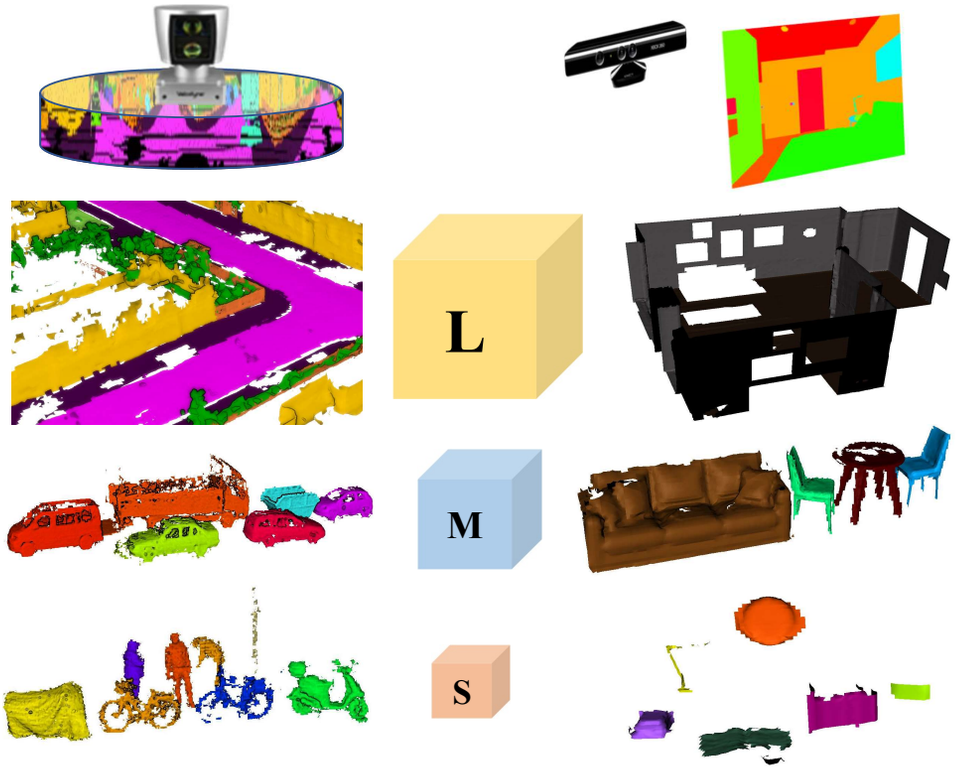


Figure 4.10: Multi-resolution voxel allocation according to the panoptic label for both the LiDAR and RGB-D camera input. For example, large voxels are assigned to backgrounds such as roads and buildings for outdoor LiDAR input and room layouts for indoor RGB-D input. Medium voxels are assigned to cars for outdoor LiDAR input and medium-size furniture for indoor RGB-D input. Small voxels are assigned to pedestrians and bicycles for outdoor LiDAR input and small instances such as lamps and books for indoor RGB-D input.

Aiming to further exploit 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 a collection of TSDF submaps, each representing a single object instance with the voxel size determined by its semantic class, just as shown in Fig. 4.10. This allows certain object classes such as pedestrians and bikes 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. The open-source implementation of the original Panmap[1] uses Voxblox as its underlying mapping framework, can only work with RGBD camera input and does not support the ESDF mapping for online path planning. 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 only for RGB-D sensors, to enable the panoptic mapping pipeline to process LiDAR input, as shown in Fig. 4.11. This is done by a point cloud to range image conversion according to the laser beam distribution of the LiDAR, as described in [61].

[1] `https://github.com/ethz-asl/panoptic_mapping`

To boost the efficiency, we propose an adaptive TSDF integration strategy, that fits well for the multi-submap multi-resolution scenario. For each submap in the collection, it selects between raycasting with the complexity of $\mathcal{O}(N_r l_r)$ and projection mapping with the complexity of $\mathcal{O}(N_v)$, where N_r, l_r and N_v are the numbers of rays, the number of voxels along each ray and the number of voxels in the view frustum, respectively. The integration method with fewer involved voxels would be selected before actually performing the integration. The updated TSDF submaps would be further parallelly reconstructed as multi-resolution meshes through the marching cubes algorithm.

