Adaptive-resolution octree-based volumetric SLAM

Emanuele Vespa Nils Funk Paul H. J. Kelly
Stefan Leutenegger

Department of Computing
Imperial College London, London, UK
[e.vespa14, nils.funk13, p.kelly, s.leutenegger]@imperial.ac.uk

Abstract

We introduce a novel volumetric SLAM pipeline for the integration and rendering of depth images at an adaptive level of detail. Our core contribution is a fusion algorithm which dynamically selects the appropriate integration scale based on the effective sensor resolution given the distance from the observed scene, addressing aliasing issues, reconstruction quality, and efficiency simultaneously. We implement our approach using an efficient octree structure which supports multi-resolution rendering allowing for online frame-to-model alignment. Our qualitative and quantitative experiments demonstrate significantly improved reconstruction quality and up to six-fold execution time speed-ups compared to single resolution grids.

1. Introduction

Dense Simultaneous Localisation and Mapping (SLAM) systems have been at the forefront of computer vision research for the past ten years. The richness of the geometric information that this family of algorithms is able to recover in real-time is of central importance for a variety of applications, from mobile robotics to augmented and virtual reality. However, reconstructing high-fidelity models of the environment brings many challenges. First, state-of-the-art methods are notoriously expensive, both in terms of computational cost and memory footprint. Second, faithfully capturing the fine details of the scene is challenging, as maintaining a uniformly high resolution map is neither feasible, nor necessary. Any depth estimation system, whether being variable baseline multi-view stereo, stereo rigs or active depth cameras, provide information at different scales. Close-up views capture fine details that are not preserved when observed at farther distances. Hence, fusing such data at uniform scale is wrong for two main reasons: i) aliasing artefacts arising from a resolution mismatch between the map and sensor data may result in degraded map quality; ii) fine details recovered from close-up views may be lost when the scene is observed from a greater distance. While much recent research has addressed the scalability limitations of dense SLAM systems, very few have explicitly tackled these issues.

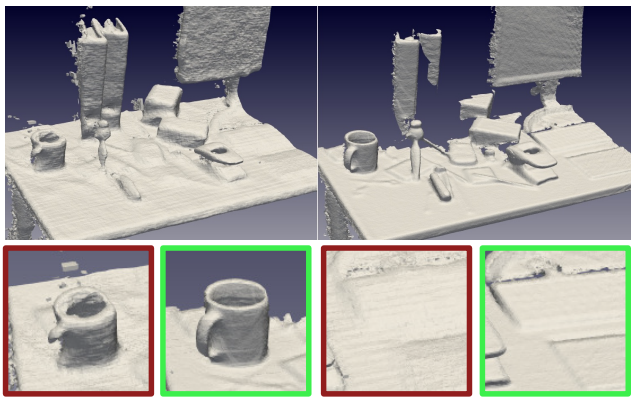


Figure 1: Comparison of final reconstruction mesh obtained with a fixed resolution grid of 2mm (on the left and in red) and our adaptive resolution system (on the right and in green). Notice how fine details like the mug handle or the shallow keyboard keys and structure are well preserved by our algorithm.

In this paper we introduce a novel multi-scale dense SLAM pipeline that supports variable resolution integration and rendering of depth data. As in many volumetric systems, our algorithm alternates between fusion of new information coming from the sensor and pose estimation against synthetic views of the scene being reconstructed. Crucially, in both integration and rendering the appropriate voxel resolution is chosen dynamically. Although we use an RGB-D camera as input, our system is sensor agnostic as long as depth is provided. As in [14], we adopt a volumetric representation based on a *truncated signed-distance function* (TSDF). To represent the volume with variable level of de-

tail (LOD), we exploit *supereight*, the octree-based framework of [21] and extend it to support dynamic resolution operations. As we discuss in the following sections, the main challenge is to keep the hierarchy consistent when fusing at variable scales. Our pipeline achieves much higher performance compared to single-resolution grids but more importantly it obtains a better reconstruction quality of cluttered scenes with thin structure.

In addition to better recovery of map details, the proposed approach substantially improves runtime thanks to more efficient map updates at larger distance. The hierarchical map nature allows for integration of new information up to the selected resolution, while only propagating updates further down when needed, effectively implementing a form of lazy information propagation up and down the octree. Summarising, our contributions are the following:

1. An extension of the hierarchical representation of [21] to support queries at any level of detail;

2. An efficient fusion algorithm that adaptively selects the appropriate level of detail and maintains the surface representation consistent across the hierarchy;

3. A qualitative and quantitative evaluation on synthetic and real datasets, which reveals that we substantially improve map accuracy, as well as runtime.

In addition, we will release our code as an open-source extension to supereight [21].

