Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping

Emanuele Vespa[1], Nikolay Nikolov[1], Marius Grimm[2], Luigi Nardi[3], Paul H J Kelly[1], Stefan Leutenegger[1]

***Abstract*—We present a dense volumetric SLAM framework that uses an octree representation for efficient fusion and rendering of either a truncated signed distance field (TSDF) or occupancy map. The primary aim of this work is to use one single representation of the environment that can be used not only for robot pose tracking, and high-resolution mapping, but seamlessly for planning. We show that our highly efficient octree representation of space fits SLAM and planning purposes in a real-time control loop. In a comprehensive evaluation, we demonstrate dense SLAM accuracy and runtime performance on-par with flat hashing approaches when using TSDF-based maps, and considerable speed-ups when using occupancy mapping compared to standard occupancy maps frameworks. Our SLAM system can run at 10-40 Hz on a modern quadcore CPU, without the need for massive parallelisation on a GPU. We furthermore demonstrate a probabilistic occupancy mapping as an alternative to TSDF mapping in dense SLAM and show its direct applicability to online motion planning, using the example of Informed RRT*.**

***Index Terms*—Mapping, SLAM, Visual-Based Navigation**

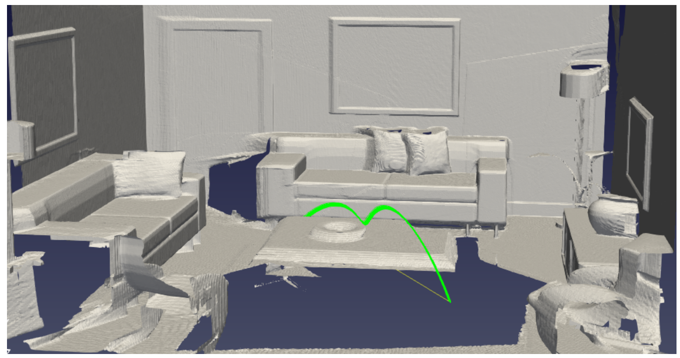

Fig. 1: Example trajectory (green) for a multicopter Micro Aerial Vehicle (MAV) computed with the Informed RRT* planning algorithm (with smoothing) depicted on a fine-grained occupancy map obtained by our SLAM system.

I. Introduction

In the past few years, SLAM research has progressed at an unprecedented speed. The widespread availability of commodity depth sensors fuelled a true paradigm shift from *sparse* systems, in which typically the maps consisted of sparse landmarks, to fully *dense* methods where essentially the full scene geometry can be reconstructed. Among the various dense SLAM systems, volumetric methods such as KinectFusion [24] rapidly became popular given the high-quality results achievable in real-time, comparable to what is attainable with more complex, inherently offline reconstruction methods such as [8][37]. At the same time, slightly different map formulations have been proposed. While occupancy mapping has been the *de-facto* standard in robotics to plan motion, recent work in high-quality dense reconstruction has recently proposed its adoption [21], reconciling the reconstruction capabilities of signed distance functions with the probabilistic rigour of occupancy grid mapping. From the point of view of efficiency, volumetric maps scale poorly as the resolution and area covered increases, implying that they could not cope with environments larger than a modest size office. A key observation is that large parts of the mapped space are actually empty, hence most voxels do not hold any significant geometric information. In this perspective, sparse representations have been introduced to considerably reduce the amount of voxels to be stored and processed. In the literature we can distinguish two predominant approaches: hierarchical data-structures, such as octrees [36] or N^3 trees [7], and flat hash-tables [26]. Although enormous speed-ups have been demonstrated with hash-tables, performance of tree based data-structures has not been very satisfactory. However, hierarchical data-structures might be desirable in different scenarios as they naturally allow for storing information at different level of details and consequently compress maps where values are constant. With this work we aim at bridging this performance gap by providing an efficient and generic fusion pipeline based on octrees. We show that the inherent overhead that the rich hierarchical structure implies is not prohibitive, while still providing a complete spatial index of the mapped scene, which is useful in many robotics applications, most prominently, planning. In summary, our contributions consist of the following elements:

1) A dense volumetric SLAM framework with Iterative-

Manuscript received: September, 10, 2017; Revised December, 5, 2017; Accepted December, 26, 2017.

This paper was recommended for publication by Editor Cyrill Stachniss upon evaluation of the Associate Editor and Reviewers' comments. This research is supported by ARM, the EPSRC grants PAMELA EP/K008730/1, Aerial ABM EP/N018494/1, and Imperial College London.

[1]Emanuele Vespa, Nikolay Nikolov, Paul H J Kelly and Stefan Leutenegger are with the Department of Computing, Imperial College London, United Kingdom. [e.vespa14, nikolay.nikolov14, p.kelly, s.leutenegger]@imperial.ac.uk

[2]Marius Grimm is with the Autonomous Systems Lab, ETH Zurich, Switzerland. mgrimm@student.ethz.ch

[3]Luigi Nardi is with the Department of Electrical Engineering – Computer Systems Laboratory, Stanford University, California, United States. lnardi@stanford.edu

Digital Object Identifier (DOI): see top of this page.

Closest-Point (ICP) tracking and fusion into an octree-based map implementation based on Morton numbers.

