Large-Scale and Drift-Free Surface Reconstruction Using Online Subvolume Registration

Nicola Fioraio[†] Jonathan Taylor[‡] Andrew Fitzgibbon[‡] Luigi Di Stefano[†] Shahram Izadi[‡]

[†]CVLab - CSE, University of Bologna
Viale Risorgimento, 2
40135 Bologna, Italy
{nicola.fioraio,luigi.distefano}@unibo.it

[‡]Microsoft Research
21 Station Road
Cambridge CB1 2FB, UK
{jota,awf,shahrami}@microsoft.com

Abstract

Depth cameras have helped commoditize 3D digitization of the real-world. It is now feasible to use a single Kinect-like camera to scan in an entire building or other large-scale scenes. At large scales, however, there is an inherent challenge of dealing with distortions and drift due to accumulated pose estimation errors. Existing techniques suffer from one or more of the following: a) requiring an expensive offline global optimization step taking hours to compute; b) needing a full second pass over the input depth frames to correct for accumulated errors; c) relying on RGB data alongside depth data to optimize poses; or d) requiring the user to create explicit loop closures to allow gross alignment errors to be resolved. In this paper, we present a method that addresses all of these issues. Our method supports online model correction, without needing to reprocess or store any input depth data. Even while performing global correction of a large 3D model, our method takes only minutes rather than hours to compute. Our model does not require any explicit loop closures to be detected and, finally, relies on depth data alone, allowing operation in low-lighting conditions. We show qualitative results on many large scale scenes, highlighting the lack of error and drift in our reconstructions. We compare to state of the art techniques and demonstrate large-scale dense surface reconstruction "in the dark", a capability not offered by RGB-D techniques.

1. Introduction

The availability of cheap depth cameras has transformed the face of 3D reconstruction. Over the past few years, researchers have demonstrated real-time high-quality 3D surface reconstructions [14, 10], even at large scales [4, 21, 15]. However, with these new capabilities have also come new challenges. One particular issue is of sensor drift which often occurs when scanning larger scenes. Here accumulation of frame-to-frame pose estimation errors results in distortions in the reconstructed 3D models.

To address these issues, researchers have explored a number of approaches for globally optimizing 3D models. The problem of bundle adjustment across RGB images is well understood [19] and some techniques combine depth data with RGB to directly build on top of this concept, adding depth data as an additional prior in the optimization step [9]. Others systems more explicitly detect loop closures and distribute the accumulated error across the underlying pose graph [18, 7].

More recently, Zhou and Koltun [23, 24] have demonstrated impressive globally optimized 3D surface models that extend the frame-to-model incremental tracking and reconstruction technique utilized in KinectFusion [14, 10, 6] to support drift-free reconstructions of larger scenes. In their first work, [23], they explicitly analyze pose trajectories to extract points of interest. These form sub-graphs around which errors can be distributed, and connections with other nodes formed. In their follow-up work, [24], non-rigid warps are computed across fragments of KinectFusion scans to support globally consistent large-scale models. Whilst resulting in compelling results, this body of work suffers from a number of issues. Firstly, these global optimization methods are expensive, requiring hours if not tens of hours to reconstruct a single model. Secondly, these approaches can require multiple passes of the depth data, meaning that the depth data needs to be stored, and reprocessed after global registration [23]. Thirdly, this approach uses an RGB-D SLAM system [7] in order to initially optimize the pose trajectory. This reliance on an initial bundle adjustment step essentially hinders the use of the system in challenging lighting conditions. This ultimately leads to an experience that cannot support online correction during scene scanning.

Similarly to [23, 24], Henry *et al*. [8] propose to globally align smaller volumes. These are, though, the result of an online segmentation of the scene in non-overlapping

locally planar surfaces, called *Patch Volumes*, which are then rendered to permit camera tracking by ICP. Instead, in terms of online correction of dense models, the work of Whelan *et al*. [20] employs a deformation graph connected to a pose graph. As poses are refined, vertices on the deformation graph are moved, and a global optimization that maintains local rigidity is performed on all other vertices and propagated to the mesh via linear blend skinning. This approach, however, relies on more explicit loop closures to be performed by the user, to distribute the accumulated error across the underlying pose graph, greatly limiting usage scenarios.