2. Related Work

Dense surface reconstruction algorithms have gained substantial attention in the scientific community in the past ten years. While a number of offline methods that achieve impressive results have been recently proposed [3, 5, 6, 20], we focus on dense reconstruction methods usable in real-time settings, such as augmented/virtual reality applications or mobile robotics. In the context of dense SLAM, after the initial KinectFusion [14] breakthrough, researchers have focused on scaling-up the size of the reconstructed environment. This has been achieved by either shifting a fixed-size fusion volume in space following camera movement [16, 22] or exploiting the sparseness of the environment using dynamically allocated data-structures such as octrees or hash-tables. Several variations of hierarchical data-structures have been proposed. Zeng *et al.* [24] introduce a GPU accelerated, octree-based volumetric representation that allow to significantly reduce the memory footprint for a given reconstruction area. However, their structure is limited to a 10-levels hierarchy, which may be too restrictive even for room-sized environment depending on the required resolution. Vespa *et al.* [21] develop an efficient octree framework which achieves real-time performance on a CPU and supports both signed-distance mapping and occupancy mapping. Chen *et al.* [2] propose a hierarchical 3D tree structure, with a parametric and possibly variable subdivision factor at each tree level. Nießner *et al.* [15] introduce an efficient reconstruction system based on flat hashing, where contiguous blocks of voxels are stored in a concurrent hash-table. Their approach has been further developed by Kahler *et al.* [10] with their InfiniTAM framework. Common to all these volumetric approaches is their reliance on single resolution grids, even if implemented with hierarchical structures.

Our work shares many aspects with the multi-scale reconstruction framework of Steinbrucker *et al.* [17], where an octree data-structure is used to store a signed-distance function representation at multiple level of detail. However, no facilities for real-time multi-scale rendering are provided and consequently it relies on an external SLAM system for pose estimation. Kahler *et al.* [11] propose a hierarchical hashing system in which the scene is represented with multiple hash-tables each with a different resolution. Interestingly, the level of refinement is chosen according to surface curvature rather than distance from the sensor. Their method is complementary to ours and we plan to unify both approaches in future work.

While till now we have focused on volumetric methods based on signed-distance fields, other representations have been proposed. Popular alternatives are explicit point-based surface representations, such as [9] and [23]. Stückler *et al.* [18] develop a complete tracking and mapping pipeline using adaptive resolution surfel maps organised in an octree. Recently, Zienkiewicz *et al.* [25] propose a reconstruction framework based on 2.5D height-maps with adaptive mesh refinement. While achieving sub-millimetre reconstructions in real-time, their method cannot handle generic 3D shapes, which is limiting it to a small number of application scenarios. However, their incremental coarse-to-fine surface refinement is certainly related to the down-propagation strategies we exploit in our work.

3. Overview

In this paper we utilise a volumetric signed-distance function (SDF) as implicit surface representation. As in the seminal work of Newcombe *et al.* [14], the world is discretised in a regular three-dimensional grid in which each voxel holds the distance from the closest surface. It is well known that volumetric methods based on pre-allocated grids are not scalable, as the memory footprint grows cubically with the resolution or area explored. To address this issue, a number of more efficient data-structures have been proposed, such as moving volumes [22], octrees [17, 21, 24], N^3 trees [2] or hash-tables [10, 15]. Since they naturally represent data at multiple scales, we choose a hierarchical

representation based on octrees. We adopt and significantly extend the framework of [21] in order to support the dynamic resolution pipeline detailed in Section 4.3.

From a high level perspective, our system follows a standard dense tracking and mapping pipeline. As in KinectFusion, the computation is structured as a closed loop divided in three main stages:

1. A *tracking* stage, where the camera egomotion is estimated by aligning the most recent depth map $\mathbf{D}_k$ coming from the sensor against a synthetic view of the model $\mathbf{D}_{k-1}$ obtained at the previous time step. This is achieved using a variant of the well-known *Iterative Closest Point* (ICP) algorithm using point-to-plane distance and projective data-association [1]. We stress that our reconstruction method is agnostic to the tracking algorithm employed and alternatives exist. A popular choice is to use a sparse SLAM system as tracking front-end while performing dense fusion, as in [17] or [25].

2. An *integration* stage, where sensor data is fused into the volume. Crucially, and in contrast to previous volumetric approaches, we do this by dynamically selecting the level of detail at which data is fused. Furthermore, we maintain the hierarchy of voxels consistent across octree levels in a subsequent step. We report our method in detail in Section 4.3.

