2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

C-blox: A Scalable and Consistent TSDF-based Dense Mapping Approach

Alexander Millane, Zachary Taylor, Helen Oleynikova, Juan Nieto, Roland Siegwart, César Cadena
Autonomous Systems Lab, ETH Zürich

Abstract— **In many applications, maintaining a consistent dense map of the environment is key to enabling robotic platforms to perform higher level decision making. Several works have addressed the challenge of creating precise dense 3D maps from visual sensors providing depth information. However, during operation over longer missions, reconstructions can easily become inconsistent due to accumulated camera tracking error and delayed loop closure. Without explicitly addressing the problem of map consistency, recovery from such distortions tends to be difficult. We present a novel system for dense 3D mapping which addresses the challenge of building consistent maps while dealing with scalability. Central to our approach is the representation of the environment as a collection of overlapping Truncated Signed Distance Field (TSDF) subvolumes. These subvolumes are localized through feature-based camera tracking and bundle adjustment. Our main contribution is a pipeline for identifying stable regions in the map, and to fuse the contributing subvolumes. This approach allows us to reduce map growth while still maintaining consistency. We demonstrate the proposed system on a publicly available dataset and simulation engine, and demonstrate the efficacy of the proposed approach for building consistent and scalable maps. Finally we demonstrate our approach running in real-time onboard a lightweight Micro Aerial Vehicle (MAV).**

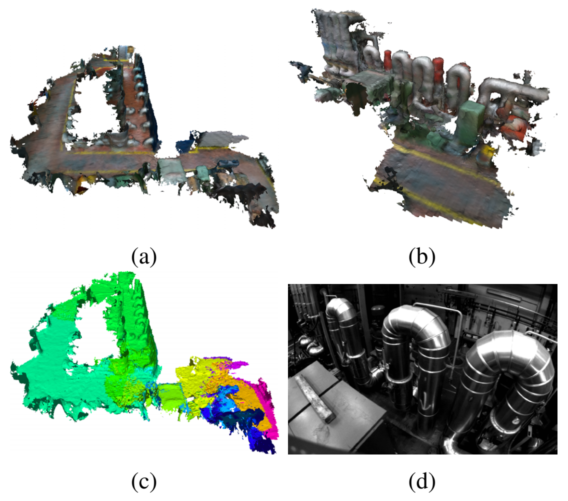


Fig. 1: Reconstructions resulting from 2 MAV flights in an indoor industrial area. The agent trajectories include several wide-baseline loop closures and frequent re-observation of similar parts of the scene. Figures (a) and (b) show the final reconstruction from the two flights, (c) shows the breakdown of the flight 1 map into its composite submaps, and (d) shows a view from the onboard camera onto the pipe-like structures visible in the reconstruction (b).

I. Introduction

Vision-based perception systems are increasingly being deployed for use on robotic platforms that operate in unstructured environments or without access to reliable GPS coverage [1]. In addition to offering a sensing solution that does not depend on any external infrastructure, the benefits of such systems include their low weight, low cost and the richness of the data they provide.

Key competencies towards achieving high level tasks for robotic systems utilizing vision, are building an internal representation of the environment and localizing within it, known as Simultaneous Localization And Mapping (SLAM). The SLAM problem, has been a focus of robotics research for the last three decades. Most successful SLAM systems utilizing visual data simplify the problem by converting incoming images to a set of visual features, before estimating the camera motion and the map as a function of only these feature observations [2]. A summary of past and present SLAM systems can be found in [3].

While feature-based systems have proven themselves effective for localization, a map comprised of a sparse collection of 3D points is insufficient for tasks such as planning or environmental interaction. Recently, the commodification of depth cameras has seen impressive results produced in the field of dense 3D reconstruction from visual sensors [4], [5]. The techniques used by these works are now making their way into robotics applications [6], [7]. However, the shortcomings of SLAM systems mean that they suffer from imperfect odometry, resulting in accumulating pose drift, and delayed loop closures. Using a SLAM system for camera tracking in an online dense reconstruction pipeline can therefore produce inconsistent reconstructions if these effects are not handled explicitly.

This paper introduces *c-blox*[1], a novel dense mapping system which addresses the challenge of maintaining map consistency and scalability. Central to our system is the representation of the observed scene as a collection of overlapping TSDF subvolumes[2]. In our approach we create new subvolumes early and often to limit the amount of intra-volume distortion. However, the approach of continuously creating subvolumes has the effect of building a map that grows over time and without bound. We therefore limit the resulting map growth by performing subvolume fusion, where doing so has a high probability of producing consistent results. We first follow a landmark covisibility [10] based approach to

This research was funded by the National Center of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation.

[1]`https://github.com/ethz-asl/c-blox`
[2]We use the term subvolume to refer to 3D dense submaps, also termed *patches* in [8] and [9].

identify subvolumes containing potentially redundant views of the environment. In a second stage, we determine if these candidate subvolumes are well localized with respect to one another by extracting a measure of the relative localization certainty of the candidates. This certainty measure is extracted from the co-constructed sparse map to which the subvolumes are attached. In summary the contributions of our system are,

- a systematic approach for maintaining consistency in a dense TSDF-based map while limiting the growth of the map, based on probabilistic measures;
- completely CPU-based implementation to allow for use on lightweight robotic platforms lacking GPU hardware;
- the addition of threading infrastructure and an approximate fast integrator to the open-source TSDF library Voxblox.