In this paper, we present a new technique for global optimization of large-scale volumetric 3D models. Unlike existing systems, our global optimization method can be performed in minutes directly on the subvolumes that are captured live, without the need to reprocess, re-fuse or store existing input data. Our method does not require an explicit loop closure in order to redistribute error, as such sub-chains of volumes can be optimized over in an online manner. Although very large exploration may require place recognition techniques, our approach can still handle common large scale scenarios without relying on visual features.

Compared to patch volumes [8], our subvolumes do not require an explicit segmentation step, being rather a byproduct of the real-time, low-drift, camera tracking algorithm. Also, our subvolumes *do* overlap, which enables on-line surface alignment, while in [8] a global optimization is performed only when a loop closure is detected by matching visual features between RGB images. Conversely, we do not carry out loop closure detection and can keep tracking at video rate while simultaneously optimizing subvolumes' poses. Instead, [8] stops camera tracking while optimizing patch volumes' poses, which requires typically from tens of ms to tens of seconds.

Finally our method does not rely on any RGB data, at any stage of the pipeline (even initialization). This means that our system can support large-scale corrected models, even in low-lighting conditions or other challenging conditions such as in complete darkness. We show qualitative results in many large scale scenes, highlighting minimal drifts and errors. We compare to state of the art techniques and demonstrate reconstruction that is on a par with these more expensive globally optimized methods [23], whilst outperforming real-time moving volume techniques [16, 21, 4, 15].

2. Preliminaries

We are given a sequence of depth images $\{D_t\}_{t=1}^{M}$ where $z = D_t(x, y)$ is the depth of the closest scene point on the ray through sensor pixel (x, y) in frame t, evaluated by bilinear interpolation for non-integral (x, y) and returning a sentinel value for locations where sensor data is not available. Associated with each depth frame is a transformation T_t encoding the sensor position and relating all measurements to a global 3D coordinate system. Given accurate estimates of T_t and sufficiently large memory, it would be possible to compute a global signed distance function (SDF) representation of the sequence and extract a surface representation [6]. In practice, however, the transformations are locally reliable, but subject to drift over time, and often there is insufficient memory to form the global SDF, even using octree-like compression techniques [22, 4, 17].

A key tool in this work is the weighted truncated signed distance function (TSDF) representation [6]. This is a pair of functions (F, W) from $\mathbb{R}^3 \to \mathbb{R}$ where $F(\mathbf{u})$ is the distance from $\mathbf{u}$ to the zero level set of F, while the weighting function $W(\mathbf{u})$ encodes a measure of confidence in the value of F at $\mathbf{u}$, which may be defined in a number of ways [3]. Typical representations of such functions are through trilinear interpolation on a voxel grid, from which the zero level set can be converted to a triangulated mesh through Marching Cubes [12]. At any point $\mathbf{u}$ on a TSDF F we define the normalized gradient of F

$$\widehat{\nabla} F(\mathbf{u}) \triangleq \frac{\nabla F(\mathbf{u})}{\|\nabla F(\mathbf{u})\|} \tag{1}$$

returning an undefined value where F is constant.

3. Problem Statement

Our proposal builds upon the successful KinectFusion framework [14, 10]. Accordingly, the model of the environment is a TSDF estimated in a 3D voxel grid and stored onto the GPU global memory as a linear array. However, similarly to [3], rather than relying on ICP alignment we track the camera by direct minimization of the distance of sensed points in the current frame to the surface using the TSDF.

To fuse depth measurements, every voxel is projected onto the image plane and compared to the sensed depth value:

$$\Delta_z(\mathbf{u}, t) = D_t\left(\pi\left(\mathrm{T}_t^{-1} * \mathbf{u}\right)\right) - \zeta\left(\mathrm{T}_t^{-1} * \mathbf{u}\right) \tag{2}$$

where π is a $\mathbb{R}^3 \to \mathbb{R}^2$ function which projects 3D points onto the image plane and ζ simply extracts the third coordinate of its argument, *i.e.* $\zeta(x, y, z) = z$. The TSDF is updated if $\Delta_z > -\delta$, where δ represents the truncation distance. Then, the updated TSDF $(F^{\mathrm{new}}, W^{\mathrm{new}})$ is given by

$$F^{\mathrm{new}}(\mathbf{u}) = \frac{F(\mathbf{u})\,W(\mathbf{u}) + \min\left(1, \frac{\Delta_z(\mathbf{u},t)}{\delta}\right)}{W(\mathbf{u}) + 1} \tag{3}$$

$$W^{\mathrm{new}}(\mathbf{u}) = W(\mathbf{u}) + 1 \tag{4}$$

Though delivering impressive, state-of-the-art surface reconstructions of small to medium sized workspaces, it is

well known that KinectFusion does not scale well to large-size environments, the reason being twofold. Firstly, dense estimation of the signed distance function requires an amount of GPU memory that limits applicability beyond medium-size rooms. Secondly, large exploratory sequences inevitably lead to drift in trajectory estimation that causes fusion of misaligned measurements into the global TSDF model. This, in turn, further deteriorates the tracking process.