3. A *rendering* stage, where the volume is ray-cast from the most recent camera position and surface points are extracted at the zero-crossing of the TSDF function. The produced renderings are then used for dense ICP tracking and visualisation. Also in this case, we exploit multi-resolution information by selecting the appropriate interpolation scale depending on distance from the camera. We detail our approach in Section 4.4.

3.1. Notation

We briefly describe the notation used throughout this paper. We denote n-dimensional vectors with lower-case, bold letters, e.g. $\mathbf{x} \in \mathbb{R}^n$. Superscripts in the form of $\mathbf{x}^l$ indicate the scale l at which voxel $\mathbf{x}$ is allocated on the tree. We denote the coordinate frame in which vectors are expressed with a subscript, e.g. $\mathbf{x}_c$. We will be using two different frames, the World frame $\{w\}$ and the Camera frame $\{c\}$. We also concatenate subscripts when necessary, e.g. $\mathbf{x}_{cz}$ denotes the z-component of the vector $\mathbf{x}$ expressed in camera coordinate frame. We denote matrices with upper-case bold letters, e.g. $\mathbf{D} \in \mathbb{R}^{n \times m}$. Euclidean transformations from coordinate frame $\{c\}$ to $\{w\}$ are denoted as $\mathbf{T}_{wc} \in \mathbb{SE}_3$.

4. Multi-resolution tracking and mapping

In this section, we present the algorithmic and theoretical contributions of our work. After a brief description of our underlying data-structure and extensions to [21], we detail our reconstruction pipeline.

4.1. Hierarchical representation

Fundamentally, as a means to avoid aliasing we store the TSDF values at a selectable resolution – where we have to keep the values consistent. Our data-structure of choice is an octree, as its regular subdivision exposes a simple relationship between refinement levels – octant's TSDF values can be expressed as a function (normally the mean) of their children.

There is a wide spectrum of possible octree layouts. The simplest solution is to keep the full tree structure explicit, as in [24]. While this allows fine grained control over which voxels are allocated, it is subject to significant overheads when it comes to depth integration and ray-casting [2]. To mitigate this, a common solution is to aggregate the last levels of the tree into contiguous blocks of voxels, usually of size 8^3 [2, 10, 15, 21]. However, as a consequence the multi-scale representation of the bottom levels is lost. In [17], this issue is addressed by allocating voxel blocks of the same aggregation factor at *each* level of the tree, effectively resulting in a hierarchy of stacked bricks. For efficiency reasons, we choose an intermediate approach. As in [21], we keep an 8^3 aggregation factor at the finest level of the tree, but we also store its mipmapped representation contiguously, i.e. blocks of 4^3 and 2^3 voxels. This is in fact equivalent to instantiating a full octree of three levels for each voxel block, but without the pointers' connectivity overhead and with guaranteed spatial locality. Figure 2 depicts the architecture described in the above. An unordered set of voxel blocks is stored in a contiguous, yet dynamically grown array and indexed via a pointer-based tree structure. Voxel blocks are allocated at the deepest level (leaves) of the tree, which also corresponds to the maximum resolution attainable. The magnified voxel block shows its internal structure. Logically, in this two-dimensional 4x4 grid example, a coarser grid (green dots) is overlaid on top of its underlying finer resolution grid (red dots). In practice, we store the grids contiguously one after the other. Compared to single resolution blocks, this implies only a 14% increase in memory footprint, while significantly simplifying block management operations.

4.2. Data indexing and retrieval

The layout described in the previous section enables voxels retrieval in their corresponding coarser or finer grids efficiently and in a simple manner. While we retain the indexing scheme of [21], we add the possibility to address voxels in the mip-mapped blocks. Voxels at any scale are denoted by their integer coordinates (3D indices) $\mathbf{x}_w^l$ at the *finest* resolution possible. A scale parameter determines the factor to be applied to retrieve the corresponding entry in coarser

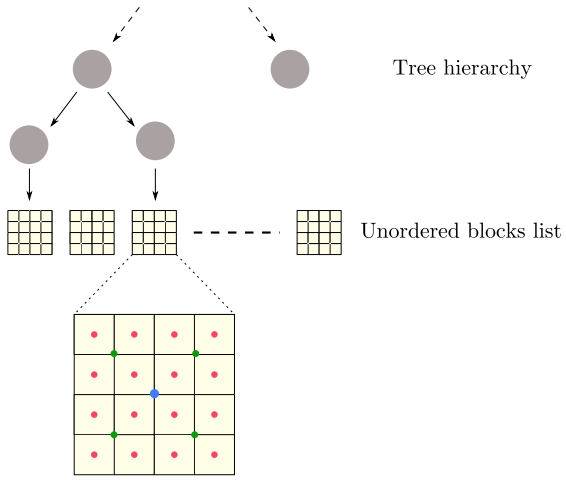


Figure 2: High-level representation of our octree structure. Coarser nodes are connected via pointers up to a maximum depth at which voxels are aggregated in continuous blocks. The focused block shows its internal structure, were coarser grids are logically overlaid on top of each other (red, green and blue dots).