2) We present an alternative to traditional TSDF-based mapping that uses fully probabilistic occupancy mapping, which explicitly represents free space, for seamless integration with robotic planning.
3) We provide comprehensive evaluation on real-world and synthetic datasets. We demonstrate accuracy on-par with state-of-the-art volumetric SLAM pipelines and runtime efficiency on-par with InfiniTAM [16] when using TSDF maps, while showing substantial speed-ups compared to de-facto standard frameworks in occupancy mapping such as Octomap [15].
4) We show a prototype integration of our occupancy map with Informed RRT* path planning demonstrating the versatility of our framework in the robotic context.

The paper is organised as it follows. In Section II we overview the most relevant related work on spatial indexing for dense volumetric maps. Section III describes our optimised octree data-structure. In Section IV we describe our dense SLAM pipelines based on TSDF and occupancy maps. The evaluation in Section V contains quantitative results in terms of accuracy, timing and memory consumption, including an example application to motion planning.

II. Related work

Real-time tracking and mapping algorithms play a central role in many robotics and vision applications and hence have been subject of extensive studies over the past three decades. In this work, we focus on dense volumetric SLAM methods, as they offer superior mapping capabilities compared to sparse [18][22] or semi-dense methodologies [11]. The base of our work is the seminal KinectFusion [24] algorithm, where the mapped space is represented with a discrete *truncated signed distance field* (TSDF) [10]. Although capable of achieving impressive reconstruction results, methods based on discrete voxel grids suffer of scalability issues as the required memory and computation time grows cubically with the resolution or space covered.

To overcome such limitations different solutions have been proposed. In [34] and [29], a fixed size volume is shifted in space as the camera moves and mapped areas that fall outside the new covered area are converted to a compact mesh representation. Another line of research focused on exploiting the inherent sparsity of the reconstructed geometry. Spatial decomposition data-structures, such as octrees, N^3 trees or kd-trees have been widely investigated and exploited to accelerate computation in a variety of fields, ranging from graphics [9], physics [4] or computational science [3]. In the context of RGB-D volumetric SLAM, their first usage dates back to [36], where a GPU-based octree is used to store non-empty voxels. However, the speed-ups attained are not particularly significant compared to KinectFusion and furthermore the ray-casting strategy proposed is prone to drift. Similarly, Steinbrucker *et al.* [31] fuse each depth frame on a multi-scale octree optimised for CPU but rely on an external SLAM system for camera tracking. Chen *et al.* [7] propose a dynamic, GPU-based, N^3 tree-structure, where N is the branching factor for a node in the tree. Interestingly, each level of the tree may have a different branching factor and they empirically demonstrate that such strategy can bring better performance compared to rigid space subdivision.

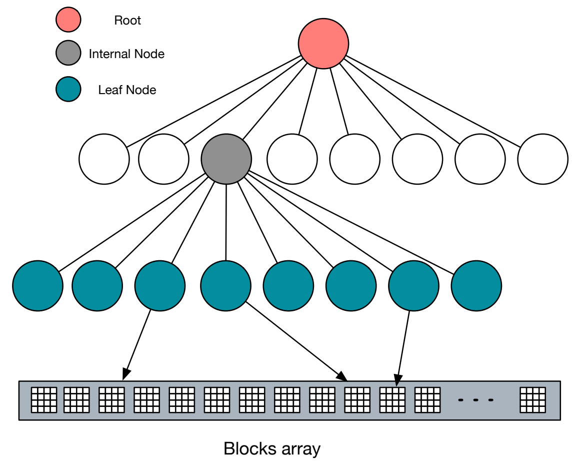

Fig. 2: Octree structure overview.

Nießner *et al.* [26] introduced a hashing-based algorithm in which non-empty voxels are organised in spatially contiguous macro-blocks and indexed via a flat hash table. This approach is taken even further with InfiniTAM [16], where speed-ups of an order of magnitude compared to previous implementations are achieved. Hashing, however, could be limiting when an explicit distinction between empty but seen space and unseen space is required. This is not a problem in case of truncated signed distance maps, as such distinction does not arise, but in case of occupancy grids, where the map is used for path planning purposes, keeping the distinction explicit is required.

Occupancy maps are commonly used in robotic contexts for path planning purposes. Frameworks as Octomap [15] use hierarchical octrees to store and update occupancy probabilities. However, as we discuss in Section IV-D, occupancy maps cannot capture the exact surface boundaries and furthermore there has not been enough focus on computational performance, making the existing solution unsuitable for real-time, incremental mapping. In a recent contribution by Oleynikova *et al.* [27], path planning is performed on an *Euclidean Signed Distance Field* (ESDF) incrementally built from a TSDF representation. This is complementary to the work presented in this paper, as we aim at providing a flexible hierarchical framework which can work with any map representations which supports the notion of empty space as required in a robotics setting, without compromising on flexibility or performance.

III. Data structure

In this section we describe the core architecture of our octree data-structure on top of which we have built the SLAM pipelines and the path-planning application described in Section IV and V-C.

