Multi-Resolution 3D Mapping with Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning

Nils Funk[1,2], Juan Tarrio[2], Sotiris Papatheodorou[1], Marija Popović[1], Pablo F. Alcantarilla[2], Stefan Leutenegger[1,2]

***Abstract*— With the aim of bridging the gap between high quality reconstruction and mobile robot motion planning, we propose an efficient system that leverages the concept of adaptive-resolution volumetric mapping, which naturally integrates with the hierarchical decomposition of space in an octree data structure. Instead of a Truncated Signed Distance Function (TSDF), we adopt mapping of occupancy probabilities in log-odds representation, which allows to represent both surfaces, as well as the entire free, i.e. observed space, as opposed to unobserved space. We introduce a method for choosing resolution -on the fly- in real-time by means of a multi-scale max-min pooling of the input depth image. The notion of explicit free space mapping paired with the spatial hierarchy in the data structure, as well as map resolution, allows for collision queries, as needed for robot motion planning, at unprecedented speed. We quantitatively evaluate mapping accuracy, memory, runtime performance, and planning performance showing improvements over the state of the art, particularly in cases requiring high resolution maps.**

I. Introduction

The past years have brought impressive advancements in the field of dense environment mapping, fueled by the advent of RGB-D cameras and ever more powerful processors, including GPUs. Applications of (near-)real-time 3D dense mapping systems are vast, ranging from digital twins and Augmented/Virtual Reality to mobile robotics. Recent technological advances are inciting the use of mobile robots for many exploration and monitoring tasks. Their growing versatility and autonomy offer safe, cost-effective solutions in a wide range of applications, including aerial surveillance, infrastructure inspection, and search and rescue [1]. However, to fully exploit their potential, a key task is enabling light-weight platforms to operate autonomously in unknown, unstructured environments with limited on-board computational resources. In the real-world, map representations are thus required that can accommodate both high-fidelity reconstructions of the environment as well as perform fast online planning.

There are several efficient mapping methods for motion planning that rely on 3D volumetric representations. A common strategy is to use probabilistic occupancy maps using octree structures for quick access [3], [4]. More recently, signed-distance-based frameworks [2], [5], [6] have become popular as they allow fast map query operations in online settings. The main drawback of previous approaches is that their computational performance degrades drastically with the discretisation of the environment, since they operate on map data in the same way regardless of the occupancy status and geometry of the underlying space. As a result, these methods are only suitable for scenarios requiring coarse reconstructions or navigating in areas with relatively large obstacles. In contrast, our work addresses the challenge of trading off mapping accuracy against computational efficiency for planning in online, on-board robotic applications that require *both*.

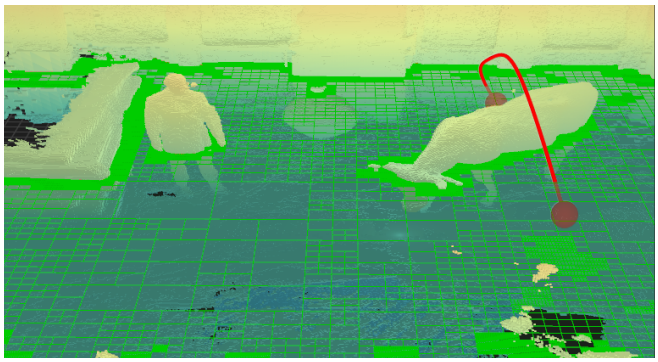


Fig. 1. The main goal of our system is to provide a navigable 3D occupancy map with a defined notion of observed free space in small to large scale scenarios, where the use of adaptive resolution is key to constraining memory consumption. The figure shows the output of our mapping pipeline for the *Cow and Lady* RGB-D dataset [2] alongside a planned trajectory (red). The chosen voxel resolutions for free space encoding are shown for a map slice.

In this paper, we introduce a volumetric adaptive-resolution *dense* mapping framework that supports multi-resolution queries and data integration using occupancy mapping [7], [8], [3]. Contrary to methods based on signed distance functions, we continuously maintain a high resolution 3D octree representation of observed occupied and *free* space in real-time.

Our key insight is to recognise the lack of a concise method for defining required resolution in the context of occupancy mapping, with few approaches tackling this problem by extending single resolution approaches with ad-hoc heuristics [4]. As a result, we designed a novel integration algorithm that selects resolution by constraining the induced sampling error in the observed occupancy, splitting an octree in a coarse-to-fine fashion until a desired accuracy is reached.

[1] Smart Robotics Lab, Imperial College London, UK {nils.funk13, s.papatheodorou18, m.popovic, s.leutenegger}@imperial.ac.uk

[2] SLAMcore Ltd., London, UK {first.name}@slamcore.com

The video is available at: https://youtu.be/XlnO3GBNPvc

Central to this idea is the introduction of a multi-scale max-min pooling of the input depth image which enables real-time operation by providing a conservative indication of measured depth variation in any given volume with only a few queries.

To further enhance performance, we retained the data structure of [6] which uses mip-mapped voxel blocks, but carefully designed a new scale selection and data propagation scheme between levels. This allowed us to increase computational and memory efficiency without introducing reconstruction artefacts.

While our method is focused on RGB-D mapping, it is flexible to different sensor modalities. This is shown by evaluating our system in a wide range of datasets, from synthetic to large scale LIDAR, showing significant improvements in reconstruction accuracy, runtime performance, and planning performance against state-of-the-art approaches.