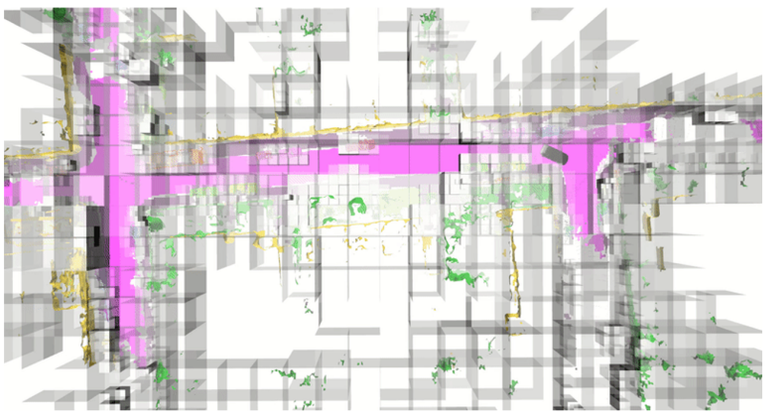


Figure 4.11: Panoptic multi-resolution volumetric mapping on the KITTI dataset. Voxel blocks (the voxel hashing block with $16 \times 16 \times 16$ voxels in it) with various sizes are visualized in a single map. Different color represents different panoptic labels.

We demonstrate the advantages of using Voxfield as a mapping back-end by conducting experiments on the KITTI dataset, where ground-truth 2D panoptic labels are available [62]. 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++ [61] combined with a depth-based clustering [63], which is light enough for real-time performance e.g. on a mobile GPU.

We compare our Voxfield Panmap against the original multi-resolution Panmap [6], as well as the single-resolution mapping systems Voxblox [3] and C-blox [30]. C-blox is a multi-submap and faster version of Voxblox, enabling scalable and efficient volumetric mapping. Fig. 4.12 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. 4.13b and Fig. 4.13c. It can also be noticed from Fig. 4.13d, Fig. 4.13e and Fig. 4.13f that the 3D reconstruction of the single resolution Voxblox cannot cover the details and background completeness at the same time. More qualitative results of our Voxfield-based Panmap on KITTI dataset with the ground truth panoptic labels are shown in Fig. 4.14.

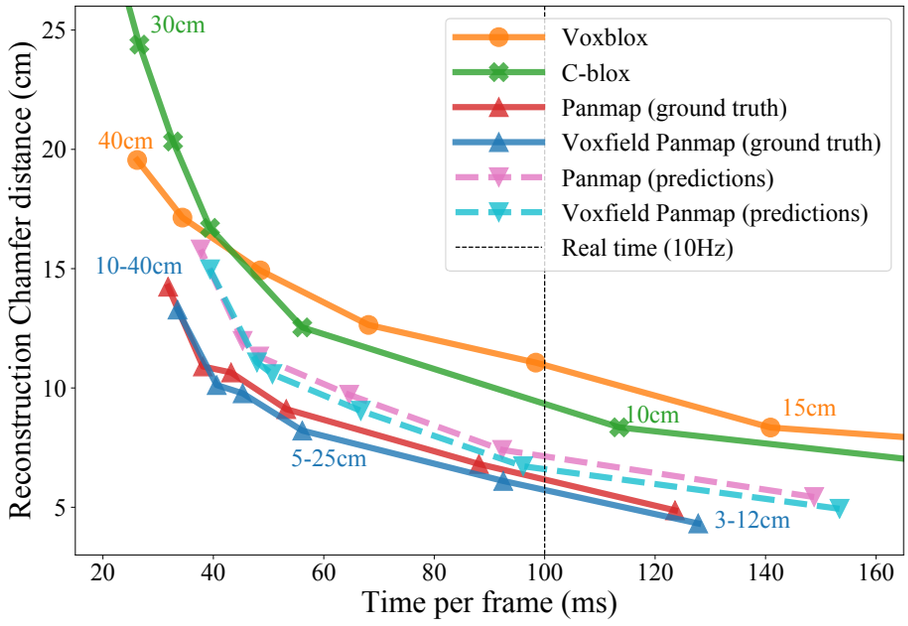
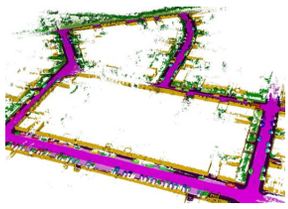
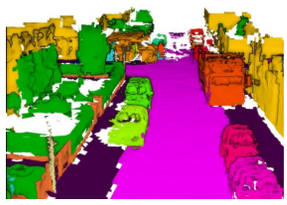