A. Core components

Our framework relies on the octree hierarchy shown in Figure 2. Similarly to [26] and [16] we aggregate voxels

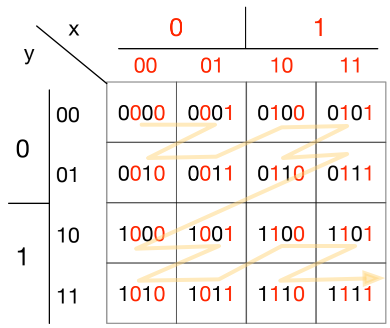

Fig. 3: Morton codes and traversal ordering for a 2D grid.

at the finest resolution into aggregated contiguous blocks of parametric size, by default 8^3 voxels. This is in contrast to previous work on octrees [36], where the deepest level stores individual voxels. In this perspective, the map simply becomes a collection of unordered sparsely allocated voxel blocks and the tree a spatial index of the scene that allows the correct piece of data to be retrieved given its integer coordinates. Optionally, internal nodes can carry data themselves, allowing for a space representation at multiple level of details. In contrast to previous works, we do not assume any particular fixed datatype, such as a signed distance function or occupancy cells. Instead we provide a flexible type traits mechanism to give full control to the application developer over the field encoded in the octree. For maximum efficiency we allocate internal nodes and aggregated voxel blocks on a memory pool which ensure thread safe lock-free batch allocations. We will discuss our allocation strategy in great details in Section III-C.

B. Information access

Efficient tree traversal is achieved via Morton coding. A Morton number can be thought of as a linear unrolling of a n-dimensional coordinate. More specifically, given a cell in a n-dimensional grid with integer coordinates $(x_1, x_2, \ldots, x_n)$ its associated Morton code is obtained interleaving the bits from each coordinate into a single number. Figure 3 shows an illustrative example of this concept on a two-dimensional four-by-four grid. As we can see, interleaved bits from the x and y coordinate form a unique code for each cell. A crucial property of these numbers is that they not only uniquely identify voxels in a regular grid, but that the higher bits recursively represent the location of parent voxels in a coarser grid, effectively specifying a full traversal of the correspondent tree and implicitly defining its structure. As an example, if we consider cell $(x, y) = (2, 1)$ with its associated code 0110, starting from the root we would first descend to the top-right sub-grid (code 01) and then select the child with code 10, i.e. our target pixel with code 0110.

C. Voxel blocks allocation

Our library targets real-time mapping applications, hence it assumes a continuously growing mapped space. This implies that the allocation of new voxel blocks in the hierarchy must be performed extremely fast and with the lowest overhead possible. Parallel tree allocation strategies have been widely explored in the computer graphics domain as hierarchical data-structures are common accelerators for ray-tracing and collision detection algorithms [20], [13]. To maximise parallelism in the tree construction, we adopt a technique based on Morton numbers inspired by [13] and [1]. We use a breadth-first top-to-bottom allocation which takes full advantage of this numbering property. First, each voxel to be allocated is associated with its Morton number and the resulting list of keys is sorted. For each level in the tree, we filter the key list by masking each code with the appropriate bit-mask for the current level. The bit-mask for a given level sets the bits corresponding to finer subgrids to 0. This procedure will generate duplicate keys which we eliminate with a compaction operation, as illustrated in Figure 4. This, together with the fact that by construction the structure to reach a given node would have been allocated at a previous step, allows us to allocate all the nodes in parallel without requiring any synchronisation between threads. This technique still requires a lock-step execution from one level to the next. However we found its performance satisfactory, since the actual bottleneck is the volumetric information update. More complex algorithms that avoid the synchronisation step are found in the literature, e.g. [17], if a faster allocation step is needed. Notice that, after a transient initial phase, the number of voxel blocks to be allocated per frame decreases considerably as new blocks will most likely be required at the frame border or in previously occluded regions.

D. Field interpolation

Iso-surface extraction algorithms, such as ray-casting and marching cubes, rely heavily on repeated field sampling, hence it is crucial to have efficient ways of querying the underlying representation. To this purpose, our framework provides optimised nearest neighbour and tri-linear interpolation functions. Tri-linear interpolations requires the eight discrete voxels surrounding the sampling points to be collected. In a sparsely allocated grid this could be expensive as several tree traversals are required to gather the desired points. We limit such performance penalty by observing that there is a finite number of access patterns which can occur. First, if the point to be interpolated falls in the middle of a voxel block, then all eight points will be local to that block and hence only one tree traversal is required. The other extreme case is when the point falls exactly on the corner of a voxel block, in which case eight tree traversals will be needed. There are six remaining configurations which correspond to the case in which the sampling point is on a voxel block edge along one or two dimension. In this case the query order is particularly important as we should insure as much locality as possible, since a bad ordering might imply more tree traversals than actually needed. In [16] this issue is addressed by caching the lastly accessed block, but this still does not help if the gathering order jumps from one block to another invalidating the cached block. Instead, we precompute statically a traversal order for all the possible configurations and at run-time we simply select the optimal for the requested sampling point position. In this way we guarantee the optimal reuse of tree

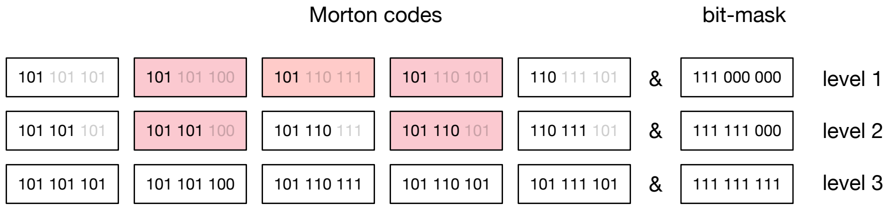

Fig. 4: Bit-masking the key-set at each allocation level, coloured boxes indicate duplicate codes.

traversals without any caching and furthermore we eliminate the unnecessary conditionals which a caching strategy implies.