4. Low-drift Local Modeling

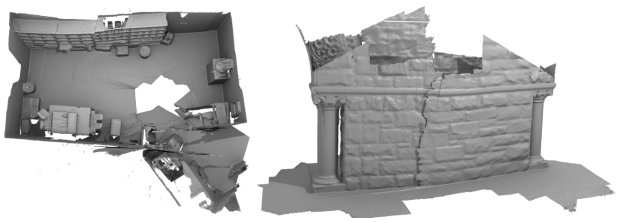
To overcome the above issues and be able to scan large environments with a KinectFusion-like framework, we adopt a simple variant of the moving volume approach described in [21, 16, 4]. Rather than relying on a fixed volumetric model, these methods fuse TSDF measurements into an active volume which is moved alongside the estimated camera trajectory. In our approach, we initially place the camera at the center of the active volume and, when the estimated camera translation differs from the volume center more than a given threshold, shift the active volume. For efficiency reasons, we allow only shifts by multiples of the voxel size, as proposed by Whelan *et al*. [21], and avoid resampling of the function as proposed by Roth and Vona [16]. However, unlike other approaches [4, 21], we do not stream voxels nor triangulated points to/from the host memory as a result of a volume shift. Instead, as it will be described later, we keep a global model of the environment made out of a collection of local TSDF models.

Based on the observation that drift is typically small over a short time interval, our active volume is built from the last K depth frames only. To avoid applying Eq. (3) and Eq. (4) K times at each step to construct the model, after fusion of the last TSDF measurements we simply *erode* the model by undoing the application of these equations for the $(t-K)$'th frame. Accordingly, the currently tracked frame D_t and its estimated pose T_t are pushed into a FIFO queue while D_{t-K} and T_{t-K} get popped and their associated measurements removed from the TSDF volume by applying

$$F^{\mathrm{new}}(\mathbf{u}) = \frac{F(\mathbf{u})\,W(\mathbf{u}) - \min\left(1, \frac{\Delta_z(\mathbf{u}, t-K)}{\delta}\right)}{W(\mathbf{u}) - 1} \quad (5)$$

$$W^{\mathrm{new}}(\mathbf{u}) = W(\mathbf{u}) - 1 \quad (6)$$

As long as fewer than K frames have been processed, no erosion is performed on the volume. After integration of the K'th frame, *i.e*. the queue is filled, the whole active volume is *copied* from the GPU to the host memory and saved as the first *subvolume*. Then, the erosion process runs continuously as the active volume integrates new measurements. We do, however, keep track of how many frames have been integrated and, every K such frames, we simply copy the active volume to create a subvolume before proceeding.



(a) (b)

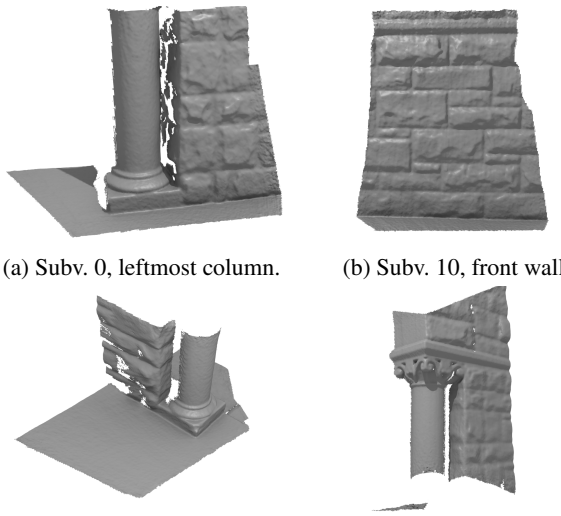
Figure 1: The inherent drift introduced in camera pose estimation by a moving volume method often leads to tracking failures (a) or inconsistencies due to fusion of misaligned TSDF measurements (b).