In summary, the main contributions of this paper are:

1) A dense volumetric multi-resolution online mapping system that consistently represents free and occupied space to fine resolutions where needed; we will release our reference implementation open-source upon acceptance of this work.
2) A novel fast octree-to-image allocation and integration method, that seamlessly adapts the map to different scene scales and sensor modalities.
3) A comprehensive evaluation with respect to the state-of-the-art revealing vast improvements in the trade-offs of mapping accuracy, speed, memory consumption, tracking accuracy, and planning performance.

II. Related Work

A large body of literature addresses the problem of obtaining a dense 3D representation of the world. In this section, we overview recent work, focusing on volumetric reconstruction methods suitable for real-time robotic applications running on constrained hardware. This leaves aside most batch integration methods, where a map is only available for planning at the end of the mapping run; note that most newer deep learning based methods fall in this category.

A major milestone in on-line RGB-D 3D reconstruction systems is the KinectFusion algorithm of [9], which enables dense volumetric modelling in real-time at sub-centimetre resolutions. However, it is limited to small, bounded environments, as the mapping is locked to a fixed volume with a predefined voxel resolution, and requires GPGPU processing to achieve real-time performance. To improve scalability, several extensions to the original algorithm have been proposed. One possibility is to use moving fixed-size sliding volumes [10], [11] to achieve mapping in a dynamically growing space. Another strategy is to exploit memory-efficient data structures, such as octree-based voxel grids [4], [12], [13] or hash tables [14], [15], for quicker spatial indexing. More recently, deep learning methods tackling this problem in an incremental fashion are also being introduced [16]. Despite this impressive progress, most research in 3D mapping has targeted the application of surface reconstruction, in which the objective is to produce a high-quality mesh/point-cloud of a given scene. From a navigation perspective, the concept of observed free space is equally or more important than surface accuracy. Unfortunately most of these methods model free space only in the vicinity of surface boundaries, making them incapable of generally distinguishing between free and unvisited areas in initially unknown environments. Our work builds on the concept of explicit distinction between the two as necessary for robotic planning, exploration, and collision avoidance.

In the context of navigation, Euclidean Signed Distance Field (ESDF)-based mapping methods are commonly used for motion planning tasks as they provide distance information for trajectory optimisation strategies. Recently, significant work has been done on incrementally building ESDF maps for planning in 3D using aerial robots, including the *voxblox* [2] and *FIESTA* [5] frameworks, which construct ESDF maps from Truncated Signed Distance Field (TSDF) maps and occupancy maps, respectively. However, as these methods are designed for fast on-board collision checking, they rely on coarsely disretised environments with voxel grid resolutions on the order of $\sim 20\,\text{cm}$ magnitude. In contrast, our approach is also motivated by applications like close-up inspection [17], where more detailed scene reconstructions are required.

An alternative representation for planning is the occupancy map [7], [8]. In 3D, OctoMap [3] is a popular framework that uses hierarchical octrees to track occupancy probabilities as sensor data is received. Similar to the original work of [7], it uses an inverse sensor model formulation that efficiently approximates the posterior using an additive log-odds update equation, which resembles the TSDF update procedure [9] in its nature. However, while OctoMap works decently with LIDAR at low resolutions, its performance degrades significantly as map resolution increases, as well as in the presence of noisier sensors. Interestingly, a very recent contribution shows improvements over OctoMap in terms of memory and run time by adaptively downsampling the pointcloud and integrating free space at lower resolutions [18], highlighting the importance of this topic to this day. While this method uses a set of distance based rules to decide the integration resolution (similar to [4]), leading in the fastest setups to non-conservative assignments of free space, our work tackles this problem by rigorously assessing the probabilistic (inverse) measurement model, introducing a methodology for choosing a sampling resolution that ensures these errors do not happen, while also improving run time performance.

Closest to our work and with the aim of providing an unified framework for trajectory reconstruction and planning, [4] proposed an efficient pipeline with an octree-based implementation. Their approach is suitable for either TSDF-based or occupancy mapping based on the spline inverse sensor model of [19]. However, the use of multi-resolution is very basic and multiple assumptions are made which limit the applicability to planning scenarios. Subsequent work [6] extended this system to handle data integration with varying levels of detail and rendering at multiple resolution scales

using TSDF maps, but limited to surface reconstruction. Our method draws some inspiration from this approach in terms of data structure and propagation; however, our focus is on volumetric occupancy-based representations for planning and space understanding, where the main goal is to have a probabilistic classification of all observed space between occupied, free, and unknown at high resolutions, while providing at the same time a high quality (surface) reconstruction of the environment.

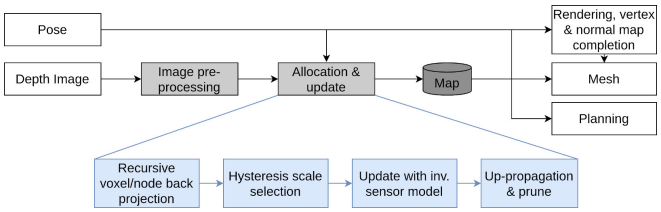


Fig. 2. Overview of our system: The different stages of the mapping pipeline are shown in grey, while the blue boxes show the steps in the *allocation and updating* procedure. The flow of information is illustrated by the arrows. The allocation and updating stages modify the map, while the rendering and planning stages utilise the map information.

III. Multi-Resolution Occupancy Mapping

Our library takes as input depth information and poses, incrementally computing a map which can be queried at any time for path planning, meshing, and rendering (see Fig. 2). We also provide an Iterative Closest Point (ICP) module that can be switched on to obtain a full Simultaneous Localisation and Mapping (SLAM) system in the spirit of [9], as shown in section IV-D.