II. Related Work

There has been extensive work on SLAM over the last three decades, of which visual SLAM forms a significant part [3]. Dense 3D mapping from image data has also received considerable research focus, and in recent years there has been a surge in the number of works in this field. A plethora of systems have been developed, for which we give a brief review of the most relevant.

Dense mapping systems have employed a number of different representations of the environment, a choice which determines many properties of the resulting system. Engel et. al. [11] represent the world as a collection of well localized depth frames produced by multi-view stereo. Whelan et. al. [12] represent the world as a collection of surfels in 3D space. Another approach is occupancy grids, which represent the world as a collection of occupancy probabilities stored over a voxel grid [13], and have been applied successfully to robotic systems in the past [1]. In this work we represent the observed world by maintaining a TSDF over a discrete grid, an approach which has shown compelling results in recent years [4]. Such representations offer a systematic method for incremental fusion of noisy depth frames, provide high fidelity reconstructions, and make no assumptions about the structure of the environment. Recent works on sparse representations of TSDFs have shown that these techniques can also be employed at larger scales [14]. Furthermore, while keyframe depth-maps and surfel-based representations produce very accurate reconstructions, these representations are challenging to use directly for other robotic tasks such as motion planning. The TSDF-based volumetric representation is more conducive for such tasks, where free space and surface connectivity information are important [7], [6].

Most existing online reconstruction systems use frame-to-frame or frame-to-model alignment for tracking the sensor pose at each frame, before integrating depth data into the map [4], [14]. These kind of incremental systems inherently suffer from camera tracking drift, which can lead to dramatic distortions in the reconstructed environment [14]. There have been several works which aim to address the challenge of producing consistent dense maps, of which we review the most relevant.

Whelan et. al. [5] introduce a system which restricts the volume of active TSDF reconstruction, converting data leaving this volume to a triangular mesh. To ensure consistency, the global mesh is deformed to reflect loop closures following a rigid-as-possible approach. ElasticFusion [12] took a similar approach to maintaining consistency, achieved by bending a surfel-based map, which allowed for place revisiting. BundleFusion [15] enforced consistency by storing the history of integrated frames. Camera poses are globally optimized with each arriving frame and the global reconstruction updated, producing compelling globally consistent reconstructions. This method however faces scalability issues during on-line use.

Another approach to maintaining a consistent map is to represent the global map as a combination of submaps. The global map can then be computed as a function of these submaps only. This approach is not new, going back at least to the *Atlas* framework [16] and DenseSLAM [17] where submapping was used to extend graph-based SLAM to large scale environments. Several works have shown the efficacy of TSDF-subvolume approaches for maintaining map consistency [9], [18], [19], [7], [20]. These works can be broadly separated into two categories; those that attempt to *partition* space to minimize subvolume overlap, and those that do not partition space.

Partitioning the workspace means that the constructed map size grows linearly with the size of the observed environment, rather than time or trajectory length. Patch Volumes [9] divides the observed scene into "map patches", which are aligned with planar surfaces in the environment. In a similar approach Kähler et al. [18] attempt to minimize subvolume overlap. While space efficient, there are two disadvantages of pursuing an approach of subdivision. Firstly, when revisiting an existing subvolume, one must ensure the camera pose is consistent with that volume in order to integrate the information, which in practice is difficult. Secondly, both [9] and [18] require ray casting into several subvolumes simultaneously to perform camera tracking, significantly increasing the computational requirements of the system.

By contrast, several systems [19], [7], [20] make no attempt to partition space or to control subvolume overlap. The authors in [19] build reconstructions as a composition of mesh fragments and produce compelling results. Both systems however, are aimed at producing high quality reconstructions in post processing, leading to high computation times, making these approaches inappropriate for real-time robotics applications. The authors in [7] and [20] represent the observed world as a number of potentially overlapping TSDF volumes and show this to be effective for maintaining map consistency. We extend this approach. The main drawback of the existing works in this direction is that the maps they produce grow linearly with the trajectory length [7], or with time [20]. Our proposed approach differs from these systems in a number of technical aspects (see section Section IV), however the main scientific novelty of our system is to address the scalability problem by introducing a systematic approach to limiting map growth.

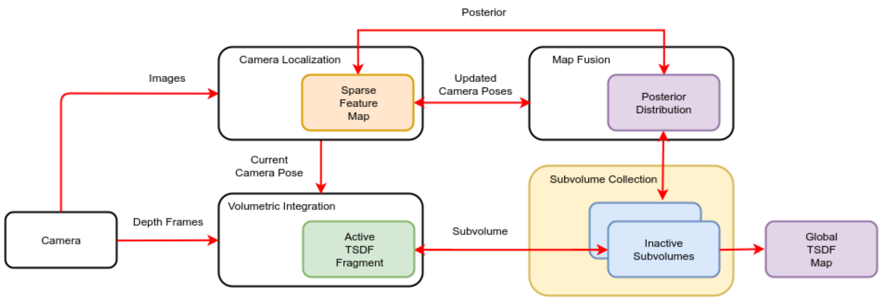


Fig. 2: An overview of the proposed mapping system.