grids. If d is the maximum depth of the tree and l the desired voxel depth, the scaling factor is given by $s = 2^{d-l}$, with $d - \log_2(8) \leq l \leq d$. Hence, index coordinates to access the multi-resolution grid at any scale l can be obtained by simple arithmetic as: $\mathbf{x}_w^l = s \left\lfloor \mathbf{x}_w^d / s \right\rfloor$. Also, we distinguish the world frame index coordinate representation $\mathbf{x}_w$ from block index coordinates $\mathbf{x}_b$, where we store the block offset $\mathbf{b}_w$. Finally, to access a value from memory relative to the dynamically allocated block, we linearise the block coordinate index as

$$i = \text{offset}(s) + \frac{\mathbf{x}_{bx}^l}{s} + \frac{\mathbf{x}_{by}^l}{s} v^l + \frac{\mathbf{x}_{bz}^l}{s} {v^l}^2, \tag{1}$$

where offset$(\cdot)$ is a precomputed function that returns the starting position of the local sub-grid and v^l is the number of voxels per side at scale l.

Another significant difference compared to [21] is our adoption of cell-centred voxels: we place the sampling location at the centre of its enclosing octant. While this can be trivially achieved by linear shifting, it has important consequences in terms of local consistency. As shown in Figure 2, in a cell-centred grid parent voxels are exactly the linear combination of its children. This is crucial in that it allows node-local exchange of data between tree levels. We exploit this property extensively in the algorithms described in Section 4.3.

4.3. Data integration

We structure the field update into two distinct phases. First, the last acquired depth map is fused into the map. In contrast to previous volumetric approaches, this is done by selecting the appropriate scale proportional to the distance from the camera. If the currently selected scale is finer than the scale at which integration was last performed, we *first* down-propagate the available information from parent octants (Section 4.3.3). In a second step, we up-propagate the newly fused information to the coarser nodes (Section 4.3.2). This is done in a *lazy* way: we perform upward and downward propagation only up to the tree levels which are needed for the fusion and rendering operations at the current time step. In the following, we report our method in detail.

4.3.1 Single layer update

Each new depth measurement will observe regions of space that possibly have not yet been allocated on the tree hierarchy. Hence, as in [21], we allocate the scene via ray-casting and collecting all the intersected voxels around the truncation region $\pm\mu$. However, we select the allocation scale dynamically, such that voxels far away from the camera that do not reach the aggregation layer of the tree will not be allocated as contiguous blocks. Processing the k^{th} depth frame, the TSDF sample f_k for each allocated voxel at scale l, at position $\mathbf{p}_w^l$, is computed by projecting it into camera frame and taking the signed distance to the corresponding depth measurement, as shown in Equation (2), where λ is a factor that transforms a distance along the z-axis to a range distance.

$$\begin{aligned} d &= \lambda(\mathbf{D}_k[\pi(\mathbf{T}_{cw}\mathbf{p}_w^l)] - \mathbf{p}_{cz}^l), \\ f_k(\mathbf{p}_c^l) &= \min\left(1, \frac{d}{\mu}\right), \end{aligned} \tag{2}$$

where $\pi(\cdot)$ denotes the projection from 3D coordinates in camera frame to pixels, and where $\mathbf{D}_k$ denotes the depth map that can be indexed via the operator $[\cdot]$. The current sample is then integrated in the global map F_k^l in an incremental fashion:

$$\begin{aligned} F_k^l &= \max\left(\min\left(\frac{y_{k-1}^l F_{k-1}^l + w f_k^l}{y_{k-1}^l + 1}, 1\right), -1\right), \\ y_k^l &= \min(y_{\max}, y_{k-1}^l + 1), \\ \Delta y_k^l &= \Delta y_{k-1}^l + 1, \end{aligned} \tag{3}$$

where y^l denotes the weighting and is clamped to a maximum weight of $y_{\max}$ (we use $y_{\max} = 100$) and F_{k-1}^l is either the function value at the previous time step or the down-propagated value in case of integration scale change. Note that we also keep track of the increment on y^l in the form of Δy_k^l, which will be needed later for down propagation. Notice that for compactness we omit the voxel parameter and denote $f_k(p_c^l) \equiv f_k^l$ when clear from the context.

(a) scale = 0 (b) scale = 1 (c) scale = 2

Figure 3: Synthetic rendering of a scene at progressively coarser voxel resolutions, ordered left-to-right starting from the finest resolution.