Our data structure is an extension of [4] and is composed of an octree where the last levels consist of densely allocated voxel blocks. Similarly to [6], each voxel block stores a pyramid representation of itself enabling fast occupancy updates at different resolutions. However, unlike [6], we only allocate data in each block down to a single dynamically changing integration scale s_c. More importantly, we augment this method by also allowing data to be stored at node level when the voxel block resolution is not needed. This two tier memory structure leverages flexibility in choosing resolution at the higher (octree) levels with efficient access at lower (voxel block) level. A visualisation of this structure's allocation can be seen in Fig. 1 for a practical use case.

By design, relevant occupancy information for a given point is maintained only at the lowest allocated voxel which contains the point. Similar to [3], upper non-leaf nodes store a max pooling of the children occupancy, enabling fast conservative queries of any given size. In addition to the max occupancy, we keep a Boolean state accounting for the presence of unknown data in the children, this allows us to disambiguate a node being partially or fully unobserved. In our system this pooling, or data up-propagation, is computed in each integration step, making it available for on-line planning at any given time.

A. Notation

We denote n-dimensional vectors with lower-case, bold letters, e.g. $\mathbf{x} \in \mathbb{R}^n$. Also, we denote the coordinate frame in which vectors are expressed with left subscripts, e.g. ${}_C\mathbf{x}$. We employ a World frame $\{W\}$ and the Camera frame $\{C\}$. Euclidean transformations from coordinate frame $\{C\}$ to $\{W\}$ are denoted as $\mathbf{T}_{WC} \in \mathrm{SE}(3)$. We denote matrices with upper-case bold letters, e.g. $\mathbf{D} \in \mathbb{R}^{n \times m}$.

B. Occupancy Map Fusion

Given a depth image $\mathbf{D}_k \in \mathbb{R}^{H \times W \times 1}$ and camera pose $\mathbf{T}_{WC_k}$ at time step k, the probability that a 3D point ${}_W\mathbf{p}$ in space is occupied is assumed to be a function of the distance to the camera and the corresponding depth measurement along the ray from the camera centre ${}_W\mathbf{c}$ to ${}_W\mathbf{p}$. Given the depth measurement from the projection of point ${}_W\mathbf{p}$ into the camera:

$$z\left({}_W\mathbf{p}, \mathbf{D}_k, \mathbf{T}_{WC_k}\right) = \mathbf{D}_k\left[\pi({\mathbf{T}_{WC_k}^{-1}}\,{}_W\mathbf{p})\right], \tag{1}$$

from one depth image we would like to represent the occupancy probabilities in 3D space:

$$P_{\mathrm{occ}}\left({}_W\mathbf{p} \,|\, \mathbf{D}_k, \mathbf{T}_{WC_k}\right) = P_{\mathrm{occ}}\left({}_W\mathbf{p} \,|\, z({}_W\mathbf{p}, \mathbf{D}_k, \mathbf{T}_{WC_k})\right), \tag{2}$$

which corresponds to the inverse sensor model, a function of the depth along the ray. For brevity, it is referred to as $P_{\mathrm{occ}}({}_W\mathbf{p}|z_k)$ in the following descriptions. An alternative way of representing the occupancy probability is using log-odds, which allows for Bayesian updates to be additive as:

$$l_k({}_W\mathbf{p}) = \log \frac{P_{\mathrm{occ}}({}_W\mathbf{p}|z_k)}{1 - P_{\mathrm{occ}}({}_W\mathbf{p}|z_k)}, \tag{3}$$

$$L_k({}_W\mathbf{p}) = L_{k-1}({}_W\mathbf{p}) + l_k({}_W\mathbf{p}). \tag{4}$$

In contrast to previous work, we do not accumulate the log-odds in a single sum, but instead use a weighted mean $\bar{L}_k^s({}_W\mathbf{p})$, with weight w_k^s, similar to how it is done in [9], with s denoting the selected scale. This allows reducing artefacts during the interpolation and propagation stage caused by neighboring voxels with significant differences in the number of observations. Moreover, by clamping the weight w_k^s below a threshold $w_{\max}$, the influence of outliers and dynamic objects is mitigated. Thus the mean log-odds is updated as:

$$\bar{L}_k^s({}_W\mathbf{p}) = \frac{\bar{L}_{k-1}^s({}_W\mathbf{p})\; w_{k-1}^s + l_k^s({}_W\mathbf{p})}{w_{k-1}^s + 1}, \tag{5}$$

$$w_k^s = \min\{w_{k-1}^s + 1, w_{\max}\}, \tag{6}$$

while the accumulated log-odds can be preserved:

$$L_k^s({}_W\mathbf{p}) = \bar{L}_k^s({}_W\mathbf{p})\; w_k^s. \tag{7}$$

C. Inverse Sensor Model

For fast computations, our inverse sensor model is a piece-wise linear function based on [19] and is illustrated in Fig. 3 (a) operating directly in log-odds space. Similarly to [19], we use a model where measured surface position matches with a log-odds value of zero.

We model depth uncertainty σ as a function of measured distance z_r (i.e. along the corresponding ray), assuming it to

be linearly or quadratically growing depending on the sensor type (see Fig. 3 (b)): quadratic for RGB-D and linear for LIDAR or synthetic (perfect) depth cameras. We relax the assumption of quadratic relation made in [19] between the surface thickness $\tau(z_r) \propto z_r^2$ with a linear model $\tau(z_r) \propto z_r$ bounded by a minimum and maximum surface thickness $\tau_{\min}$ and $\tau_{\max}$, to avoid overgrowing of distant objects.