We believe that the proposed method in itself is valuable for creating a collection of low-drift local reconstructions of the explored environment. Indeed, even when closing the loop of a medium-size room, measurements integrated at the beginning of the sequence can be misaligned and a standard moving volume approach would either fail to track or attempt to fuse misaligned TSDF measurements (see Fig. 1), inevitably corrupting the final reconstruction. Instead, the *erosion* procedure allows us to continuously track the sensors pose against a locally reliable model of the local space. This reduces the possible sources of error that may yield failures with any specific tracking method adopted. Moreover, the active volume on the GPU memory corresponds to the fusion of a particular subset of camera frames, *i.e*. the last K ones, so it is an ideal candidate for retrieving low-drift local reconstructions, *i.e*. subvolumes, as suggested by Zhou *et al*. [24]. Unlike [24], though, our approach is fully integrated in the tracking pipeline and, as its complexity depends only on the number voxels, it can operate in real-time.

To this end, we reduce the time needed to copy a complete volume from the GPU to the host memory by subdividing the volume grid into blocks of $16 \times 16 \times 16$ voxels. Then:

- blocks including only null voxels, *i.e*. having zero weight, are immediately discarded;
- blocks made out truncated distance values are marked as *empty* and only the maximum weight is retained;
- blocks containing at least one non-truncated distance value are completely downloaded.

Since not all the blocks are taken, this representation exploits the sparsity of the TSDF, but the final structure is not designed for fast data queries. Therefore, on the host side we create a multi-resolution grid by further grouping blocks into $4 \times 4 \times 4$ bricks. Then, if a brick includes only empty blocks, again only the maximum weight is saved. In Fig. 2 we show triangulated meshes extracted from typical subvolumes with



(a) Subv. 0, leftmost column. (b) Subv. 10, front wall.

(c) Subv. 20, rightmost column. (d) Subv. 45, leftmost column.

Figure 2: A subvolume is a TSDF produced by fusing K depth frames. Here we show surfaces extracted from subvolumes obtained on the *stonewall* sequence (*cfr*. the leftmost image in Fig. 8).

$K = 50$. The corresponding final reconstruction given by registration and blending (see Sec. 5 and 6) is depicted in the leftmost image of Fig. 8.

5. Subvolume Registration

As described in the previous section, every K frames a new subvolume (F_j, W_j) is produced and saved on the host main memory together with its pose, referred to hereinafter as V_j. Such poses are initially translation-only (cfr. center image of Fig. 3) and, although each subvolume enjoys low-drift "locally", the collection of subvolumes resulting from a large exploratory motion would inevitably exhibit drift and present misalignments. As shown in Fig. 3, we handle this issue by globally optimizing subvolumes' poses each time a new subvolume is spawned, while non-rigid deformations are addressed through a novel volume blending scheme (see Sec. 6). It is noteworthy to point out that the result of this optimization is not required by the camera tracking module, which can therefore keep operating in real-time and push new subvolumes into a shared buffer allowing pose optimization to be run as a concurrent process.

The proposed optimization framework is inspired by the well-known ICP method [5, 2]. A cost function is set up as follows:

1. for each subvolume, the zero-level set is extracted so to attain a set of points together with their normals;
2. for each subvolume, we consider its bounding box and find those other subvolumes having an overlapping bounding box;
3. for each point in the current subvolume's point set, we search for correspondences by following the gradient of the distance function in overlapping subvolumes (see Subsec. 5.1);
4. each valid correspondence introduces a point-to-plane distance constraint in the optimization problem [5] (see Subsec. 5.2);
5. if a pose appears as underconstrained, at least one pose-to-pose error term is added to ensure global consistency (see Subsec. 5.2);

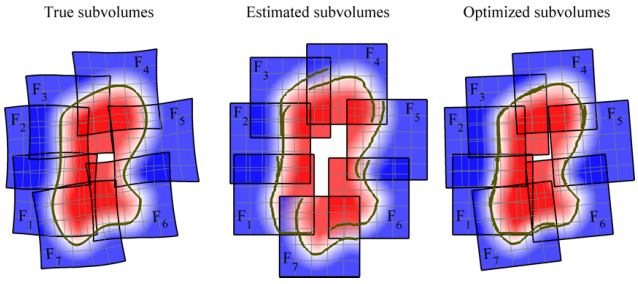


Figure 3: Subvolume $F_j : \mathbb{R}^3 \to [-1, 1]$ is a truncated signed distance function associated with a global transformation V_j. The estimated subvolumes are initially axis-aligned (*i.e*. V_j is translation-only), while the optimized transformations are general isometries. Due to sensor noise, though, *true* subvolumes are often subjected to a non-rigid transformation.