IV. Dense Tracking and Mapping

We contribute with two dense volumentric SLAM pipelines implemented with our octree library presented in the previous sections. The first is the standard KinectFusion [25] pipeline, where the world map is encoded in a implicit *truncated signed distance function* (TSDF). Our second pipeline instead is based on the occupancy mapping framework introduced in [21], which we extend and refine in order to make it suitable for incremental tracking and mapping. In fact, we adopt the formulation of log-odds widely used in traditional robotic occupancy mapping. While the two mapping methods differ semantically, they both implicitly represent surfaces as zero-crossings. Therefore, the reconstruction pipelines share the same structure: i) a *tracking* stage to perform frame-to-model alignment and recover the camera pose, which includes *ray-casting* to extract a synthetic point-cloud from the model; ii) a *fusion* stage to integrate the new sensor data into the map.

A. Notation

We denote the camera pose relating the camera coordinate frame C to the World coordinate frame W as $\mathbf{T}_{WC} \in \mathbb{SE}_3$, short $\mathbf{T}$ and further indicate the respective time step k with $\mathbf{T}_k$. $\mathbf{R}_k$, $\mathbf{V}_k$ and $\mathbf{N}_k$ denote the input raw depth, vertex and normal maps, respectively. Three dimensional points $\mathbf{p} = [p_x, p_y, p_z, 1]^T$ are generally expressed in their homogeneous coordinates. To ease the notation, we introduce the function $\pi(\cdot)$ which performs dehomogenisation, perspective projection and application of the intrinsics matrix. $\pi^{-1}(\cdot)$ denotes its inverse using the depth map. Finally, we use $\mathbf{u} \in \mathbb{R}^2$ to describe pixels in image frame.

B. Surface prediction and tracking

Different approaches are possible to track the camera movement. KinectFusion adopts a variant of the well known *iterative closest point* (ICP) [2] algorithm with point-to-plane metric. Others, such as [6] and [5], have proposed to align the new camera frame directly to the TSDF map, demonstrating levels of accuracy on par with the current state-of-the-art. While such techniques could be implemented in our framework, in this work we opt for ICP alignment, as it can be easily shared between the mapping approaches discussed in Sections IV-C and IV-D. Hence, we minimise the following energy function via Gauss-Newton:

$$E_k = \sum_{\mathbf{u} \in \Omega_k} \|(\mathbf{T}_k \mathbf{V}_k(\mathbf{u}) - {}_W\hat{\mathbf{V}}_{k-1}(\hat{\mathbf{u}}))^T \, {}_W\hat{\mathbf{N}}_{k-1}(\hat{\mathbf{u}})\|_2, \quad (1)$$

where Ω_k is the set of all pixels in image frame. $\hat{\mathbf{u}}$ is the pixel in the previous image corresponding to $\mathbf{u}$ in the current image,

$$\hat{\mathbf{u}} = \pi(\mathbf{T}_{k-1}^{-1} \mathbf{T}_k \pi^{-1}(\mathbf{u})). \quad (2)$$

${}_W\hat{\mathbf{V}}_{k-1}$ and ${}_W\hat{\mathbf{N}}_{k-1}$ are the vertex and normal maps represented in the world frame rendered into the previous camera. We extract both of them directly from the volumetric representation via ray-casting. For each pixel in image frame, we cast a ray and find the closest zero-crossing along the ray. To speed-up this step, we exploit our octree data-structure to prune the ray-casting range. We use a modified version of the hierarchical algorithm described in [20] which allows us to march a ray starting from the tree root and navigating the branches till we reach the first intersected leaf node. We then continue the ray marching performing tri-linear interpolation only when field values are sufficiently close to the zero-crossing. Once an intersection has been found, we compute the surface gradient via central difference directly on the volumetric representation.

C. Signed-distance function mapping

Before proceeding with the actual map update, it must be assured that the portion of the volume affected by it is allocated. So the first step is to infer from the current depth frame which voxel blocks will be effectively updated. A key observation is that TSDF fields encode significant information only within the truncation region $\pm\mu$. In the literature, different techniques have been proposed. [36] and [7] sweep over the hierarchical grid projecting voxels from coarse to fine grain resolution, marking which voxels fall within truncation region of the current frame. A similar approach, proposed in [19], is to allocate all the blocks that fall within the camera view-frustum bounding box. However this technique significantly over-allocates and requires garbage collection to deallocate voxels that fall outside the truncation region. Instead, we choose to follow the ray-casting method proposed in [26], [7]. For each pixel in the image frame, we march a ray along the line of sight within the user specified μ bandwidth enclosing the corresponding depth measurement.