We are left with the issue of selecting the fusion scale l. We aim to keep the ratio between the back-projected pixel size and the corresponding voxel as close as possible to one. Hence, we select the appropriate resolution proportional to the distance from the camera. In principle, each voxel can be updated at a different scale as long as it is consistent among its siblings. However, this would imply unnecessary overhead when updating the densely allocated voxel blocks and complicate the subsequent up-propagation operations. Instead, we simplify the problem by assigning each *block* a uniform scale. This works very well in practice and it has the main advantage of enabling uniform iteration over the appropriate mipmapped grid (see Section 4.3.2).

4.3.2 Upward propagation

Once a depth map has been integrated, we propagate the updated information along the hierarchy in the upward direction. As we report in Section 4.4, this is required to guarantee scale consistency amongst neighbouring blocks and octants. Furthermore, we stop the up-propagation at the coarsest scale observed during the current frame, bounding the number of octants to be updated. The cell-centred data-model ensures that each octant is fully described by its children by means of simple linear interpolation:

$$\begin{aligned} F^l &= \frac{\sum_{i=1}^{8} F_i^{l+1}}{8}, \\ y^l &= \frac{\sum_{i=1}^{8} y_i^{l+1}}{8}, \end{aligned} \tag{4}$$

where the subscript i denotes the i-th child at scale $l+1$. In the rare event that not all children are initialised, we compute the mean of the initialised ones. Figure 3 demonstrates the soundness of our approach. In an artificial experiment, we fuse information at the finest resolution possible and up-propagate. We then render the same frame at progressively coarser voxel resolution via ray-casting, obtaining consistent results. Notice how the reconstruction appears smoother as the voxel size used for interpolation and gradients grows.

4.3.3 Downward propagation

We perform downward propagation of information when the camera moves closer to the surface and requires a finer resolution for depth integration. However, coarse-to-fine propagation is significantly more challenging compared to upward propagation. One possible approach is to trilinearly interpolate field values from the coarser grid, in a similar fashion to multi-grid methods [7], but this would smooth already reconstructed details which are then revisited. Instead, we take an approach similar to that of Zienkiewicz *et al.* [25] in the context of height map reconstruction. In order to preserve details, we propagate to the children octant a fixed delta which represents the difference between the last updated values at the parent scale. In order to enable lazy propagation we need to keep track of all the updates performed in the time interval $[k, \ldots, k+n]$. A key observation is that this would be equal to the difference between the children's means $\bar{F}_k^l$ at time k and the last parent value F_{k+n}^l at time $k+n$. Since $\bar{F}_k^l$ is unchanged between consecutive frames in which children's values have not been up-propagated, it holds that $\bar{F}_{k+n}^l = \bar{F}_k^l$. Using similar reasoning for the weight values, we can formulate down-propagation in compact form as:

$$\begin{aligned} \Delta F_k^l &= F_{k-1}^l - \bar{F}_{k-1}^l, \\ F_{k,i}^{l+1} &= F_{k-1,i}^{l+1} + \Delta F_k^l, \\ y_{k,i}^{l+1} &= \min(y_{\max}, y_{k-1,i}^{l+1} + \Delta y_k^l), \\ \Delta y_k^{l+1} &= \Delta y_{k-1}^l + \Delta y_{k-1}^{l+1}, \\ \Delta y_{k+1}^l &= 0. \end{aligned} \tag{5}$$

Finally, to achieve smoother propagation, we enforce scale changes in unitary steps, i.e. the current integration scale can be at most double or half the resolution of the previous used.

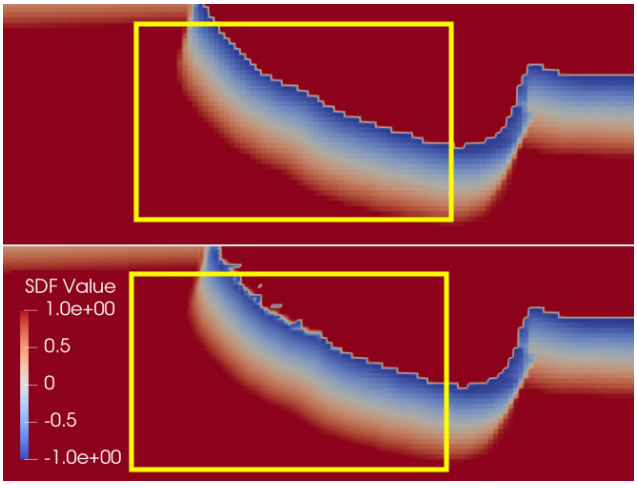


Figure 4: Progressive initialisation of fine resolution voxels from parent octants. Images display a horizontal slice of the TSDF volume ordered top-to-bottom from the older to the newer. Red indicates empty and unseen space, salmon-to-white shades denote space in front of the surface and blue corresponds to back surfaces. Regions of space that appear blocky have not yet been refined at the rendering resolution. Notice how the field is progressively smoothed in proportion to the scale refinement caused by the camera moving closer towards to the surface (highlighted by the yellow box).