Then, we optimize for subvolumes' poses by minimizing a suitable cost function. The pose of the lastly created subvolume is kept fixed during the optimization to get the final estimates in the same reference frame as the tracked camera. Accordingly, the next subvolume can be immediately placed in a position consistent with previous subvolume's pose, provided, of course, that camera tracking has not undergone a failure. The cost function is minimized until convergence, then a new set of correspondences is computed and the cost minimized again. The whole match/optimize process is iterated until final convergence. The next two subsections delineate the point matching stage and the optimization problem, respectively.

5.1. The Correspondence Set

Given a subvolume (F_j, W_j), with zero-level set points $\left\{\mathbf{p}_i^{(j)}\right\}$ and computed normals $\mathbf{n}_i^{(j)} = \widehat{\nabla} F_j\left(\mathbf{p}_i^{(j)}\right)$, we define the minimal bounding box enclosing all its valid voxels and find the subset $S^{(j)}$ of subvolumes having overlapping bounding boxes. Then, for each point $\mathbf{p}_i^{(j)}$ we traverse the candidate set and, for every $k \in S^{(j)}$, compute the distance

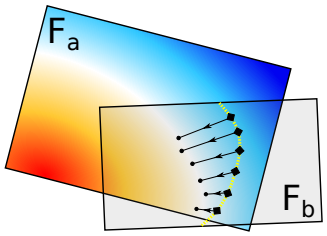


Figure 4: From each point (black diamonds) on the zero-level set (yellow) of F_b, we move in the direction of $\widehat{\nabla} F_a$ to find a match (black circles).

function and its gradient at $\mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{p}_i^{(j)}$ by trilinear interpolation (see Fig. 4). If defined, a match is set for this point as follows:

$$\mathbf{q}_k^{ji} = \mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{p}_i^{(j)} - F_k\left(\mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{p}_i^{(j)}\right) \widehat{\nabla} F_k\left(\mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{p}_i^{(j)}\right). \quad (7)$$

The procedure is then repeated for all sampled points in all subvolumes.

5.2. The Optimization Problem

Given the correspondence set, a cost function is built and minimized. For each point pair $\left(\mathbf{p}_i^{(j)}, \mathbf{q}_k^{ji}\right)$ we note that, while we have no guarantees about the shape of F_k at $\mathbf{q}_k^{ji}$, we already have estimated the normal vector for $\mathbf{p}_i^{(j)}$ in F_j. Therefore, we derive a point-to-plane constraint as:

$$e_k^{ji} = \left(\mathbf{p}_i^{(j)} - \mathrm{V}_j^{-1} * \mathrm{V}_k * \mathbf{q}_k^{ji}\right) \cdot \mathbf{n}_i^{(j)}. \quad (8)$$

The minimization problem is finally written as

$$\underset{\{\mathrm{V}_1,\ldots,\mathrm{V}_N\}}{\arg\min} \sum_j \sum_i \sum_k \left\| e_k^{ji} \right\|^2 \quad (9)$$

and solved by Ceres [1] with a Levenberg-Marquardt method [11, 13].

It could happen that the correspondence set for a particular subvolume, say F_h, is empty or it has a very low cardinality, leading to an underconstrained problem or to a poor pose estimate. In that case, pose-to-pose constraints can be introduced in order to reinforce the estimate given by camera tracking. Up to two error terms are added, *i.e.* one connecting to the previous subvolume, if any, and one connecting to the following subvolume, if any. If $\mathrm{Z}_{h-1,h}$ is the tracker estimate which maps points from F_h reference frame to F_{h-1} reference frame, the associated pose-to-pose error term is given by:

$$e^{h-1,h} = \Phi \log\left(\mathrm{Z}_{h-1,h} * \mathrm{V}_h^{-1} * \mathrm{V}_{h-1}\right), \quad (10)$$

where $\log : \mathrm{SE3} \to \mathfrak{se}3$ is the logarithmic map and Φ a 6×6 stiffness matrix. We empirically set Φ to the identity matrix.