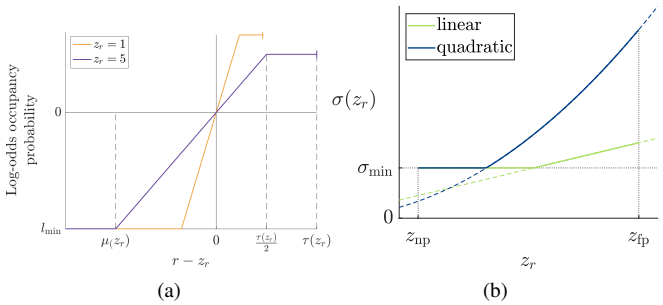


Fig. 3. (a) Inverse sensor model for two measurements (1 m / 5 m) expressed as a function of radial distance r from the camera. Log-odd values in front of the surface are clipped at $l_{\min}$ reached at $\mu = 3\sigma$ and grow linearly up to half the surface thickness $\tau(z_r)$. (b) Distance-dependent growth of two sensor uncertainty models (linear and quadratic) within the minimum and maximum sensor range z_{np} and z_{fp} with minimum sensor uncertainty $\sigma_{\min}$.

D. Adaptive-Resolution Volume Allocation

The data structure is interpreted as a non-uniform partition of a continuous occupancy 3D scalar field [4]. Consequently we do not assign any probabilistic meaning to the size of the voxel storing a particular value and instead aim at choosing a partition of the tree which can represent the underlying continuous function up to a certain accuracy.

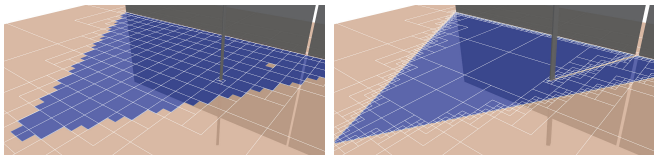


Fig. 4. Motivational example: Wall at 5m with pole at 3m. (**Left**) OFusion [4] - While obstacles are allocated at the finest scale s_0, the remaining free space is allocated at scale s_4. In contrast to our work, scale is chosen purely based on distance to measurement along a ray, thus ignoring variations of occupancy inside voxel's volume. As a result the frontiers to unknown space are non-conservative. Note the difference of the free space at the frustum boundaries and the unknown space behind the pole. (**Right**) Ours - The mapping strategy naturally integrates the volume at the appropriate scale, allocating obstacles and frontier boundaries to unknown space at fine scales, while representing free space at the coarsest possible scale. As a result our mapping strategy conservatively allocates free space while avoiding the overhead of allocating and updating the entire free space at the finest scale and pruning it afterwards [3].

Choosing this partition -on the fly- in a volumetric space is not a trivial task. Methods like *OctoMap* [3] use a ray-casting scheme to allocate densely the observed space, simplifying resolution in later stage called tree pruning. Newer methods like *OFusion* [4] and *UFOMap* [18] mitigate the need for dense allocation by using a set of heuristics to choose the needed resolution during the allocation step. The main problem with these approaches is illustrated in Fig. 4. The fact that resolution is chosen as part of the ray-casting and mainly as function of distance tends to ignore changes in occupancy that happen because of depth changes between pixels, which often require high resolution allocation to be accurately captured, leading to parts of space being erroneously labeled as free. This is particularly important in occlusions caused by thin objects and frustum boundaries.

To solve this problem we take a radically different approach, discarding the raycasting and using a so called 'map to camera' allocation and updating process, thereby following an 'as coarse as possible, as fine as required' mapping scheme.

Given a new depth image $\mathbf{D}_k$, we start from the root of the octree analysing each node recursivelly in order to decide whether the variation of occupancy log-odds inside the node meets a bounding criterion:

$$\max |l_k({}_W\mathbf{p}_i) - l_k({}_W\mathbf{p}_j)| < \varepsilon \tag{8}$$

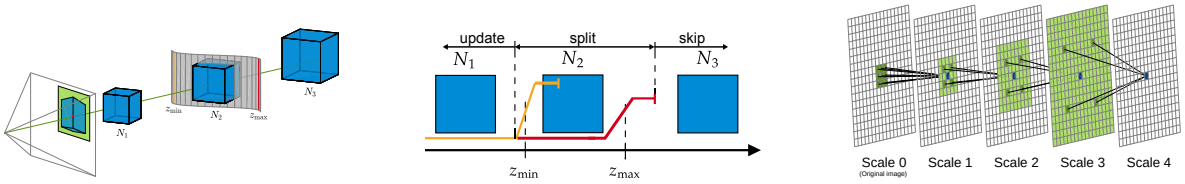
for every ${}_W\mathbf{p}_i, {}_W\mathbf{p}_j$ in the node volume. This criterion has been used for 3D data compression on an octree [20]. If it is met, we update the node at the given scale. Otherwise the node is split into its eight children, and the process is recursively repeated until (8) is fulfilled or voxel block level is reached.

In order to make this algorithm feasible in a real-time setting, an efficient method to evaluate (8) for a voxel of any given position and size has to be provided. We observe that, due to the particular structure of the inverse sensor model, the occupancy span inside a given voxel can be connected with the span of depth values inside the voxels' projection into the depth image, taking the problem from a 3D query to a 2D one. This is illustrated in Fig. 5 for three different cases.