To summarize, the existing approaches which address the challenge of maintaining consistency in dense maps either: a) maintain a map in a form which requires further processing, for example point/surfel clouds or keyframe depth maps; b) make fusing new information into previously mapped locations difficult for example mesh-based representations; c) use offline or computationally expensive techniques for global optimization of the reconstruction; d) build maps which grow in size without bound, even in a bounded size environment. In constrast to existing work, the proposed system maintains the map in a form suitable for use in robotics, allows seemless place revisiting, and limits the growth of maps online.

III. Problem Statement

Given a sequence of images $\{I^i\}_{i=1}^M$ and associated depth maps $\{D^i\}_{i=1}^M$, we aim to build a dense 3D map of an observed scene. The camera coordinate frames $\{C^i\}_{i=1}^M$ are parameterized as rigid transformations with respect to a global reference frame G as $\mathbf{T}_{GC^i} \in \text{SE}(3)$. Central to our approach is the use of a TSDF for implicit surface representation, which we denote as the function $F : \mathbb{R}^3 \rightarrow (d, w, \mathbf{c})$ mapping 3D points to a tuple, consisting of d the distance to the nearest observed surface, w a weighting/confidence measure, and $\mathbf{c}$ the observed color [4]. As is common in recent reconstruction systems [14], [21], we store this function as a collection of sparse samples over a discrete uniformly-spaced voxel grid.

If the set of poses $\{\mathbf{T}_{GC^i}\}_{i=1}^M$ is well determined at the time of image capture, existing techniques [4], [14], [21] can be used to construct a global TSDF map. We however, focus on the case where camera frames are not well *globally* localized at the time of capture, for example due to accumulated drift or delayed loop closure.

IV. System Description

The proposed system builds on the TSDF mapping techniques first presented in [4] and later expanded by a number of works [21], [5], [14]. An overview of our system is shown in Fig. 2 and consists of modules for camera tracking/localization (Section IV-B), volumetric integration (Section IV-C), map fusion (Section IV-D), and subvolume fusion (Section IV-E). We start with a description of our map respresentation as a subvolume collection.

A. TSDF Sub-volumes

When fusing observation data into a single map, inconsistencies can be caused by integrating data from a poorly localized sensor. Particularly in the case of dense maps, such errors tend to be difficult to recover from because correlations between observation and localization information are usually lost in order to ensure mapping remains efficient. We pursue a map structure which remains consistent *by construction*. We represent the scene as a collection of locally consistent sub-volumes. However, we make no attempt to *partition* the global space, allowing for a one-to-many relationship between the world and the map. This is advantageous for maintaining consistency as multiple (potentially conflicting) hypotheses about the structure of the environment are able to remain separate in the map until conflicts can be disambiguated, which can be arbitrarily delayed.

Each TSDF subvolume represents a small section of the environment and the total map is represented by the collection of all such subvolumes, such that

$$\Pi = \{\pi^p\}_{p=1}^N \tag{1}$$

where N is the number of currently constructed subvolumes. Each subvolume π^p has a local coordinate system M^p associated with it, parameterized by $\mathbf{T}_{GM^p} \in \text{SE}(3)$. The 3 operations possible on the subvolume collection are 1) addition of a new subvolume (Section IV-C) 2) modification of a subvolume's pose (Section IV-D), or 3) destruction of a subvolume by fusion of its content with another subvolume (Section IV-E).

B. Camera Localization

We utilize sparse, feature-based SLAM for providing the current sensor pose as well as updates to past poses. Typical TSDF systems have tended to rely on depth image registration for camera tracking, however in the context of this work sparse, feature-based localization offers two advantages: 1) modern sparse systems run efficiently on a CPU allowing their application to lightweight robotic platforms (see Section V-C), and 2) a sparse feature map, in which correlations between localization information and the environment are maintained, allows for probability-based decision making (see Section IV-D).

For this work we make use of ORB-SLAM2 [2] with the modifications proposed in Section IV-D. In the vein of the de-facto standard in modern visual slam, ORB-SLAM2 maintains a graph comprised of a set of keyframes and landmarks, with keyframe poses $\mathcal{K}$ and landmark positions $\mathcal{L}$ respectively. Camera tracking and mapping then are formulated as non-linear least squares optimizations of feature re-projection errors on the image plane,

$$\begin{aligned}\{\mathcal{K}_{\mathcal{O}}^{*}, \mathcal{L}_{\mathcal{O}}^{*}\} &= \underset{\mathcal{K}_{\mathcal{O}},\mathcal{L}_{\mathcal{O}}}{\operatorname{argmin}} \sum_{k\in\mathbb{K}} \sum_{l\in\mathbb{L}_k} \rho\left(\|\mathbf{e}_{k,l})\|_{\mathbf{\Sigma}}^{2}\right) \\ \mathbf{e}_{k,l} &= \mathbf{l}_{(\cdot)}^{l} - \boldsymbol{\pi}_{(\cdot)}(\mathbf{T}_{K^kG}(\mathbf{L}^l)),\end{aligned} \tag{2}$$