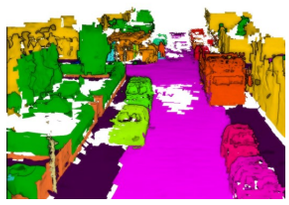
Figure 4.12: Reconstruction error (Chamfer distance) versus update time for different voxel sizes on the KITTI dataset. Each marker represents the result of one run. For the single-resolution mapping systems Voxblox and C-blox, we test voxel sizes ranging from 5 cm to 40 cm with an interval of 5 cm. 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.



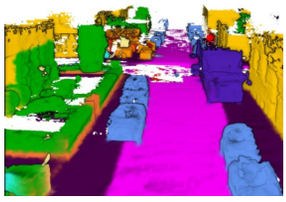
(a) Large-scale reconstruction by Voxfield Panmap



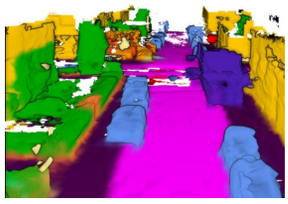
(b) A close-up of Voxfield Panmap's reconstruction



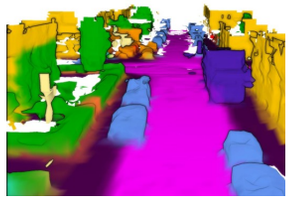
(c) A close-up of Panmap's reconstruction



(d) A close-up of Voxblox's reconstruction (voxel size = 10 cm)

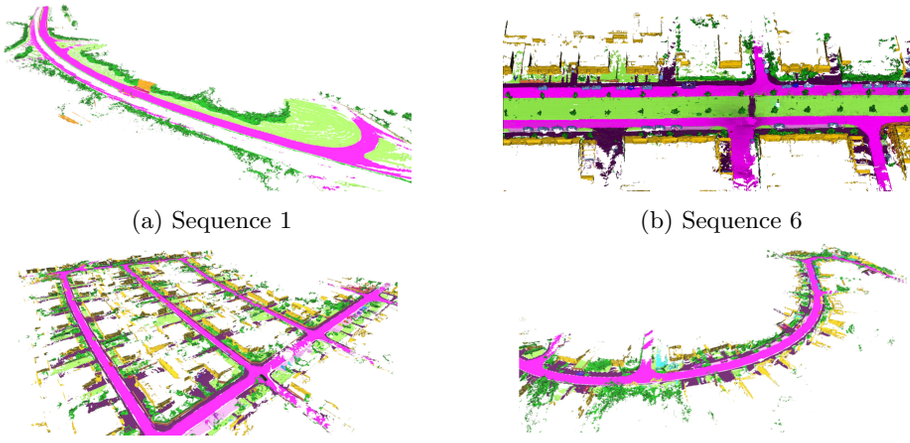


(e) A close-up of Voxblox's reconstruction (voxel size = 20 cm)



(f) A close-up of Voxblox's reconstruction (voxel size = 30 cm)

Figure 4.13: Qualitative results of the multi-resolution panoptic mapping (Panmap) reconstruction and the single-resolution voxblox reconstruction on the KITTI dataset. For the multi-resolution mapping, ground-truth panoptic labels and voxel sizes in the range of 5 cm–25 cm are used. For the single-resolution mapping, the semantic color is only used for better visualization.



(a) Sequence 1 (b) Sequence 6

(c) Sequence 8 (d) Sequence 9

Figure 4.14: Qualitative panoptic reconstruction results on the KITTI dataset by Voxfield Panmap. Ground truth panoptic labels are used. Different colors represent different panoptic labels.