More importantly, to quickly query the depth variation in a particular image region we pre-compute a max-min pooling image vector $\mathbf{S}_k(\mathbf{D}_k) \in \mathbb{R}^{H \times W \times f_{\max}+1}$ at the *pre-processing* stage. This pooling consists of $f_{\max}+1$ scales $f \in \{0, 1, \ldots, f_{\max}\}$ which aggregate the information of the original depth image for different square sizes $B^f \in \mathbb{R}^{n^f \times m^f}$ centred around each pixel in the image. The square areas for a single pixel at different scales are illustrated in Fig. 5 (c). Each pooling image pixel at scale f summarises the minimum and maximum depth measurement $z_{\min}$, $z_{\max}$ within B^f, as well as a validity state and an image crossing state to handle image boundaries and invalid data. Once this structure is computed, a conservative evaluation of (8) can be made by simply computing a bounding box of the voxel in the image and querying the max-min pooling at the level with the maximum square size that is still fully contained in the bounding box, reducing the amount of queries to no more than 4 in the majority of cases.

E. Multi-resolution Probabilistic Occupancy Fusion

While data at node level may be scattered at different resolutions, each voxel block is evaluated at a common resolution scale s_c, further improving performance, reducing aliasing effects and simplifying interpolation required by operations like raycasting and meshing. Similar to [6], measurements are integrated at a mip-mapped scale in the voxel block



(a) (b) (c)

Fig. 5. (a) Three nodes N_1-N_3 of different sizes and positions project into in the same bounding box (light green) in the image. The bounding box holds the depth measurements of the surface illustrated in grey. The minimum $z_{\min}$ and maximum $z_{\max}$ surface depth measurements of the bounding box are shown in yellow and red on the surface. (b) 2D view of the example in (a) to illustrate how depth variation is used together with the inverse sensor model to deduce a splitting decision based on the minimum and maximum measurement in the bounding box. Node N_1 is in front of the surface uncertainty of the minimum measurement and can be updated at the current scale, while node N_3 is behind the approximated wall thickness of the maximum measurement and will be skipped (neither allocated nor updated). Node N_2 crosses the surface boundary and must be split and its children re-evaluated. Additionally, the node will be split if it projects into partly unknown data or crosses the frustum. (c) Illustration of the first five scales of max-min pooling. At each scale, the centre pixel summarises the light green area (of the original image) by summarising only the four surrounding pixels at the previous scale.

based on the current distance of the block's centre from the camera. Additionally we integrate blocks that are known to only contain frontiers from free to unknown space not finer than scale $s_f \in [0,1]$.

As the distance changes, the desired integration scale s_d may change as well. In contrast to [6], we apply a scale change hysteresis requiring the camera to move a certain amount $\Delta_C \mathbf{m}_z$ closer to or further away from the voxel block before changing the scale again, to avoid constant scale changes of blocks located at a scale boundary.

We adapt the propagation strategies previously applied in [6], to our occupancy map representation to keep the hierarchy consistent between scale changes. As in [6], we replace the parent with the mean of all observed children when up-propagating information to a coarser mip-mapped scale ($s_d > s_c$). However, when down propagating to a finer integration scale ($s_d < s_c$), we allocate the data first and assign the parent's value to all its children. In either case, we do not change scale immediately. Instead we wait for the block to be fully projected into the image plane multiple times and observed at the desired scale s_d. During this time we update the data at both scales s_c and s_d. Once the scale changing condition is fulfilled we switch to the new integration scale, thus deleting the previous buffer. This logic reduces artefacts by smoothing values during initialisation and preventing blocks that are occluded or just about to exit the camera frustum from changing scale.

Once all depth information for an image $\mathbf{D}_k$ is integrated into the map, the mean values of all updated blocks are up-propagated to the mip-mapped scale within each block to enable tri-linear interpolation between neighbours of different scales [6]. Additionally, the maximum occupancy of all updated blocks and nodes is propagated through all levels up to the root of the tree to support fast hierarchical free space queries. Finally, pruning is applied to merge node children whose weighted log-odd occupancy is close enough according to a lower threshold $L_{\min}$. This extra step simplifies the tree when the occupancies converge to similar values.

F. Multi-resolution ray-casting and Meshing modules

Fast raycasting is required to render the surface in real-time and/or to enable tracking as part of a full dense SLAM system. With the up-propagated maximum occupancy and observed state stored in our data structure, we implemented a multi-resolution ray-casting strategy to quickly move a ray through large volumes of free space [20].

A meshing module is also provided which computes an adaptive resolution dual mesh following [21]. This approach had been adapted to support our two tier data structure.

IV. Evaluation

This section reports on our experimental results. All our tests were performed on a laptop with an Intel Core i7-8750H CPU operating at 2.20 GHz, 8 GB of memory and running Ubuntu 18.04. We used GCC 7.4.0 with OpenMP acceleration for software compilation. For computing the TSDF with *voxblox* [2] the *fast* method was used throughout the comparisons, *OctoMap* [3], *UFOMap* [18] and *voxblox* were run in ROS. The parameter values used in the experiments are shown in Table I unless otherwise specified.