where the sets $\mathcal{L}_{\mathcal{O}} \subset \mathcal{L}$ and $\mathcal{K}_{\mathcal{O}} \subset \mathcal{K}$ are the optimized landmarks and keyframes respectively, and are subsets of the full map. The extent of these subsets determines the level of mapping fidelity. The set of keyframe indices is denoted $\mathbb{K}$ and the set of landmark indices observed by keyframe k is denoted $\mathbb{L}_k$. The cost function ρ is the Huber robust cost and $\mathbf{\Sigma}$ is the covariance matrix associated with the keypoint. The position of the l^{th} landmark is $\mathbf{L}^l \in \mathbb{R}^3$ and $\mathbf{T}_{K^kG} \in SE(3)$ parameterizes the pose of the k^{th} keyframe. ORB-SLAM2 allows for both monocular and stereo (depth) feature points $\mathbf{l}_{(\cdot)}^{l}$, such that $\mathbf{l}_m^l \in \mathbb{R}^2$ and $\mathbf{l}_s^l \in \mathbb{R}^3$, and $\boldsymbol{\pi}_{(\cdot)}$ is either the stereo or monocular projection function. In this work we limit ourselves to the scenario where *some* 3D observations are available to constrain the scale of the estimated map.

C. Volumetric Integration

Incoming image data is added to the collection by integrating arriving frames I^i and D^i into the currently active subvolume $\pi_A \in \Pi$. We first transform the estimated sensor pose, supplied by camera tracking, into the active subvolume frame M_A

$$\mathbf{T}_{M^AC^i} = (\mathbf{T}_{GM^A})^{-1} \oplus \mathbf{T}_{GC^i}, \tag{3}$$

where $\oplus$ denotes composition on SE(3). For integration of depth-frame data we use the open source TSDF toolbox, *voxblox*, introduced in [21] and extend it to provide interfaces for submapping.

Periodically, a new subvolume is created, marked as active and the last active subvolume is transferred to the collection, growing the map. Subvolumes are rigidly attached to the first keyframe after their creation such that $M^p \equiv C^k$ where k is the image index of the kth keyframe. We assume that data *within* a subvolume is consistent, and therefore our approach is to generate new subvolumes early and often, relying on the map fusion module to ameliorate the scalability cost of doing so. We generate a new active subvolume after either: the number of keyframes contributing to a subvolume reaches a maximum value, or there is significant change in the sparse map which is likely to cause a jump in the camera tracking.

The original voxblox system employed a computationally demanding frame integration method which was not suitable for real-time dense mapping on computationally-limited platforms. Also, unlike most TSDF libraries, voxblox operates on generic pointclouds rather than depth images. It also explicitly maps all free-space. This means that many of the strategies other libraries use for increasing runtime performance cannot be applied (chiefly image-space preprocessing and voxel projection). To overcome these issues we implement a new multi-threaded "fast" integration approach[3].

The fast integrator operates by terminating the ray casting early in two cases where the structure of an area is likely to have already been captured by other rays. The first case governs the maximum density of the starting points. Each point is first inserted into a "starting location" set that has twice the resolution of the standard voxel map. If a point already occupies this starting location the point is discarded. The second termination condition is to check if a ray is updating voxels that have already been updated by other rays in the same frame. This check is done by inserting the point into an "observed voxel" set. If a ray attempts to update two voxels in a row that have already been updated, it is deemed to be adding minimal information and is terminated. As the rays draw together as they are near the camera location this acts to terminate many rays shortly after the surface while still ensuring the vast majority of freespace voxels are allocated and at least partially updated. The speed of this approach depends on the speed of insertions into the sets. They are implemented as fixed-size, one-element-per-bucket, hash-sets, where collisions are resolved by discarding the old value. This allows synchronous lock-free insertions and reads using atomic compare-and-swap instructions. These sets are cleared at the end of each frame integration.

The time to integrate a frame depends on the structure of the input pointcloud, thus some frames may be significantly more expensive then others. To ensure real-time performance is always maintained, a maximum integration time was implemented that will terminate integration early if a time budget is exceeded.

D. Map Fusion

In this section we propose an approach for maintaining consistency of the map while also limiting its growth. Following a loop closure, camera localization provides an updated set of keyframe poses resulting from global optimization of the sparse map (2). The representation of the environment within each subvolume, as relative to its base frame, means that the collection is corrected simply by updating subvolume coordinate frames $\{M^p\}_{p=1}^N$ with their updated poses.

The system described thus far will produce a collection of subvolumes which increases in size linearly with trajectory length, meaning that during operation, even in bounded size environments, the map grows without bound. This limits the practical applicability of the proposed system. However, when revisiting a place, the collection is likely to contain multiple, potentially redundant, views of the same area. Our approach to limiting map growth is to identify these redundant views and fuse them, where doing so has a high probability of producing consistent results. We utilize the information contained in the sparse feature map to govern this process.