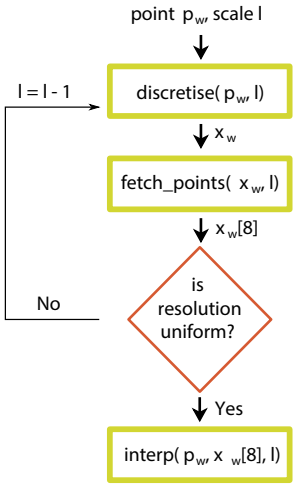
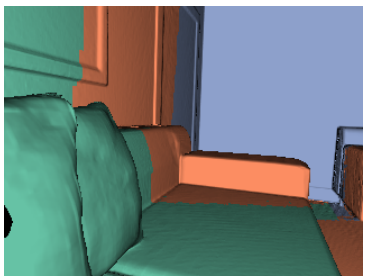


Figure 5: Schematic representation of the interpolation point-search algorithm. Given a point $\mathbf{p}_w$ and scale l, the point is converted to voxel coordinates $\mathbf{x}_w^l$ and the corresponding 8-neighbour is fetched (*fetch* function). If the neighbours' resolution is uniform, interpolation is performed at scale l (*interp* function), otherwise the search is repeated at a coarser scale.

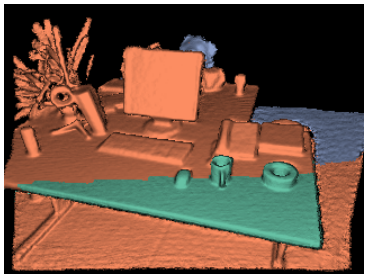
One subtle issue that remains is how to deal with voxels at finer scales that have yet to be initialised, since the delta propagation described in this section clearly would not work. Instead, on initialisation we interpolate the values from the parent grids. To keep computation as local as possible, we resort to extrapolation in order to compute values for voxels at the boundaries of contiguous blocks. This avoids complex synchronisation issues to guarantee that that the neighbouring blocks are up-to-date and it ensure a smooth initialisation, as we show in Figure 4.

4.4. Multi-resolution rendering

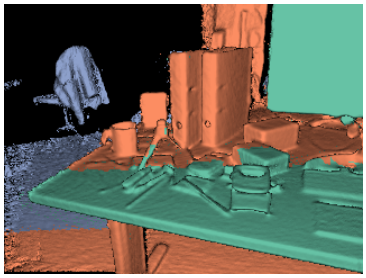
Volume rendering is implemented via ray-casting. As in [21], the hierarchy is traversed to skip empty space and advance the ray as close as possible to the surface. In the proximity of the surface the ray is marched in steps proportional to the distance from the zero-crossing. At each step, the SDF field is sampled via tri-linear interpolation. Once the ray transitions from positive to negative space the accurate 3D position of the surface is computed as in [14]. Our fusion method and delayed propagation impacts the way that interpolation and gradient calculations have to be performed. The key issue is that interpolating points at the boundaries of voxel blocks requires access to neighbouring blocks. However, there is no guarantee that the lastly integrated data has the same resolution. This would mean interpolating between points with variable spacing (i.e. akin to an unstructured grid) and consequently simple tri-linear interpolation rules would not work. There are several ways to solve this problem. In [11], a linear system is built and solved to derive the interpolation coefficients. However, this is computationally expensive. While the occurrence of interpolation across blocks of different resolution should be relatively rare and restricted to the depth regions at which the switch occurs, we opted for a simpler strategy, schematically shown in Figure 5. Given a 3D sampling point, we look for its corresponding voxel in the hierarchy at the finest, *most recently* updated resolution. Once we have found the base point in voxel space, we search for the neighbouring points at the *same* resolution. If any of the required points violates the resolution constraint, i.e. its last updated scale is greater than the current one, we repeat the search process of fetching points at the coarser scale. In other words, we perform the interpolation on the common finest grid between neighbouring voxel blocks. Figure 6 shows an example of renderings obtained with the ray-casting strategy detailed in the above. The colour-coding denotes progressively coarser interpolation scales proportionally to the surface distance. Notice how surfaces at the interface of scale changes are smoothly rendered.