$l_{\min,\text{total}} = 100$	$l_{\min,\text{iter}} = -5.015$	$w_{\max} = l_{\min,\text{total}}/l_{\min,\text{iter}}$
$\sigma_{\min} = v_{\text{res}}$	$\sigma_{\max} = 3 \times v_{\text{res}}$	$z_{\text{np}} = 0.4$ m $\quad z_{\text{fp}} = 6$ m
$\tau_{\min} = 3 \times v_{\text{res}}$	$\tau_{\max} = 12 \times v_{\text{res}}$	$s_f = 1$
Cow and Lady FR3 - Long Office ICL - LR2	$\sigma(z_r) = 0.0025 {z_r}^2$ $\sigma(z_r) = 0.05 z_r$	$\tau(z_r) = 0.05 z_r$ $\tau(z_r) = 0.05 z_r$

TABLE I

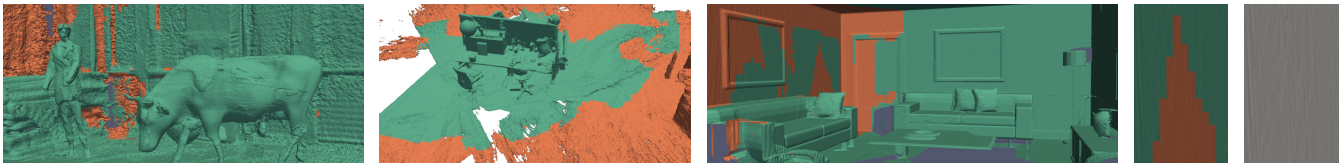
Parameter values used in the experiments.

Three widely known datasets were used to quantitatively evaluate our system, namely the *TUM* RGB-D [22], *Cow and Lady* [2] and *ICL-NUIM* [23].

A. Reconstruction Accuracy

To evaluate how our method performs in terms of surface reconstruction, we evaluate in the *ICL-NUIM Living Room 2* dataset [23]. Table II shows the reconstruction Root Mean Squared Error (RMSE) to the ground truth mesh as computed by the *SurfReg* tool provided by the dataset, with given poses and without extra ICP alignment. Our method outperforms previous approaches on both exercised resolutions.

Additionally we investigate the potential degradation induced by down-sampling the input image. It can be seen the effect in the reconstruction metric in all cases is minor,



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

Fig. 6. Our multi-scale mesh reconstruction for different sensor types with green encoding sampling scale 0 (finest) and orange scale 1. (a) In the *Cow and Lady* dataset and (b) *TUM* dataset, scale is adapted to reflect higher Kinect sensor noise, particularly for places far from sensor trajectory, which loops around the central desk. (c) In the *ICL-NUIM* synthetic dataset a similar model is used to show how, in lack of noise, no artefacts appear due to scale changes. (d) A close up in ICL showing an scale change in one of the walls. (e) Same as (d) but with uniform color, showing the lack of artefacts in the transition.

motivating the use of this later step for improving both running time and memory consumption.

To show how our method is able to adaptively select sampling resolution, multiple reconstructions are presented in Fig. 6 with color encoding scale.

	Image scale	1 cm	2 cm
Ours	×1	0.0146	**0.0193**
	×0.5	**0.0250**	**0.0341**
SE OFusion	×1	0.0166	0.0259
	×0.5	0.0478	0.0465
SE TSDF	×1	**0.0142**	0.0200
	×0.5	0.0429	0.0367

TABLE II

RECONSTRUCTED MESH RMSE [M] ON THE *ICL-NUIM LR2* DATASET FOR DIFFERENT RESOLUTION AND IMAGE SUBSAMPLING.

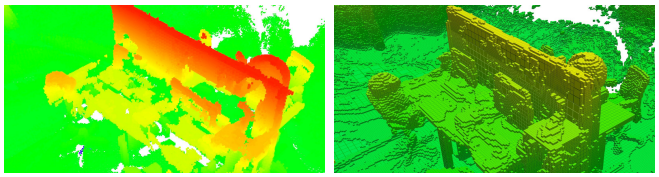


Fig. 7. Voxelised reconstruction in FR3 - Long Office for *UFOMap* ($n = 0, d \hat{=} 16cm$, their visualiser) (**left**) and our system (**right**). Note the holes in the *UFOMap* surface due to non-conservative free space allocation.

B. Runtime Performance

Running times for our system and competing methods can be seen in Fig. 8. Our method achieves the highest relative performance at 1cm resolution for both image configurations, beating all other methods in datasets using real sensors. While at 2cm most methods fall in the same range, as resolution decreases to 8cm the relative cost of computing the pooling image becomes significant, making OFusion significantly faster. This is expected as the main goal of the pooling is to boost performance in high resolution cases were the algorithm is targeted to. More importantly, speed achieved by OFusion is in many cases driven by non-conservative assumptions that render the output unsuitable for navigation, as shown in section IV-E. A similar issue happens in *UFOMap*, where the *fast* setup achieves excellent speeds, but severely compromises the quality of the output map, as shown in Fig. 7. Our system is able to overcome these issues while providing similar or faster performance.

Other approaches like *voxblox* present reasonable results, but have the drawback of requiring an extra step of computing an ESDF in order to allow direct planning on them, which becomes prohibitively expensive in high resolution cases. This limits its usability in cases requiring a high resolution navigable map in real-time. Finally *OctoMap* presents significantly higher integration times than newer methods, since it targets low resolution LIDAR made maps.

C. Multi-resolution, Memory and Efficiency

To assess memory efficiency of our multi-resolution approach, we compare our memory usage to supereight OFusion [4] and supereight MutiresTSDF [6]. In particular OFusion uses an octree, similarly to our system for storing both free and occupied space, while MultiresTSDF also utilises an octree but for sparsely allocating a narrow-band TSDF. As a conservative measurement of memory usage, we use the main process' Resident Set Size (RSS) as reported by the Linux kernel to compute the RAM usage.