4.4 Real-world Path-Planning Experiment

To verify Voxfield's usefulness as a mapping module for real-world path-planning applications, we conduct an indoor experiment with an MAV equipped with an Intel Core i5-1145G7 CPU and a RealSense D455 depth sensor, as shown in Fig. 4.15. State estimation by VINS-Fusion [64], TSDF and ESDF mapping with Voxfield and an optimization-based path planner [51] all run onboard the MAV. As shown in Fig. 1.1, the robot is able to plan a path and safely navigate to a goal position in a previously unknown environment, with the ESDF mapping conducted on-the-fly. Voxfield spends 1.9 ms and 22.7 ms on average for each ESDF map update with a 20 cm and 10 cm voxel size, respectively, indicating that our Voxfield can achieve real-time performance onboard the MAV.

Figure 4.15: The real-world path planning experiment is conducted onboard a MAV in an indoor environment with several obstacles in the middle.

Chapter 5

Summary

5.1 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 calculating 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 and safe navigation, even when the environment is previously unknown.

5.2 Future work

The current Voxfield takes the pose estimation that got output by an external sensor or estimator as an input to the system. In the future, it' better to avoid relying on an external pose estimator and also include the localization into our framework, as shown in Fig. 5.1. The localization module should consist of an odometry front-end and a pose graph optimization back-end. The odometry can be realized by the scan-to-ESDF map registration, where the grid-based registration algorithm normal distribution transform (NDT) [65] can be potentially used. A free-space feature based on local ESDF submap such as Freetrue [66] can be used for place recognition and loop closure detection so as to construct the pose graph. As the odometry drifts in time, a pose-graph optimization back-end such as Voxgraph [31] can be used to correct the accumulated error and improve the fidelity of the mapping process. Thanks to our Voxfield's superior performance on TSDF and ESDF mapping, the localization module based on the constructed ESDF map would also be more accurate and reliable.
Apart from the localization module, based on the Voxfield-empowered multi-resolution panoptic mapping, we can further try to use the segmentation information to do high-level tasks such as semantic-aware path planning [67] and real-time scene graph generation [68].

Another direction of future work would be exploiting the multi-robots collaborated volumetric mapping [69] based on Voxfield. The central topics would also be the SDF-based or mesh-based pose graph optimization and merging [38].

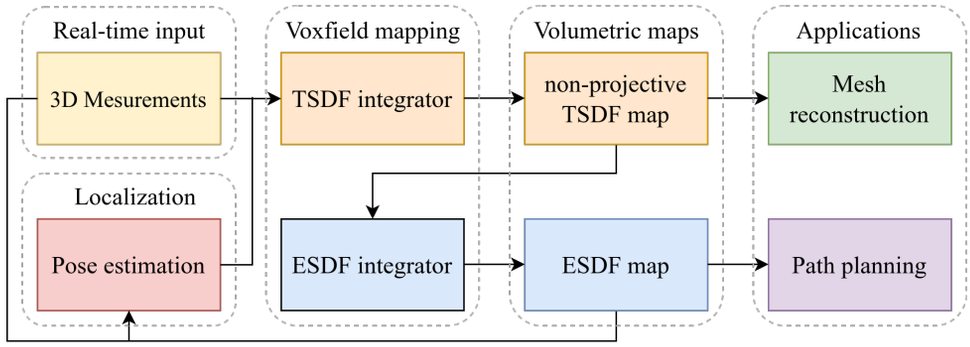


Figure 5.1: Future work of Voxfield on involving robot localization into the system

Bibliography

[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*, 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*, 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 *2017 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 *2019 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," *CoRR*, 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] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[8] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*, 2016.

[9] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "Mulls: Versatile lidar slam via multi-metric linear least square," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 633–11 640.

[10] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, vol. 2018, 2018, p. 59.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *IEEE international conference on robotics and automation (ICRA)*, 2014, pp. 15–22.

[12] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[13] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[14] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[16] L. Teixeira, I. Alzugaray, and M. Chli, "Autonomous aerial inspection using visual-inertial robust localization and mapping," in *Field and Service Robotics*. Springer, 2018, pp. 191–204.