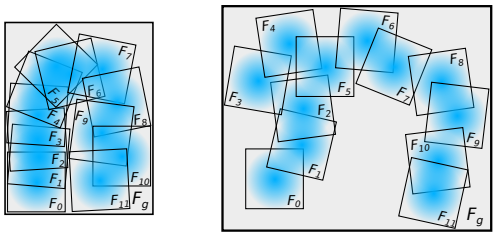


Figure 5: Building a single global volume by subvolume averaging forces to traverse the whole collection even for zero-weight voxels. Therefore, evaluation of the global volume shown on the left is faster than the one on right including the same subvolumes with the same extent but different poses.

6. Volume Blending

In the previous sections we have described how subvolumes are created and their poses optimized to obtain a correct reconstruction of a large environment. However, the different data sources of the subvolumes generate slightly different positions of the distance function's zero-level set, mainly due to the limited noise filtering achievable through only K integrations. Hence, multiple overlapped surfaces and artifacts, such as non-rigid deformations, may appear. To address this issue, a possible approach would be the estimation of a global volume F_G extending across the space covered by all the subvolumes. We could define such global TSDF as the fusion of all the depth frames through the subvolumes' TSDF trilinear interpolation:

$$F_G(\mathbf{u}) = \frac{\sum_j F_j\left(\mathrm{V}_j^{-1} * \mathbf{u}\right) W_j\left(\mathrm{V}_j^{-1} * \mathbf{u}\right)}{\sum_j W_j\left(\mathrm{V}_j^{-1} * \mathbf{u}\right)}. \quad (11)$$

However, resampling the subvolumes into a new global volume requires traversing the whole collection even though the queried position is out of all the bounding boxes, *i.e.* it is a null voxel. Indeed, this can be discovered only by querying every single subvolume. Moreover, the size of the global volume is directly dependent on the camera path, so that an extensive exploration might produce a large amount of null voxels (see Fig. 5). Therefore, computation time is affected more by the relative position of subvolumes rather than by their number and extent.

To address and overcome this problem, we avoid computation of a global volume and instead propose to *blend* each volume with its neighbors. Let (F_j, W_j) be a subvolume and $S^{(j)}$ the subvolumes having bounding boxes that overlap with its own. Then at each voxel $\mathbf{u}$ the TSDF is updated as:

$$F_j(\mathbf{u}) = \frac{\sum_k F_k\left(\mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{u}\right) W_k\left(\mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{u}\right)}{\sum_k W_k\left(\mathrm{V}_k^{-1} * \mathrm{V}_j * \mathbf{u}\right)}, \quad (12)$$

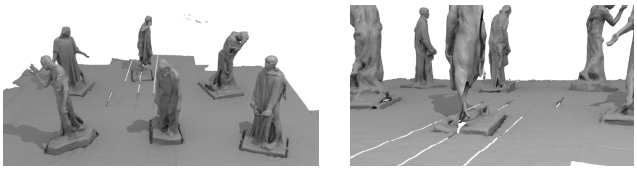


Figure 6: Reconstruction of the *burghers* sequence by the implemented moving volume method.

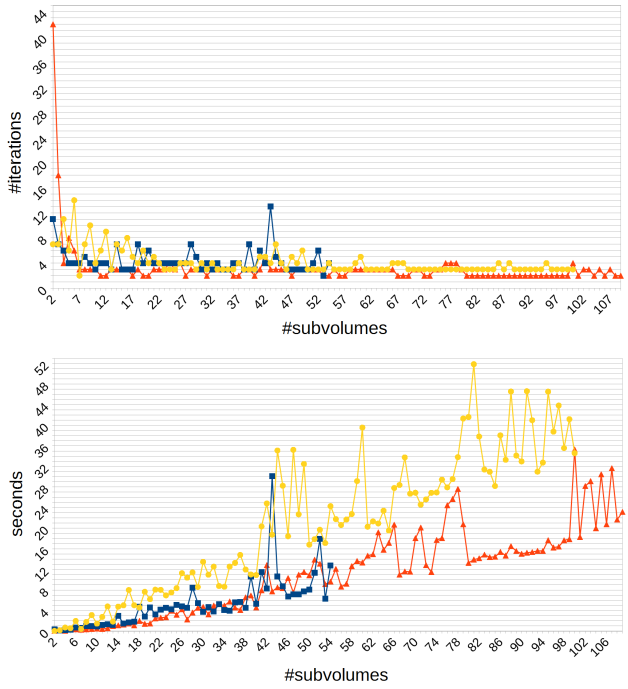


Figure 7: Optimization statistics for *stonewall* (blue squares), *copyroom* (red triangles) and *bookshop1* (yellow circles).