(a) ICL-NUIM liv_traj_2



(b) TUM fr2_desk



(c) Desk sequence

Figure 6: Example of multi-resolution ray-cast rendering on different scenes. The colour coding indicates the variable resolution used (green finest, purple coarsest).

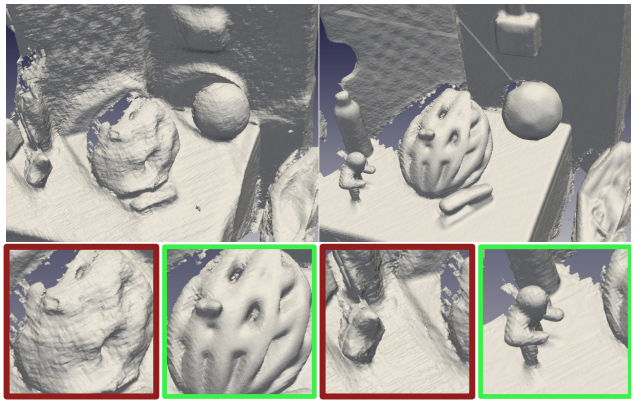


Figure 7: Reconstruction comparison between a single resolution pipeline (left and in red) and our multi-resolution approach (on the right and in green) on the helmet data-set.

5. Experimental evaluation

In this section, we report our experimental results. All our tests have been performed on a Intel Haswell i7-4770 CPU with 16GB of memory, Ubuntu 16.04 and frequency scaling disabled. The software has been compiled with GCC 8.0 with OpenMP acceleration. We evaluate our dense SLAM system, which we refer to as *multires*, against the single resolution pipeline of [21], denoted as *baseline* and InfiniTAM's voxel hashing implementation [10]. We have configured all systems with 2mm maximum voxel resolution and 5cm truncation bandwidth. ICP tracking uses a three level image pyramid with 10^{-5} convergence threshold. As the finest sensor resolution, we use QVGA, 320x240 pixels (i.e. downsampling the VGA resolution by a factor of two). We stress that the same set of parameters has been used for *all* the datasets.

5.1. Qualitative evaluation

To demonstrate the effectiveness of our multi-scale fusion approach we have recorded a series of sequences with fine structure components, such as scissors, cables, handles and objects with more complex geometry such as a helmet or a drone. In all the recorded sequences we have simulated a realistic scanning scenario, where first the scene is observed closely and then the camera slowly moves away to scan other parts of the environment. We believe this kind of scenario is particularly relevant for instance in case of augmented reality (AR) applications. It is common that in a bootstrap phase, the user is required to scan the surrounding environment. Then virtual characters and objects may be placed on desktops or cluttered scenes and have to navigate and interact with the map in an accurate way. Similarly, the scenario is very relevant in robotic exploration, e.g. using a drone, with the aim of accurately reconstructing an indoor space. Figures 1 and 7 compare the output of our novel reconstruction pipeline against a traditional single-resolution system. Both meshes are extracted via marching cubes [4] at the maximum available resolution. Our method is able to preserve the fine structure details and furthermore provides a smoother estimate of planar surfaces. We attribute this to the reduction of aliasing and better smoothing of sensor noise at coarser resolutions.

5.2. Quantitative evaluation

We evaluated the run-time performance, tracking and reconstruction accuracy of our pipeline on standard synthetic (ICL-NUIM [8]) and real (TUM RGB-D [19]) data-sets using the SLAMBench framework [13]. Table 1 shows the absolute trajectory error (ATE) across different sequences from each data-set. With regard to tracking accuracy, our multi-resolution system achieves same or better results than the single resolution methods. Note that InfiniTAM results reported in [10] are slightly superior to what we report, but were obtained at VGA resolution and at coarser voxel resolution, while for fairness we used the same voxel and in-

Figure 8: Mean execution time per frame, in milliseconds. The top and bottom respectively rows indicate timings for the ICL-NUIM and TUM RGB-D datasets.

	ATE (m)		
Dataset	Baseline	Multires	InfiniTAM
ICL_LR_0	0.1315	0.0031	0.1384
ICL_LR_1	0.0024	0.0061	0.0044
ICL_LR_2	0.0083	0.0052	0.0065
ICL_LR_3	0.1213	0.0539	0.1071
TUM_fr1_xyz	0.0137	0.0140	0.0273
TUM_fr1_desk	0.0633	0.0622	0.0492
TUM_fr2_desk	0.0838	0.0853	0.0887
TUM_fr3_office	0.0226	0.0227	0.1022

Table 1: Absolute trajectory error (ATE) comparison on sequences from the ICL-NUIM and TUM datasets.

put resolutions (2mm, QVGA) across all experiments. Also note that we manually tuned various InfiniTAM parameters such as the truncation distance and found that best results were obtained when using the same parameters as our approach. Table 2 reports the root mean squared error (RMSE) of the reconstruction error on the ICL-NUIM sequences, where the results are consistent with the tracking accuracy. Notice that this synthetic dataset lacks fine structure elements and consequently, given the same tracking accuracy, it is expected to have a reconstruction precision very close to single resolution grids.