Once the new parts of the scene have been allocated, the measurement integration is done in the same fashion as in [24]. Each voxel at position $\mathbf{p}$ is projected into the current depth image $\mathbf{R}_k$ with known pose $\mathbf{T}_k$ and its TSDF value $\mathbf{F}_k(\mathbf{p})$ from the corresponding depth measurement is computed. Mathematically:

$$\begin{aligned} \eta &= \mathbf{R}_k(\pi(\mathbf{T}_k^{-1}\mathbf{p})) - p_z, \\ \mathbf{F}_k(\mathbf{p}) &= \min(1, \frac{\eta}{\mu}) \;\text{ iff }\; \eta \geq -\mu, \end{aligned} \tag{3}$$

The computed TSDF sample $\mathbf{F}_k(\mathbf{p})$ is then integrated in the global TSDF by means of block averaging.

D. Occupancy mapping

TSDF mapping works extremely well for surface reconstruction but, compared to occupancy grid mapping, it does not come with a probabilistic interpretation and it cannot properly capture information about mapped, yet empty space. On the other hand, classic occupancy grid mapping is not able to properly express the map geometry as the surface boundaries are not well defined. Loop *et al.* [21] bridge this gap by introducing a new occupancy mapping framework in which the surface geometry is well defined and hence it could be used for 3D reconstruction retaining at the same time all the semantic information of an occupancy grid. This is achieved by setting the surface boundary where the occupancy probability transitions from less than $\frac{1}{2}$ to greater than $\frac{1}{2}$. The consistency of such estimate is guaranteed by a b-spline noise model which allows to overcome both theoretical and technical difficulties arising from standard Gaussian noise. We refer to the original paper for further discussion on its theoretical aspect.

While the proposed approach works well for static scenes and a static multi-camera configuration, it is not well fit for SLAM applications. One of the reasons is that the quadratic b-spline noise model has finite support, implying that a single outlier measurement would cause erroneous holes that could never be recovered. This means that the formulation cannot be used for SLAM, where incremental fusion into a persistent map is desired and outliers will occur sooner or later. Moreover, dynamically changing environments cannot be properly handled for the same reason. Given the above discussion, we alter the formulation proposed in [21] to make it suitable for SLAM and dynamically changing environments.

We start with the model for the true depth m_r given a noisy measurement z_r adopted from [21], in the form of a quadratic b-spline, defined as:

$$p(m_r|z_r) = q(s) = \begin{cases} \frac{1}{16}(3+s)^2, & \text{if } -3 \leq s \leq -1, \\ \frac{1}{8}(3-s^2), & \text{if } -1 \leq s \leq 1, \\ \frac{1}{16}(3-s)^2, & \text{if } 1 \leq s \leq 3, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where $s := (m_r - z_r)/\sigma_r$ denotes the distance from the camera centred around the true distance and normalised with the standard deviation σ_r of the measurement. Importantly, we can now set σ_r to be proportional to r^2 corresponding to more realistic triangulation-based depth camera noise model than assuming it constant.

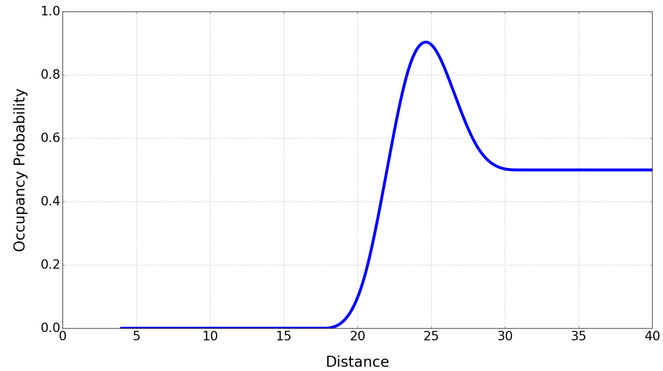

Fig. 5: Example occupancy probabilities along a ray using the analytic formulation [21] (with a slightly unrealistically large depth uncertainty for visualisation purposes).

We can proceed to derive the occupancy probabilities $P(S_r = 1|z_r)$ along the ray as

$$P\left(S_r = 1|M_r\right) = \int_{m_r=0}^{\infty} P(S_r = 1|m_r)p(m_r|z_r)dm_r. \tag{5}$$

Here, $P(S_r = 1|m_r)$ is modelled as zero in front of the true surface, 1 from the true surface to $3\sigma_r$ behind, and then followed by 0.5.

Conveniently, the above integral has an analytic solution

$$P\left(S_r = 1|M_r\right) = h(s) = q_{\text{cdf}}(s) - \frac{1}{2}q_{\text{cdf}}(s-3), \tag{6}$$

with

$$q_{\text{cdf}}(s) = \begin{cases} 0, & \text{if } s < 3, \\ \frac{1}{48}(3+s)^3, & \text{if } 3 \leq s \leq 1, \\ \frac{1}{2} + \frac{1}{24}s(3+s)(3-s), & \text{if } 1 < s < 1, \\ 1\frac{1}{48}(3s)^3, & \text{if } 1 \leq s \leq 3, \\ 1, & \text{if } 3 < s. \end{cases} \tag{7}$$

We visualise an example of this per-ray occupancy "measurement" function in Figure 5.