Results can be seen in Table III. For the intended use case of high resolution reconstruction and planning, our system overcomes competing methods in real life scenarios, mainly through gains made by the adaptive sampling resolution at voxel block level. As resolution lowers, or in small datasets like the *ICL-NUIM* where the relative size of voxel block grows in a way that brings the allocation near to dense, the cost of storing this extra multi-resolution information together with our conservative allocation of frustum boundaries, takes a toll on the usage with respect to simpler methods like OFusion. This extra cost is justified by superior planning performance.

	Cow & Lady		FR3 Long Office		ICL LR2	
	1cm	8cm	1cm	8cm	1cm	8cm
Ours	**1100**	65	**493**	62	397	105
SE OFusion	1734	**57**	1009	**56**	**192**	**54**
SE MultiresTSDF	2922	65	1728	68	406	56

TABLE III

MEMORY CONSUMPTION IN MB AT DIFFERENT VOXEL RESOLUTIONS.

D. Tracking Performance

In this section we evaluate our mapping system integrated to a Dense SLAM Pipeline similar to [9], which is included

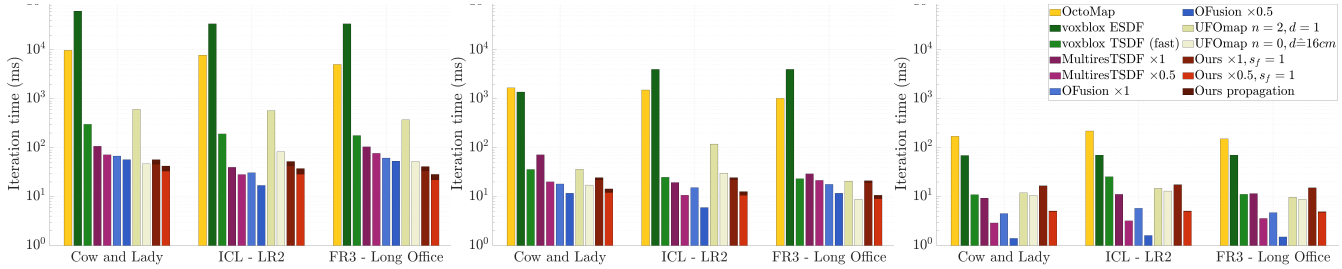


Fig. 8. Timings for our system alongside other planning libraries, including the widely used *OctoMap* [3], for 1cm (**left**), 2cm (**centre**) and 8cm (**right**) maximum resolution. Notice the logarithmic scale. In the case of *voxblox* both timings for TSDF and ESDF calculation are shown, as both are needed for path planning. For some libraries including ours, results are presented for full resolution input images (×1) and down-sampled images (×0.5). The cost for computing data up-propagation (pooling) in our method is also separated to highlight the cost of having this multi-resolution feature in our map.

as an additional component of the presented library.

Table IV shows trajectory accuracy results using the Absolute Trajectory Error (ATE) metric against TSDF and OFusion [4] methods on the FR1 - Desk and FR3 - Long Office sequences of the *TUM* RGB-D datasets. The same ICP tracking approach presented in [9] is used for all pipelines. For point-cloud extraction our system relies on the efficient multi-resolution raycasting method described in section III-F. Results indicate that maps produced by our approach are suitable for accurate ICP tracking, with the ATE being similar to that of previous methods.

	ATE (m)			
	FR1 - Desk		FR3 - Long Office	
Pipeline	1 cm	2 cm	1 cm	2 cm
Ours	**0.093**	0.098	0.209	0.239
SE TSDF	0.099	0.103	0.314	FAIL
SE OFusion	0.100	**0.086**	**0.165**	**0.172**

TABLE IV

ATE OF VARIOUS PIPELINES ON *TUM* RGB-D DATASETS.

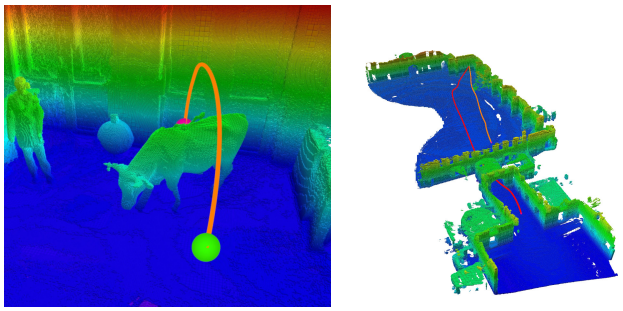


Fig. 9. (**left**) Planning in the *Cow and Lady* dataset : Trajectory 1 over the cow obtained with our mapping system, OFusion is unable to solve this planning problem due to clutter. (**right**) Planning in The Newer College dataset [24]: Trajectores 3 (**orange**) and 4 (**red**). Remarkably the method is able to find the shortest path though the centre corridor.

E. Planning

Finally, we evaluate our mapping system as input for path planning. We show that kinodynamically feasible and collision free quadrotor trajectories with map resolutions up to 1 cm can be planned in real-time. To guarantee feasibility, we compute a Safe Flight Corridor (SFC) from the start to the end position and optimise 10^{th} order Bernstein polynomial motion primitives within each segment. As a corridor primitive we use cylinders connected by spheres with a minimum radius r_{min}. We use the open motion planning library's (OMPL [25]) informed rapidly-exploring random tree* (informed RRT* [26]) planner to create the SFC connecting the start and end positions. OMPL confirms the safety of each corridor segment by verifying that none of the cylinder and sphere volumes is occupied. This is challenging using a regular volumetric grid where the number of checks grows cubically with the map resolution.