To identify subvolumes which contain redundant views, we propose to use a modified version of the *covisibility graph* [10]. During construction of the map we associate each

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

keyframe K^i with the subvolume to which it contributed. We then build a weighted graph $G = (V, E)$ where the vertex set V represents the subvolumes $\{\pi^p\}_{p=1}^N$, and the edge set E, with associated weights W, represent landmark covisibility information. Formally, vertex i and j have an edge of weight

$$W_{ij} = |\mathbb{L}_i \cap \mathbb{L}_j|, \tag{4}$$

where $\mathbb{L}_i$ and $\mathbb{L}_j$ are the sets of landmark indices observed by keyframes contributing to subvolumes π_i and π_j, and here $|\cdot|$ indicates the cardinality of the set. Subvolume pairs connected with an edge of high weight in this graph are likely to have viewed the same area, and are therefore passed as candidate pairs to the next stage.

At this stage we have a list of subvolume pairs which contain views of similar sections of the environment. However, the estimated poses of these subvolumes may contain significant localization error. Naive fusing of such pairs is likely to produce inconsistent results, and we would therefore rather maintain these views separately (as they currently exist) in the collection. Thus, before fusing subvolume pairs we determine a measure of the accuracy of their relative localization. Formally, given a candidate pair (π_i, π_j), with base-frames M_i and M_j, and associated with keyframes K_i and K_j, we define $q(i, j)$ the quality measure of their relative localization as

$$q(i, j) = 1/\|\mathbf{\Sigma}_{i|j}\|, \tag{5}$$

where $\Sigma_{i|j}$ is the covariance matrix associated with the conditional distribution $P(K_i|K_j)$. For this work the norm $\|\cdot\|$ is the 2-norm, which is proportional to the volume of an ellipsoid of constant probability density defined by $\Sigma_{i|j}$. Subvolumes pairs meeting a minimum value of q are fused together. In the remainder of this section we discuss how $\mathbf{\Sigma}_{i|j}$ is determined.

1) Extraction of the Subvolume Conditional Covariance: Given an initial guess of the optimization parameters $\theta = (\mathcal{K}, \mathcal{X})$, solving bundle adjustment proceeds by iteratively linearizing the graph and solving the linear system,

$$\mathbf{H}\theta = \mathbf{b}, \tag{6}$$

where the typically sparse matrix $\mathbf{H}$ and vector $\mathbf{b}$ are determined from the measurements, their Jacobians, covariances, and linearization point (see [22] for a detailed exposition). At convergence we are left with an approximation of the information matrix

$$\mathcal{I} \equiv \mathbf{H}^* \tag{7}$$

where $\mathbf{H}^*$ results from linearizing the graph at the convergence point θ^*. Inverting the (typically large) information matrix $\mathcal{I}$ to get covariance matrix $\mathbf{\Sigma}$ associated with the posterior distribution is computationally expensive and does not scale to practical problems. However, determining $\mathbf{\Sigma}_{i|j}$ for each subvolume-pair requires very few blocks of the full matrix $\mathbf{\Sigma}$; two diagonal blocks and a single off-diagonal block. We aim to perform the minimum computation required to extract these blocks, and proceed as follows.

We first marginalize out the landmarks from the graph using the so-called *Schur Compliment trick* to form the reduced camera matrix, here denoted $\mathcal{I}_{\mathcal{P}}$. This reduction can be performed very efficiently for problems of the size typically encountered in SLAM (see [23] full exposition on this method). From here we utilize the dynamic programming approach based on Cholesky decomposition of the information matrix introduced in [24]. This algorithm, calculates a requested covariance matrix element $\sigma_{i,j}$, as a function of a collection of other covariance matrix entries which follow a sparsity pattern which is a subset of the non-zero entries of the Cholesky factor $\mathbf{R}$.

We propose a modification to the original algorithm which increases its speed for calculating a small number of covariance matrix elements. The speed of the original algorithm depends predominantly on the number of intermediate entries required during computation of the desired element $\sigma_{i,j}$. The number of these elements needed is a product of the ordering of optimization variables prior to Cholesky decomposition. Using an ordering aimed at reducing overall fill-in, such as Approximate Minimum Degree (AMD) [25], can lead to widely varying and potentially large computation times when extracting a very small set of elements. We propose then to reorder the matrix $\mathcal{I}_p$ prior to Cholesky decomposition for *each* keyframe candidate pair for which we need covariance entries. By reordering with each pair, we are free to force the blocks associated with $\Sigma_{i|j}$ to appear last in the Cholesky factor. This leads to a vast reduction in the number of intermediate elements computed, and a corresponding increase in speed. To achieve this reodring we use a constrained version of AMD [25].

We pay for this ordering method in two ways. Firstly, constraining the position of some blocks of $\mathcal{I}$ is likely to increase the overall fill-in of the Cholesky factor. Secondly, we perform a decomposition of $\mathcal{I}_p$ for *each* subvolume-pair candidate, instead of just once for all candidates, as is done in the original algorithm [24]. In our testing, despite these drawbacks, there was a substantial benefit in taking the proposed approach.