We can now use occupancy measurements $o_k(\mathbf{p})$ associated with the above values $h(s)$ along every ray observing depth for fusion into our octree-based map volume storing occupancy values $O(\mathbf{p})$ at each position $\mathbf{p}$. We assume a uniform prior of $O_0 = \frac{1}{2}$. But rather than the multiplicative update following direct application of Bayes' theorem (as used in [21]), we adopt the *log-odds* space, which is mathematically equivalent, i.e. using the measurement log-odds l_k to update the values L_{k-1} in the volume:

$$l_k(\mathbf{p}) = \log \frac{o_k(\mathbf{p})}{1 - o_k(\mathbf{p})}, \tag{8}$$

$$L_k(\mathbf{p}) = L_{k-1}(\mathbf{p}) + l_k(\mathbf{p}), \tag{9}$$

where $L_0 = \log \frac{0.5}{1-0.5} = 0$. A surface in log-odd space is thus defined by the zero-crossing equivalently to the TSDF formulation.

Since $h(s)$ contains values of 0, which is neither desirable, since related outliers won't be recovered, nor feasible in the log-odds formulation, we clamp h to the interval $[P_{\min}, P_{\max}]$.

In our experiments, we choose the admissible interval as $[0.03, 0.97]$.

Our second contribution to the Bayesian fusion model is a windowed update step which introduces uncertainty proportionally to the time difference between subsequent updates, in order to accommodate for otherwise unmodelled effects, most importantly dynamic scene content and uncertainty of the tracking. Specifically, we apply a moving average before each measurement is fused into the map. Thus, our final update rule is defined as:

$$L^{+}_{k-1} = L_{k-1}(\mathbf{p})\frac{1}{1+\frac{\Delta t}{\tau}}, \tag{10}$$

$$L_k(\mathbf{p}) = L^{+}_{k-1}(\mathbf{p}) + l_k(\mathbf{p}), \tag{11}$$

where Δt is the time difference since the last update for the current cell and τ is a time constant. In our experiments we chose $\tau = 5\text{sec}$. Note that this also acts as a forgetting feature: when $\Delta t \to \infty$, $O^{+}_{k-1}(\mathbf{p}) \to 0.5$. In other words, we assume that if we have not updated a specific cell for a long period of time, we don't know its occupancy state.

V. Experimental evaluation

In this section we will detail our experimental results. All our tests have been performed within the SLAMBench framework [23] on a Skylake i7-6700HQ CPU with 16GB of memory, Ubuntu 16.10 and frequency scaling disabled. All the software has been compiled with GCC 5.4.1. All the systems used in this evaluation have been configured with 1cm voxel resolution at the finest level and with depth only tracking – to ensure a fair comparison both in terms of accuracy and computational performance.

A. Tracking accuracy TUM/ICL-NUIM

We evaluate the accuracy of our pipelines across two widely used datasets, i.e. TUM RGB-D [32] and the ICL-NUIM [14]. The former provides RGB-D sequences with trajectory groundtruth, estimated via a high frequency motion capture system. Likewise, the latter provides synthetic RGB and depth data, together with groundtruth poses. The metric chosen is the root mean square error (RMSE) of the absolute trajectory error (ATE), using Euclidean distances between the groundtruth positions and the corresponding estimated positions [32]. For a fair comparison, we use the same parameters throughout.

	ATE (m)		
Dataset	TSDF	OFusion	InfiniTAM
ICL_LR_0	0.0113	0.0305	0.3052
ICL_LR_1	0.0117	0.0207	0.0214
ICL_LR_2	0.0040	0.0050	0.1725
ICL_LR_3	0.7582	0.0786	0.4858
TUM_fr1_xyz	0.0295	0.0293	0.0273
TUM_fr1_floor	×	×	×
TUM_fr1_plant	×	×	×
TUM_fr1_desk	0.1030	0.0995	0.0647
TUM_fr2_desk	0.0641	0.0902	0.0598
TUM_fr3_office	0.0686	0.0604	0.0996

TABLE I: Absolute trajectory error (ATE) comparison between our TSDF fusion, occupancy mapping and InfiniTAM. Crosses indicate tracking failure.

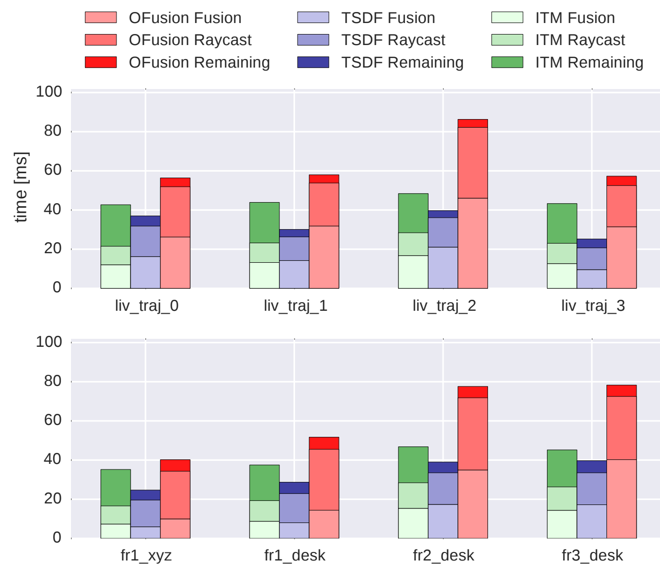

Fig. 6: Per-frame performance evaluation of InfiniTAM (ITM), octree-based TSDF fusion (TSDF) and full octree-based occupancy mapping (OFusion).