As shown in Figure 8, multi-resolution fusion brings reductions in computational cost, achieving up to 6x higher frame rate compared to the baseline and always performing better than voxel hashing. This is due to the combination of our dynamic resolution fusion, as it reduces the number of voxels updated per frame according to the distance from the camera, and the delayed propagation of newly fused

	RMSE (m)		
Dataset	Baseline	Multires	InfiniTAM
ICL_LR_0	0.0532	0.0054	0.0541
ICL_LR_1	0.0048	0.0080	0.0057
ICL_LR_2	0.0051	0.0048	0.0049
ICL_LR_3	0.0568	0.0110	0.0547

Table 2: Reconstruction precision evaluation in terms of the Root Mean Square Error (RMSE) distance on the ICL-NUIM dataset.

information. Notice that rendering is faster when using a single resolution grid. In our current implementation, the extra control logic required by the interpolation scheme described in Section 4.4 induces a non-negligible execution time penalty compared to the simpler interpolation algorithm usable in case of uniform scale. Overall, the system presented in this paper achieves frame rates between 5hz and 10hz on a commodity CPU, while providing very fine scale reconstructions. This is a strong indication that the method is scalable and if ported to GPU accelerators could reach levels of performance far exceeding the camera's frame rate of 30fps even at the full VGA resolution.

6. Conclusions

We have presented a method for the online volumetric fusion of depth images at adaptive levels of detail. Our system dynamically select the best integration scale to match the sensor resolution and propagate up and down the octree hierarchy as required in a lazy fashion, guaranteeing reconstruction consistency and significantly improved runtime performance compared to equivalent single-resolution grids.

There are several directions in which we plan to extend our work. The surface refinement criteria based on local curvature proposed by Kahler *et al.* [11] allows for significant reduction of memory usage in correspondence of flat surfaces. We believe that this can be combined with our distance-based criteria and yield significantly better performance while preserving reconstructed details. We also plan to port our approach to occupancy mapping, specifically extending the formulations introduced in [12, 21]. This would allow us to better take into account sensor noise in a probabilistically sound framework.

7. Acknowledgements

We would like to thank ARM, SLAMCore Ltd., the EPSRC grants PAMELA EP/K008730/1, Aerial ABM EP/N018494/1, and Imperial College London for generously funding this research. We also would like to thank Pablo F. Alcantarilla for the numerous fruitful discussions.

References

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

[2] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. Graph.*, 32(4):113:1–113:16, July 2013.

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

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

[5] S. Fuhrmann and M. Goesele. Fusion of depth maps with multiple scales. *ACM Trans. Graph.*, 30(6):148:1–148:8, Dec. 2011.

[6] S. Fuhrmann and M. Goesele. Floating scale surface reconstruction. *ACM Trans. Graph.*, 33(4):46:1–46:11, July 2014.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[8] 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.

[9] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. *3Dv*, pages 1–8, 2013.

[10] O. Kähler, 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*, PP(99):1–1, 2015.

[11] O. Kähler, V. Prisacariu, J. Valentin, and D. Murray. Hierarchical voxel block hashing for efficient integration of depth images. *IEEE Robotics and Automation Letters*, 1(1):192–197, Jan 2016.

[12] 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)*, pages 380–388, Oct 2016.

[13] 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.

[14] 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*, pages 127–136, Oct 2011.

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

[16] H. Roth and M. Vona. Moving volume KinectFusion. In *Proceedings of the British Machine Vision Conference*, pages 112.1–112.11. BMVA Press, 2012.

[17] 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*, pages 2021–2028, May 2014.

[18] J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *J. Vis. Comun. Image Represent.*, 25(1):137–147, Jan. 2014.

[19] 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.

[20] B. Ummenhofer and T. Brox. Global, dense multiscale reconstruction for a billion points. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[21] 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*, 3(2):1144–1151, April 2018.

[22] T. Whelan, M. Kaess, and M. Fallon. Kintinuous: Spatially extended KinectFusion. *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[23] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, Rome, Italy, July 2015.

[24] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based fusion for realtime 3D reconstruction. *Graph. Models*, 75(3):126–136, May 2013.

[25] J. Zienkiewicz, A. Tsiotsios, A. Davison, and S. Leutenegger. Monocular, real-time surface reconstruction using dynamic level of detail. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 37–46, Oct 2016.