[17] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "Octree-based fusion for realtime 3d reconstruction," *Graphical Models*, vol. 75, no. 3, pp. 126–136, 2013.

[18] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM siggraph computer graphics*, 1987.

[19] 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 *2011 10th IEEE international Symposium on Mixed and Augmented Reality*, 2011.

[20] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3d reconstruction using signed distance functions." in *Robotics: Science and Systems*, vol. 2, 2013, p. 2.

[21] 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*, 2015.

[22] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," 2012.

[23] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[24] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (ToG)*, 2013.

[25] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[26] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "Vdbfusion: Flexible and efficient tsdf integration of range sensor data," *Sensors*, vol. 22, no. 3, p. 1296, 2022.

[27] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1144–1151, 2018.

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

[29] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph." Robotics: Science and Systems, 2015.

[30] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[31] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, 2019.

[32] Y. Wang, N. Funk, M. Ramezani, S. Papatheodorou, M. Popović, M. Camurri, S. Leutenegger, and M. Fallon, "Elastic and efficient lidar reconstruction for large-scale exploration tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[33] E. Palazzolo, J. Behley, P. Lottes, P. Giguere, and C. Stachniss, "Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7855–7862.

[34] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 international conference on 3D vision (3DV)*, 2018, pp. 32–41.

[35] 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*, 2019.

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

[37] 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*, 2021.

[38] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems," *IEEE Transactions on Robotics*, 2022.

[39] 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*, 2022.

[40] J. Frey, H. Blum, F. Milano, R. Siegwart, and C. Cadena, "Continual learning of semantic segmentation using complementary 2d-3d data representations," *arXiv preprint arXiv:2111.02156*, 2021.

[41] E. Vespa, N. Funk, P. H. Kelly, and S. Leutenegger, "Adaptive-resolution octree-based volumetric SLAM," in *2019 International Conference on 3D Vision (3DV)*, 2019.

[42] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. F. Alcantarilla, and S. Leutenegger, "Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning," *IEEE Robotics and Automation Letters*, 2021.

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

[44] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 165–174.

[45] S. Weder, J. Schonberger, M. Pollefeys, and M. R. Oswald, "Routedfusion: Learning real-time depth map fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4887–4897.

[46] S. Weder, J. L. Schonberger, M. Pollefeys, and M. R. Oswald, "Neuralfusion: Online depth fusion in latent space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3162–3172.

[47] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu, "Di-fusion: Online implicit 3d reconstruction with deep priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 8932–8941.

[48] S. Lionar, L. Schmid, C. Cadena, R. Siegwart, and A. Cramariuc, "Neuralblox: Real-time neural representation fusion for robust volumetric mapping," in *2021 International Conference on 3D Vision (3DV)*, 2021, pp. 1279–1289.

[49] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE international conference on robotics and automation (ICRA)*, 2011, pp. 4569–4574.

[50] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. of IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2016, pp. 5332–5339.

[51] 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*, 2018.

[52] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[53] S. Scherer, D. Ferguson, and S. Singh, "Efficient c-space and cost function updates in 3d for unmanned aerial vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2049–2054.

[54] N. Kalra, D. Ferguson, and A. Stentz, "Incremental reconstruction of generalized voronoi diagrams on grids," *Robotics and Autonomous Systems*, vol. 57, no. 2, pp. 123–128, 2009.

[55] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of Euclidean distance maps and Voronoi diagrams," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[56] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1, pp. 189–214, 2012.

[57] H. Cover, S. Choudhury, S. Scherer, and S. Singh, "Sparse tangential network (spartan): Motion planning for micro aerial vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2820–2825.

[58] C. Sommer, L. Sang, D. Schubert, and D. Cremers, "Gradient-SDF: A Semi-Implicit Surface Representation for 3D Reconstruction," *CoRR*, 2021.

[59] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2012.

[60] 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.

[61] 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.

[62] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

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

[64] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, 2018.

[65] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2003, pp. 2743–2748.

[66] A. J. Millane, H. Oleynikova, C. Lanegger, J. Delmerico, J. Nieto, R. Siegwart, M. Pollefeys, and C. C. Lerma, "Freetures: Localization in signed distance function maps," *IEEE Robotics and Automation Letters*, 2021.