where $k \in S^{(j)}$. Also, for higher speed, we skip the computation if $W_j(\mathbf{u}) = 0$ or $F_j(\mathbf{u}) = 1$. On one hand, if the subvolumes do not overlap, the time required is zero. On the other hand, if all the subvolumes were overlapped, the complexity would be quadratic in the number of subvolumes. However, we experimentally found that the computation time is strongly dominated by the total number of voxels, rather than the amount of overlapping, so that the blending approach is roughly five to ten times faster than global volume resampling.

It is worth pointing out here that the volume blending process is only aimed at improving the final surface reconstruction, but neither the camera tracking nor the subvolume mapping stages require this feature to work properly.

Table 1: Statistics for the dataset used in our experiments.

Sequence	#frames	#subvolumes
stonewall	2700	55
copyroom	5490	110
lounge	3000	61
burghers	11230	225
dark room	4841	97
bookshop1	5000	101
bookshop2	5700	115

Table 2: Our method (top table) requires significantly less time than Zhou and Koltun [23] (bottom table) for both pose optimization and final surface reconstruction. *POI det.*: Points-Of-Interest detection; *Reg.*: fragment registration; *Opt.*: pose optimization.

Sequence	Pose Optimization			Volume Blending
	Min	Max	Average	
stonewall	0.154s	30.136s	5.388s	3m 3s
copyroom	0.085s	35.297s	11.238s	10m 30s
lounge	0.084s	15.149s	5.830s	4m 3s
burghers	0.047s	5m 4s	1m 30s	22m 27s
dark room	0.064s	45.882s	12.416s	7m 17s

Sequence	Pose Optimization				Depth Map Integration
	POI det.	Reg.	Opt.	Total	
stonewall	1m	17m	1h 54m	2h 12m	21m
copyroom	1m	14m	52m	1h 7m	47m
lounge	1m	12m	16m	29m	40m
burghers	5m	40m	10m	55m	2h 1m
dark room	–	–	–	–	–

7. Results

We devised several qualitative experiments to assess the quality of our reconstructions comparatively to existing methods, such as, in particular, an implementation of a moving volume approach with voxel streaming and the method proposed in [23]. As for the RGB-D data, we have considered four sequences from the dataset introduced by Zhou and Koltun [23], namely *stonewall*, *copyroom*, *lounge*, *burghers*, and three sequences captured by ourselves: *dark room*, *bookshop1*, *bookshop2*. The number of frames in each sequence is reported in Tab. 1 together with the number of subvolumes spawned by our method.

Fig. 1 and Fig. 6 show reconstructions attained by the moving volume method. As already pointed out in Sec. 4, both Figures highlight how the accumulated drift causes misalignments showing up as artifacts in the final reconstruction. For the sake of comparison, the results achieved by our method can be observed in Fig. 8, 9 and 10a.

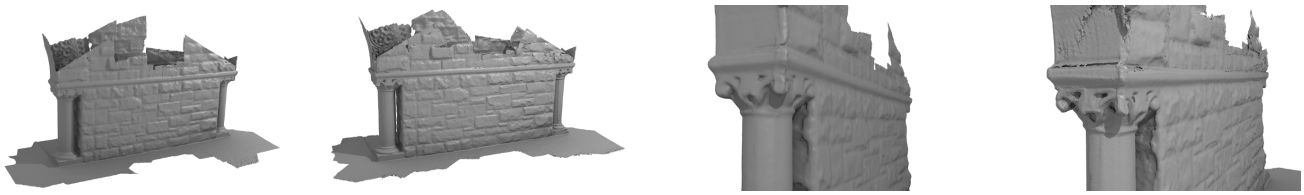


Figure 8: Reconstruction of *stonewall* sequence given by our approach (on the left) and by [23] (on the right). While preserving high frequency details, some artifacts remain in [23], *e.g*. above the capital.