For our occupancy mapping, we use depth uncertainties proportional to the square of the distance ($\sigma_r = 4$ cm at 2 m). Table I reports our experimental results. We compare the two volumetric pipelines described in this paper, denoted as TSDF and OFusion respectively, and compare to the state-of-the-art volumetric pipeline InfiniTAM [16]. This has been tested using the default tracker with depth-only tracking to ensure a fair comparison with our solution. As we can see, our two pipelines obtain accuracy levels on par with the state-of-the-art. Interestingly, our occupancy-based fusion outperforms both TSDF and InfiniTAM in the long ICL_LR_3 and TUM_fr3_office sequences. On some sequences, all systems lose track completely, and on others, at least some systems fail partly. Note that the extension to use combined geometric and photometric tracking would be straightforward, and we consider this future work.

B. Runtime performance

Figures 6 reports the runtime performance of each pipeline benchmarked in the previous section. For each implementation, we provide timings for the depth fusion and ray-casting stage, plus an aggregated time for the rest of the pipeline which accounts for preprocessing and tracking. It is worth stressing that we are comparing fully engineered pipelines which have very different code-bases, hence part of the differences in runtime performance are attributable to

Dataset	TSDF	OFusion	Dataset	TSDF	OFusion
LR_0	7.67%	11.15%	fr1_xyz	1.95%	3.01%
LR_1	8.45%	13.68%	fr1_desk	7.70%	8.81%
LR_2	13.77%	22.52%	fr2_desk	10.15%	17.70%
LR_3	13.33%	17.68%	fr3_office	12.50%	17.95%

TABLE II: Relative memory consumption compared to a pre-allocated grid covering the same area at the same resolution.

different implementation choices.

First, we want to highlight how our octree-based TSDF mapping offers performance comparable or even superior to the state-of-the-art InfiniTAM's voxel hashing implementation. Note that apart from the voxel allocation and retrieval, the two pipelines are in fact very similar in principle. Clearly, the traversal and interpolation strategies described in Sections III-B and III-D allow us to amortise the overall cost of querying the tree. Admittedly, the occupancy grid mapping formulation is more expensive. This is inherent to the method itself as the b-spline sampling and the log-odd update is more expensive than the simple weighted average performed by the TSDF method. Furthermore, occupancy mapping has to process a larger amount of information as empty-seen space is explicitly stored and updated. Additionally, finding the zero-crossing in ray-casting is more efficient in TSDF compared to occupancy mapping, since the distance encoding allows for efficient sampling step size selection.

We also benchmarked Octomap on the test sequences used in this experiment set, configured with 5cm voxel size, but we omitted these results from Figure 6 for visualisation purposes. Mapping times per frame range between 338ms (fr1_desk) to over 1500ms (fr2_desk). The large performance gap compared to our pipeline is attributable to the slower algorithm Octomap employs, i.e. ray-cast based measurement integration, and a lack of a proper parallelisation strategy. Oleynikova *et al.* [27] propose various optimisations for ray-cast based map update and demonstrate interesting speed-ups, but still requiring at least 60ms per scan at 5cm voxel resolution. Notice that for use cases in which a coarser map is suitable, other approaches are possible. Saarinen *et al.* [30] demonstrate how using *normal distribution transform occupancy maps* (NDT-OM) they are able to achieve comparable results to Octomap while using an 8 times coarser grid, achieving 20 times faster measurement integration.

In Table II we show memory usage of our TSDF and occupancy fusion maps relative to a statically allocated grid, as used in the standard KinectFusion algorithm. As expected, the sparse data-structure allows for considerable savings in terms of memory consumption, even in case of full occupancy mapping.

C. Path planning

As a use case, we have analysed the mapping framework in a path planning application for a multicopter Micro Aerial Vehicle (MAV), with the same 1cm finest map resolution. We used THE OPEN MOTION PLANNING LIBRARY (OMPL) [33] to generate collision-free paths in our occupancy-based environment. We used Informed RRT* [12] for the straight-line segment planning. Furthermore, we compared our Octree implementation with the OCTOMAP LIBRARY [35]. The times needed to find the first feasible path for an obstructed 2.83 m start-goal distance can be seen in Table III. They were obtained on an Intel Core i7-6600U CPU at 2.60GHz, compiled on GCC version 5.4.0, averaged over 10'000 executions.

Furthermore, we calculated a smooth trajectory from the initial RRT* plans based on polynomial planning as described in [28]. We fixed the start and goal position in both mapping implementations and recorded the time needed to linearly optimize a collision free polynomial trajectory. The timings averaged over 1'000 executions are listed in Table IV. It can be seen in the recorded timings in Table III and Table IV that the straight-line planning and the linear polynomial trajectory optimization is faster with our implemented method.

	time	std dev	min	max
OFusion	12.6ms	14.6ms	4.29ms	109.6ms
Octomap	17.7ms	11.4ms	5.16ms	113.2ms

TABLE III: Timings for straight-line planning for a start-goal distance of 2.83 metres averaged over 10'000 executions.

	time	std dev	min	max
OFusion	1.57ms	0.59ms	0.43ms	3.47ms
OctoMap	2.06ms	0.78ms	0.32ms	4.17ms

TABLE IV: Timings for the linear trajectory optimization averaged over 1'000 executions.