[67] L. Bartolomei, L. Teixeira, and M. Chli, "Perception-aware path planning for uavs using semantic segmentation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5808–5815.

[68] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization," *arXiv preprint arXiv:2201.13360*, 2022.

[69] L. Bartolomei, M. Karrer, and M. Chli, "Multi-robot coordination with agent-server architecture for autonomous navigation in partially unknown environments," in *proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 1516–1522.

Appendix A

Experiment data sheet

Table A.1: Comparison of the TSDF mapping error on the Flat synthetic RGB-D dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	1.49	3.28	4.58	6.36	8.87
Voxfield (Ours)	**1.06**	**2.41**	**3.65**	**5.14**	**7.24**

Table A.2: Comparison of the reconstruction Chamfer L1 distance on the Flat synthetic RGB-D dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	1.74	3.40	5.31	7.18	**9.04**
Voxfield (Ours)	**1.65**	**3.26**	**5.07**	**6.82**	9.08

Table A.3: Comparison of the reconstruction coverage on the Flat synthetic RGB-D dataset. The unit is %. A larger value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	75.58	80.41	82.85	85.31	88.23
Voxfield (Ours)	**77.02**	**82.08**	**84.15**	**87.46**	**90.49**

Table A.4: Comparison of the ESDF mapping error on the Flat synthetic RGB-D dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	3.59	5.96	6.49	8.33	11.50
FIESTA	2.10	4.05	4.73	5.80	9.68
EDT	2.09	4.21	4.84	6.07	9.82
Voxfield (Ours)	**1.38**	**2.66**	**3.97**	**5.11**	**6.50**

Table A.5: Comparison of the ESDF mapping time on the Flat synthetic RGB-D dataset. The unit is ms. A smaller value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	347.9	44.2	14.7	8.1	4.6
FIESTA	109.2	17.5	7.0	2.8	2.1
EDT	127.9	19.6	7.8	3.4	2.6
Voxfield (Ours)	**86.7**	**13.7**	**6.0**	**2.4**	**1.5**

Table A.6: Comparison of the TSDF mapping error on the Cow&Lady real-world RGB-D dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	2	5	10	15	20	25
Voxblox	2.55	4.72	7.00	8.88	10.23	11.71
Voxfield (Ours)	**2.17**	**4.12**	**6.14**	**7.70**	**8.94**	**10.03**

Table A.7: Comparison of the reconstruction Chamfer L1 distance on the Cow&Lady real-world RGB-D dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	2	5	10	15	20	25
Voxblox	1.82	4.23	7.53	**9.78**	**12.15**	**14.49**
Voxfield (Ours)	**1.81**	**4.14**	**7.40**	9.82	12.24	14.56

Table A.8: Comparison of the reconstruction coverage on the Cow&Lady real-world RGB-D dataset. The unit is %. A larger value is preferred.

Voxel size (cm)	2	5	10	15	20	25
Voxblox	73.51	76.67	77.57	78.84	79.35	80.08
Voxfield (Ours)	**74.74**	**77.41**	**78.19**	**79.63**	**79.80**	**80.87**

Table A.9: Comparison of the ESDF mapping error on the Cow&Lady real-world RGB-D dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	4.29	8.54	10.88	13.45	15.40
FIESTA	3.94	7.29	9.72	11.99	14.07
EDT	3.96	7.47	9.69	12.30	14.28
Voxfield (Ours)	**3.56**	**6.89**	**9.07**	**11.46**	**13.51**

Table A.10: Comparison of the ESDF mapping time on the Cow&Lady real-world RGB-D dataset. The unit is ms. A smaller value is preferred.