E. Subvolume Fusion

We frequently are required to fuse subvolumes, both at the request of the map maintainance module, and for visualizing the current state of the subvolume collection. Given two subvolumes to be fused π_i and π_j with a reference frames M_i and M_j the transformation between them and global map is then found

$$\mathbf{T}_{M^i M^j} = (\mathbf{T}_{GM^i})^{-1} \oplus \mathbf{T}_{GM^j}. \tag{8}$$

We then transform the voxels in π_j, allocate new voxels in π_i where needed, and integrate their contributions using trilinear interpolation. For visualization of the whole manifold we repeat this process multiple times, fusing all subvolumes into π_1.

V. Results

The results presented in this section aim to validate our main contribution; a system for maintaining consistency while limiting map growth. We therefore evaluate the performance

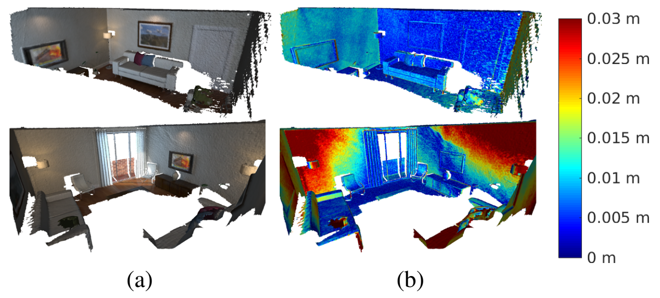


Fig. 3: Reconstructions from 2 *living_room* trajectories of the ICL-NUIM dataset [26] with (a) observed color and (b) colored by error. The median error across all 4 trajectories is 0.015 m for a voxel size of 0.02 m.

	RMSE (m)				
System	kt0	kt1	kt2	kt3	median
ElasticFusion	0.006	0.009	0.010	0.048	0.001
Voxblox (GT Poses)	0.010	0.017	0.014	0.011	0.013
Ours	0.011	0.024	0.016	0.013	0.015

TABLE I: Comparison of surface reconstruction RMSE for the proposed system, ElasticFusion [12] and voxblox [21] supplied with ground truth poses evaluated of the ICL-NUIM dataset [26]. The results indicate that our CPU-based system achieves surface reconstruction performance comparable to a state-of-the-art approach utilizing a GPU.

of our system in terms of the accuracy of reconstructed surfaces, and the size of the produced maps on several datasets. Finally, we validate our claim of applicability to lightweight robotic platforms by demonstrating the system in use on an MAV.

A. *Reconstruction Quality*

We evaluate the surface reconstruction accuracy of our system using the publicly available ICL-NUIM dataset [26]. This dataset provides RGB and depth images captured by a camera moving through a synthetic room. In addition, ground truth geometry is provided for evaluation of reconstructed surfaces. We reconstruct the scene using the proposed system as well the baseline system ElasticFusion [12], a top performing reconstruction system which makes heavy use of GPU-based computation. Reconstructions are performed on a desktop PC with an Intel Core i7-7820X CPU at 3.60 GHz, and a Nvidia Titan Xp GPU (donated by the Nvidia Corporation), which is used only by ElasticFusion. Fig. 3 shows reconstructed meshes produced by the proposed system for the kt0 and kt1 *living_room* sequences.

We align the surface reconstructions with the ground truth geometry and calculate the Root-Mean-Square Error (RMSE) between mesh vertices (or surfel centers) and the closest point on the ground-truth surface. In addition, we evaluate the surface reconstruction produced by voxblox [21] employing our fast integration method and supplied with ground truth camera poses. The performance of this combination is an indication of the minimum achievable error using the frame integration method proposed in Section IV-C. Both the proposed method and voxblox use a voxel size of 0.02 m. Table I shows the results of these evaluations.

Our system achieves a median RMSE error across the four sequences of 0.015 m. Given our voxel size of 0.02 m, an error less than this amount indicates the system is performing well. Furthermore, our system performs only slightly worse than the same integration system utilizing ground truth poses, which scores 0.013 m. The proposed method scores slightly worse than ElasticFusion on sequences *kt0*, *kt1*, and *kt2*, which scores a median error of 0.001 m. Given our design choices, which enable use on systems lacking a GPU this performance is to be expected. As we will show in Section V-C one advantage of our system is that it can be used on lightweight robotic platforms. Sequence *kt3* is the only sequence containing a loop closure. On this sequence the proposed system outperforms ElasticFusion. While one cannot judge a systems performance on a single dataset, this is an indication that our mechanism for maintaining map consistency is performing as expected.

B. *Map Maintenance*

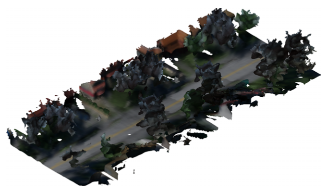
We evaluate the performance of the map-maintenance module for limiting map-growth, and analyze the effects of subvolume fusion on reconstruction accuracy. To obtain input data we use CARLA [27], an open-source driving simulator. The simulator produces photo realistic scenes of a car driving in a synthetic city, providing sensor output with realistic (non-ideal) camera effects, (noiseless) depth-maps, and ground-truth camera poses.