For illustration, we have plotted an example trajectory into a map rendering in Figure 1. These results confirm that our octree-based occupancy map is at least as fast at handling spatial queries as [35], the *de-facto* standard used in research for robotic planning.

VI. CONCLUSION

In this paper we have presented an efficient octree-based dense SLAM system. Apart from supporting TSDF mapping, we contribute an extension of fully probabilistic fine-grained occupancy mapping: the occupancy map is not only used for camera tracking, but enables real-time, in-the-loop path planning on the very same representation. We experimentally evaluate our formulation in a variety of sequences, demonstrating state-of-the-art accuracy and performance results, including a comparison. We furthermore demonstrated the capabilities for planning using a probabilistic planner and trajectory smoothing. Importantly, efficient spatial queries as needed for planning are intrinsically not supported by flat hashing architectures as employed by competing SLAM systems. We thus believe this work will help to further bridge the gap between SLAM and down-stream operations and increase related efficiency by sharing a map representation of wider usefulness.

In future work, we will extend our framework in two directions. First, we will explore more aggressive optimisations to reach better run-time performance in case of occupancy mapping. Second, we will integrate our solution on drone platforms to performing fast, but safe navigation in cluttered environments.

REFERENCES

[1] J. Bédorf, E. Gaburov, and S. Portegies Zwart, "A sparse octree gravitational n-body code that runs entirely on the GPU processor," *J. Comput. Phys.*, vol. 231, no. 7, pp. 2825–2839, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2011.12.024

[2] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[3] C. Burstedde, L. C. Wilcox, and O. Ghattas, "p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[4] M. Burtscher and K. Pingali, "An efficient CUDA implementation of the tree-based barnes hut n-body algorithm," in *GPU Computing Gems Emerald Edition*. Morgan Kaufmann, 2011, pp. 75–92. [Online]. Available: http://iss.ices.utexas.edu/Publications/Papers/burtscher11.pdf

[5] 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 Conference (RSS)*, June 2013.

[6] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal, "Sdf tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 3671–3676.

[7] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 113:1–113:16, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461940

[8] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] C. Crassin, F. Neyret, M. Sainz, and E. Eisemann, "Efficient rendering of highly detailed volumetric scenes with gigavoxels," in *GPU Pro*. A K Peters, 2010, ch. X.3, pp. 643–676. [Online]. Available: http://maverick.inria.fr/Publications/2010/CNSE10

[10] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," *SIGGRAPH 96 Conference Proceedings*, pp. 303–312, 1996.

[11] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Computer Vision ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, vol. 8690, pp. 834–849. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10605-2_54

[12] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic," *CoRR*, vol. abs/1404.2334, 2014. [Online]. Available: http://arxiv.org/abs/1404.2334

[13] K. Garanzha, J. Pantaleoni, and D. McAllister, "Simpler and faster hlbvh with work queues," in *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, ser. HPG '11. New York, NY, USA: ACM, 2011, pp. 59–64.

[14] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[15] 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, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[16] O. Kahler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *Visualization and Computer Graphics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[17] T. Karras, "Maximizing parallelism in the construction of bvhs, octrees, and k-d trees," in *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, ser. EGGH-HPG'12. Eurographics Association, 2012, pp. 33–37.

[18] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[19] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3d reconstruction onboard a mobile device," in *Robotics Science and Systems 2015*, July 2015.

[20] S. Laine and T. Karras, "Efficient sparse voxel octrees," in *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '10. New York, NY, USA: ACM, 2010, pp. 55–63.

[21] 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)*, Oct 2016, pp. 380–388.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *CoRR*, vol. abs/1502.00956, 2015. [Online]. Available: http://arxiv.org/abs/1502.00956

[23] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. J. Kelly, A. J. Davison, M. Luján, M. F. P. O'Boyle, G. Riley, N. Topham, and S. Furber, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2015, arXiv:1410.2167.

[24] 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 *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, Oct 2011, pp. 127–136.

[25] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2320–2327.

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

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

[28] C. Richter, A. Bry, and N. Roy, *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. Springer International Publishing, 2016, pp. 649–666.

[29] H. Roth and M. Vona, "Moving volume kinectfusion," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 112.1–112.11.

[30] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3d normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, 2013. [Online]. Available: https://doi.org/10.1177/0278364913499415

[31] F. Steinbrucker, J. Sturm, and D. Cremers, "Volumetric 3d mapping in real-time on a cpu," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 2021–2028.

[32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[33] 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.

[34] T. Whelan, M. Kaess, and M. Fallon, "Kintinuous: Spatially extended kinectfusion," *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. [Online]. Available: http://18.7.29.232/handle/1721.1/71756

[35] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems," in *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[36] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "Octree-based fusion for realtime 3d reconstruction," *Graph. Models*, vol. 75, no. 3, pp. 126–136, May 2013. [Online]. Available: http://dx.doi.org/10.1016/j.gmod.2012.09.002

[37] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 112:1–112:8, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461919