With our multi-resolution maximum occupancy queries, we utilise a 'coarse-to-fine' collision checking approach recursively increasing the resolution in parts of the corridor where needed, thus requiring significantly less checks in most cases.

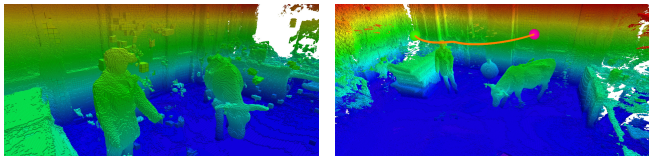


Fig. 10. (**left**) Clutter in OFusion makes the planner fail in several cases; (**right**) For a different problem (Trajectory 2) both systems find a solution but OFusion struggles with noise while planning times with our pipeline are significantly lower.

We investigate the use of our method for several path planning problems illustrated in Figs. 9 and 10 and compare against the OFusion library, which also achieves excellent mapping times and provides some multi-resolution output. While our method is able to find suitable trajectories in every iteration, notably in cluttered datasets like the *Cow and Lady* or large scale like *The Newer College* [24], OFusion fails in several cases. The reason why this happens is due to OFusion lacking a proper mechanism for leveraging information from different scales, instead relying on a simple rule of giving preemptiveness to data in high resolution voxel blocks over values stored at node levels. With the later normally storing information about free space observations and the former allocating surface measurements, the consequence is an algorithm that is fast but heavily biased against noise measurements and outliers. In our system we solve this issue

by not only propagating data down from nodes to voxel blocks, but also by carefully considering the variation of occupancy inside the node's whole volume (not only the centre value), as described in section III-D, to avoid having the opposite effect of biasing towards free space.

As an exercise to further confirm these aspects and highlight the benefits of our library, we modified the OFusion library to perform dense allocation, i.e. everything integrated at a single lowest voxel block level. The results can be seen in Table V which shows the minimum required solving time for the planner to find a SFC. While the dense allocation approach is able to remove OFusion noise and find suitable trajectories, the planning time required is significantly worse. This is easily explained by the lack of multi-resolution sampling capabilities, a key feature of our system.

	Traj 1	Traj 2	Traj 3	Traj 4
Ours	**0.01**	**0.1**	**0.06**	**0.4**
SE OFusion	4	FAIL (too noisy)	3	FAIL (too noisy)
SE Dense	5	15	3	6

TABLE V
MINIMUM TIMES (SEC) TO COMPUTE 'SAFE FLIGHT CORRIDOR'.

V. CONCLUSION

We have introduced a multi-resolution 3D mapping framework that is using an underlying two-tier octree data structure to encode log-odds occupancy probabilities. Thanks to explicit free space encoding in a hierarchical way, the approach supports fast collision checking, crucial in robotic path planning and collision avoidance, while providing high resolution reconstructions simultaneously.

Our framework was evaluated extensively in synthetic and real-world RGB-D datasets: we showed that surface accuracy is competitive with state-of-the-art TSDF-based frameworks, while timings enable real-time or near-real-time operation even at resolutions as fine as one centimetre. We finally show how our maps can be used in real-time 3D trajectory planning for different scenarios, including large scale LIDAR ones, dramatically improving planning time without an extra step of converting the map into ESDF which many state-of-the-art approaches rely on, thus providing seamless integration between mapping and planning that is unprecedented.

REFERENCES

[1] F. Nex and F. Remondino, "UAV for 3D mapping applications: A review," *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

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

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

[4] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. 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.

[5] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots," *arXiv:1903.02144*, 2019.

[6] E. Vespa, N. Funk, P. H. J. Kelly, and S. Leutenegger, "Adaptive-Resolution Octree-Based Volumetric SLAM," in *International Conference on 3D Vision*, 2019, pp. 654–662.

[7] A. Elfes, "Occupancy grids: A probabilistic framework for robot perception and navigation," Ph.D. dissertation, Carnegie Mellon University, 1989.

[8] S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, 2003.

[9] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[10] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially Extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[11] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 215–222.

[12] 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.

[13] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2021–2028.

[14] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-Time 3D Reconstruction at Scale Using Voxel Hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.

[15] M. Klingensmith, I. Dryanovski, 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*, vol. 4, 2015, p. 1.

[16] S. Weder, J. Schönberger, M. Pollefeys, and M. R. Oswald, "Routedfusion: Learning real-time depth map fusion," *arXiv:2001.04388*, 2020.

[17] S. Stent, R. Gherardi, B. Stenger, and R. Cipolla, "Detecting change for multi-view, long-term surface inspection." in *BMVC*, 2015, pp. 127–1.

[18] D. Duberg and P. Jensfelt, "Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6411–6418, 2020.

[19] C. Loop, Q. Cai, S. Orts-Escolano, and P. A. Chou, "A closed-form bayesian fusion equation using occupancy probabilities," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 380–388.

[20] A. Knoll, I. Wald, S. Parker, and C. Hansen, "Interactive isosurface ray tracing of large octree volumes," in *2006 IEEE Symposium on Interactive Ray Tracing*. IEEE, 2006, pp. 115–124.

[21] I. Wald, "A simple, general, and gpu friendly method for computing dual mesh and iso-surfaces of adaptive mesh refinement (amr) data," *arXiv:2004.08475*, 2020.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[23] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 1524–1531.

[24] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld lidar, inertial and vision with ground truth," *arXiv:2003.05691*, 2020.

[25] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[26] J. Gammell, S. Srinivasa, and T. Barfoot, "Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic," *IEEE International Conference on Intelligent Robots and Systems*, 04 2014.