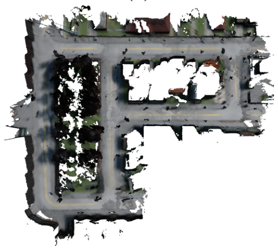
We evaluate our system using data from drives through two synthetic cities which include significant exploration, wide-baseline loop closures, and place revisiting. Our system is evaluated in two configurations: subvolume fusion turned ON and turned OFF. For comparison we evaluate a naive approach that uses voxblox for reconstruction and ORB-SLAM2 for camera tracking, but has no method to correct the dense map following loop closure. Lastly, we reconstruct the scene using ground truth camera poses and voxblox which represents a measurement of the minimum achievable error using our fast frame integration method and voxel size (0.5 m). In all configurations tracking and frame integration occurs at 10 Hz.

Carla does not provide ground-truth geometry for their maps, so we reconstruct our own "ground-truth" geometry with which we evaluate surface reconstructions. We generate a fine-grained reconstruction with voxblox using ground-truth poses, in a non real-time configuration which uses 0.25 m voxels and the more costly frame integration method described in [21]. This is an approximate measure of ground-truth structure, but for the remainder of this section we will assume errors are predominantly produced by the tested reconstruction systems.

Reconstructions from our system for two drives through different cities are shown in Fig. 4. Visible in the reconstructions is the structure of the synthetic cities, showing streets, houses, and trees. For quantitative evaluation we compute

(a)


(b)


(c)

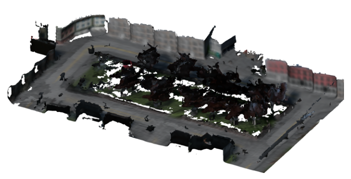
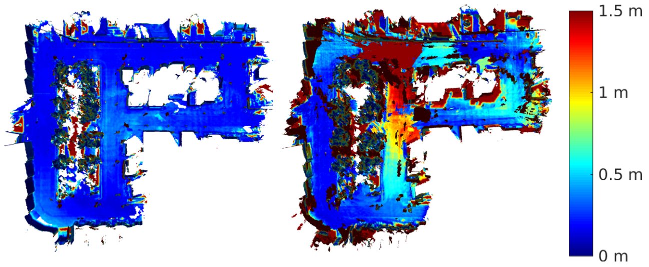

(d)

Fig. 4: Reconstructions produced by our system with data generated by a car driving through 2 synthetic cities [27]. Sub-figures (a) and (c) show bird's-eye views of the reconstructed areas, and (b) and (d) are closes ups of areas of these maps showing a tree-lined street and a square with house fronts visible respectively.



(a) (b)

Fig. 5: Reconstructions of an urban environment produced by the proposed system (a) and a naive approach not explicitly addressing map consistency (b). The reconstructions are colored by surface error. The area of high error in the center of reconstruction (b) is the site of several wide-baseline loop closures, indicating that error accumulates in these areas, if consistency is not explicitly maintained.

	RMSE (m)			Size (blocks)	
System	l0	l1	med.	l0	l1
Voxblox (GT Poses)	0.52	0.70	0.60	6315	4561
Voxblox (ORB-SLAM Poses)	2.12	2.06	2.09	7205	5240
Ours (subvolume fusion OFF)	0.59	0.77	0.68	28908	17856
Ours (subvolume fusion ON)	0.66	0.77	0.72	15873	12220

TABLE II: Comparison of surface reconstruction RMSE and final map size, in number of allocated voxel blocks, for the proposed system in two configurations (subvolume fusion ON and OFF), and voxblox [21] supplied with either ground-truth or ORB-SLAM2 [2] estimated poses. Data is collected during two drives (*l0* and *l1*) through two synthetic cities using CARLA [27].

RMSE reconstruction error (computed as in Section V-A) and final map size, in terms of allocated blocks. The results of this evaluation for each system configuration are shown in Table II.

For both trajectories, the results show that the lowest error is achieved by the system with access to ground truth poses as expected. This error score of median 0.60 m is approximately the dimension of one voxel, which is reasonable given the approximations described in Section IV-C. The system using voxblox with ORB-SLAM2 for camera tracking yields a median score of 2.09 m, 348% of the error score of the system using ground-truth poses. Fig. 5b shows the reconstructed mesh colored by error. The areas of high error in the center of the map are at the location of two wide-baseline loop-closures. It is expected that in these areas, where drift has accumulated maximally, error is the highest. The median reconstruction errors of our system in the two tested configurations (0.68 and 0.72 m) approach the error of the system with access to ground-truth poses, with increases in the error of 13% and 20% respectively, indicating a significant improvement over the naive approach.

The subvolume fusion system is effective in reducing the map size to 55% in *l0* and 68% on *l1* with respect to the system with no method of map fusion. The amount of compression achievable is heavily dependent on the path taken, as frequent revisiting will lead to areas of significant redundancy and produce good candidates for fusion. Similarly, the threshold in terms of the quality measure q at which fusion occurs will determine the number of subvolumes fused, and the incurred cost in terms of reconstruction error. There is a tradeoff to be made here, earlier fusion resulting from a lower q threshold will result in smaller maps, however will likely incur higher costs in terms of reconstruction accuracy, as submaps are fused before being optimally localized.