Voxel size (cm)	5	10	15	20	25
Voxblox	198.5	31.4	11.9	5.9	3.3
FIESTA	84.0	11.2	4.8	2.3	1.3
EDT	125.9	14.8	7.1	2.7	1.7
Voxfield (Ours)	**64.9**	**10.8**	**4.5**	**2.1**	**1.2**

Table A.11: Comparison of the TSDF mapping error on the MaiCity synthetic LiDAR dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	10	15	20	25	30	35	40
Voxblox	4.37	7.83	11.80	16.49	19.24	20.92	22.35
Voxfield (Ours)	**1.50**	**2.77**	**4.05**	**6.47**	**7.45**	**9.47**	**10.49**

Table A.12: Comparison of the reconstruction Chamfer L1 distance on the MaiCity synthetic LiDAR dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	10	15	20	25	30	35	40
Voxblox	4.37	6.49	8.51	**11.20**	**12.90**	15.00	16.30
Voxfield (Ours)	**4.05**	**6.35**	**7.95**	11.37	13.19	**14.45**	**16.24**

Table A.13: Comparison of the reconstruction coverage on the MaiCity synthetic LiDAR dataset. The unit is %. A larger value is preferred.

Voxel size (cm)	10	15	20	25	30	35	40
Voxblox	98.42	98.61	98.79	98.92	99.01	99.34	99.65
Voxfield (Ours)	**98.89**	**99.04**	**98.94**	**99.16**	**99.24**	**99.52**	**99.76**

Table A.14: Comparison of the ESDF mapping error on the MaiCity synthetic LiDAR dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	15	20	25	30	35	40
Voxblox	11.83	17.56	21.79	26.10	28.39	30.08
FIESTA	**5.52**	10.46	14.13	17.80	18.60	18.92
EDT	5.75	10.95	14.45	18.06	19.10	19.80
Voxfield (Ours)	6.17	**9.21**	**11.79**	**14.10**	**16.26**	**18.21**

Table A.15: Comparison of the ESDF mapping time on the MaiCity synthetic LiDAR dataset. The unit is ms. A smaller value is preferred.

Voxel size (cm)	15	20	25	30	35	40
Voxblox	2793.1	1254.9	685.0	423.2	332.8	283.6
FIESTA	1910.3	671.4	422.1	284.7	202.7	137.9
EDT	1502.3	901.7	449.3	261.8	195.7	121.6
Voxfield (Ours)	**1295.5**	**619.4**	**351.6**	**206.3**	**160.3**	**101.7**

Table A.16: Comparison of the TSDF mapping error on the KITTI real-world LiDAR dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	10	15	20	25	30	35	40	45
Voxblox	7.67	11.16	14.26	17.28	19.82	22.91	26.14	29.24
Voxfield (Ours)	**5.47**	**7.61**	**9.13**	**11.74**	**14.02**	**16.33**	**18.42**	**20.75**

Table A.17: Comparison of the reconstruction Chamfer L1 distance on the KITTI real-world LiDAR dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	10	15	20	25	30	35	40	45
Voxblox	6.35	8.34	11.06	**12.64**	14.93	**17.14**	**19.55**	**21.87**
Voxfield (Ours)	**6.13**	**8.04**	**10.96**	12.85	**14.86**	17.60	19.60	22.27

Table A.18: Comparison of the reconstruction coverage on the KITTI real-world LiDAR dataset. The unit is %. A larger value is preferred.

Voxel size (cm)	10	15	20	25	30	35	40	45
Voxblox	92.00	94.27	95.88	96.89	96.86	97.44	97.70	97.79
Voxfield (Ours)	**93.28**	**95.87**	**96.86**	**97.62**	**97.88**	**98.17**	**98.40**	**98.61**

Table A.19: Comparison of the ESDF mapping error on the KITTI real-world LiDAR dataset. The unit is cm. A smaller value is preferred.

Voxel size (cm)	25	30	35	40	45
Voxblox	24.30	28.52	32.73	36.81	40.97
FIESTA	17.06	20.02	22.70	25.55	28.48
EDT	17.22	20.24	23.00	25.96	28.89
Voxfield (Ours)	**15.01**	**17.59**	**20.22**	**22.57**	**25.13**

Table A.20: Comparison of the ESDF mapping time on the KITTI real-world LiDAR dataset. The unit is ms. A smaller value is preferred.

Voxel size (cm)	25	30	35	40	45
Voxblox	877.4	518.3	384.4	256.3	186.7
FIESTA	473.1	250.0	166.2	112.3	86.9
EDT	468.0	258.3	176.0	121.6	94.9
Voxfield (Ours)	**425.7**	**227.8**	**144.6**	**96.4**	**74.7**