C. Platform Results

To demonstrate the applicability of the proposed framework to a lightweight robotic platform we reconstruct an industrial environment using a lightweight MAV. The hexacopter is based on a DJI F550 frame equipped with a Visual-Inertial (VI)-sensor [28] and an Intel RealSense D415 Depth Camera. A px4 autopilot performs low-level attitude stabilization, while high-level computation, including the proposed approach, was run entirely on-board and in real-time on an Intel NUC Core i7-7567U. For our purposes, the VI-sensor provided tightly synchronized stereo images from a pair of global shutter cameras, and is used for camera tracking at 10 Hz. The D415 provided colored pointclouds which are integrated into the map at the same rate. Calibration between the tracked camera and depth camera frames is performed offline using the *Kalibr* toolbox[4].

The modifications made to the voxblox integrator described in Section IV-C reduce the average time required to integrate depth data in this environment from 93ms to 25ms per frame, allowing the dense mapping to run in real-time while still providing sufficient CPU time for the other components of the proposed pipeline, as well as other software components related to control of the MAV.

Figure 1 shows reconstructions from two flights, *f0* and *f1*, as well as the mesh produced during from *f0* colored

[4]https://github.com/ethz-asl/kalibr

by subvolume membership and a view from the VI sensor during *f1*. During flight *f0* the MAV performed a 251 s trajectory which included two wide-baseline loop-closures and subsequent activation of the bundle adjustment and map fusion pipeline. In total 18 subvolumes were fused in flight *f0*, from a total of 34 created. The view from the onboard VI sensor shows pipe structures which are visible in the reconstruction Fig. 1b.

VI. Conclusion

In this paper we proposed a novel system for producing consistent dense 3D maps, which includes a systematic approach for limiting map growth, and runs in real time on a CPU. Our system is based on maintaining a collection of overlapping TSDF subvolumes which together constitute the global map. To limit map growth we propose a method for fusing redundant, well-localized subvolumes. Our evaluations on public, simulated and self-collected datasets, show that the proposed system is able to correct for significant inconsistencies that stem from imperfect camera localization. In our evaluation of the system on the ICL-NUIM benchmark, our system achieves a median RMSE of 0.015 m, which approaches the error score of a state-of-the-art reconstruction system requiring a high-power GPU. In a simulated urban environment we show the efficacy of our system for maintaining map consistency versus a naive approach, and that our subvolume fusion system is effective in significantly reducing the size of the resulting map. Finally, we deploy the proposed system to a MAV and generate a consistent reconstruction over two flights through an industrial area, demonstrating the system's applicability to lightweight robotic platforms. The ability to produce consistent and scalable dense maps, represents a step towards allowing robotic platforms to maintain detailed 3D maps in large scale environments.

References

[1] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 1872–1878, IEEE, 2015.

[2] R. Mur-Artal and J. D. Tardos, "Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras," *arXiv preprint arXiv:1610.06475*, 2016.

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

[4] 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*, pp. 127–136, IEEE, 2011.

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

[6] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, 2017.

[7] R. Wagner, U. Frese, and B. Bäuml, "Graph slam with signed distance function maps on a humanoid robot," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2691–2698, IEEE, 2014.

[8] A. Howard, "Multi-robot mapping using manifold representations," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, pp. 4198–4203, IEEE, 2004.

[9] P. Henry, D. Fox, A. Bhowmik, and R. Mongia, "Patch volumes: Segmentation-based consistent mapping with rgb-d cameras," in *3DTV-Conference, 2013 International Conference on*, pp. 398–405, IEEE, 2013.

[10] C. Mei, G. Sibley, and P. Newman, "Closing loops without places," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3738–3744, IEEE, 2010.

[11] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[12] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph," *Proc. Robotics: Science and Systems, Rome, Italy*, 2015.

[13] 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.

[14] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

[15] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundle-fusion: Real-time globally consistent 3d reconstruction using online surface re-integration," *arXiv preprint arXiv:1604.01093*, 2016.

[16] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An atlas framework for scalable mapping," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1899–1906, IEEE, 2003.

[17] J. Nieto, J. Guivant, and E. Nebot, "Denseslam: Simultaneous localization and dense mapping," *The International Journal of Robotics Research*, vol. 25, no. 8, pp. 711–744, 2006.

[18] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3d reconstruction with loop closure," in *European Conference on Computer Vision*, pp. 500–516, Springer, 2016.

[19] Q.-Y. Zhou, S. Miller, and V. Koltun, "Elastic fragments for dense scene reconstruction," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 473–480, 2013.

[20] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4475–4483, 2015.

[21] 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.

[22] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3607–3613, IEEE, 2011.

[23] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *European conference on computer vision*, pp. 29–42, Springer, 2010.

[24] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robotics and autonomous systems*, vol. 57, no. 12, pp. 1198–1210, 2009.

[25] T. A. Davis, "User guide for cholmod: a sparse cholesky factorization and modification package,"

[26] 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.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[28] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 431–437, IEEE, 2014.