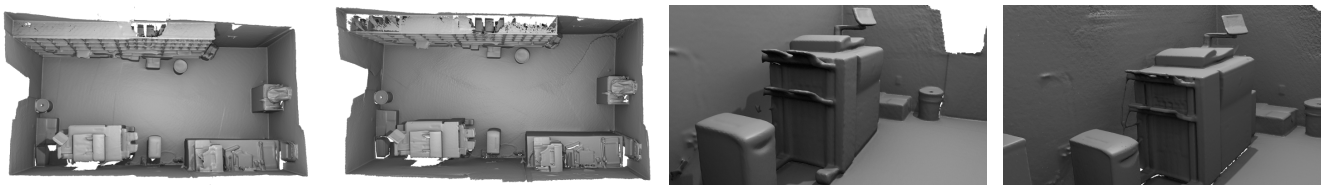


Figure 9: Reconstruction of *copyroom* sequence given by our approach (on the left) and by [23] (on the right).

As regards the comparison to [23], Fig. 8 and 9 point out that their reconstructions tend to include finer details but, at the same time, more high-frequency artifacts, such as, *e.g*., the wall above the pillar in Fig. 8 and the ground floor in Fig. 9. Overall, the quality of the reconstructions provided by the two methods are comparable, although, unlike our proposal, [23] uses an off-line post-processing method relying on an external tracking and mapping engine, *i.e*. RGB-D SLAM [7], rather than a full-fledged on-line SLAM system. Therefore, low-light conditions or even complete darkness would not allow [23] to operate due to the lack of descriptive visual features to match. In contrast, our system is able to recover the full geometry of *dark room* explored in complete darkness, as shown in Fig. 10b. We do not show any RGB frame from *dark room* because they are almost entirely black.

Besides, we report three additional reconstructions. The *lounge* sequence in Fig. 10a is a straightforward sequence, allowing us to show the quality reached by our method. The *bookshop1* and *bookshop2* sequences in Fig. 10c deal with mapping large environments.

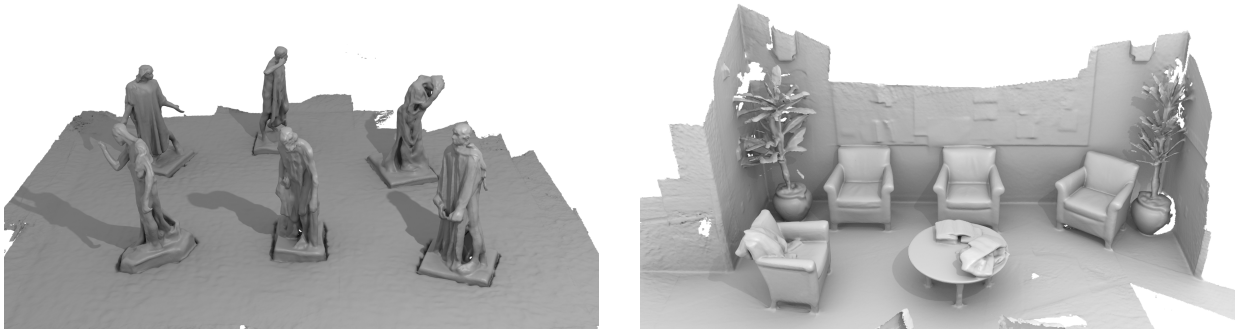
To analyze the impact on performances of the optimization step, we consider the number of required iterations and the time spent for pose optimization as the number of subvolumes increase. Fig. 7 reports these data for three sequences, *i.e*. *stonewall*, where the camera moves around an object, *copyroom*, with the camera performing a complete loop inside a room, and *bookshop1*, an exploratory sequence. As shown, the number of iterations (top chart) is somehow constant and low, whereas the time spent (bottom chart) seems to increase linearly with the number of optimized subvolumes' poses. Moreover, the optimization step usually requires less than a minute, which is far less than the time needed by other high quality reconstruction methods, *e.g*. [23].

Finally, Tab. 2, provides a detailed comparison between the computational cost of our method and that of [23]. It is worth highlighting that our method triggers pose optimization upon subvolume spawning, and each such run has a different computational effort. Conversely, [23] carries out a single global optimization based on the initial guess provided by RGB-D SLAM [7]. However, even the maximum time required by our pose optimization is much less than that required by [23], the latter being also unable to process the *dark room* sequence due to the lack of RGB data. It is also worth pointing out that our volume blending step is from 5 to 10 times faster than the global volume resampling deployed by [23].

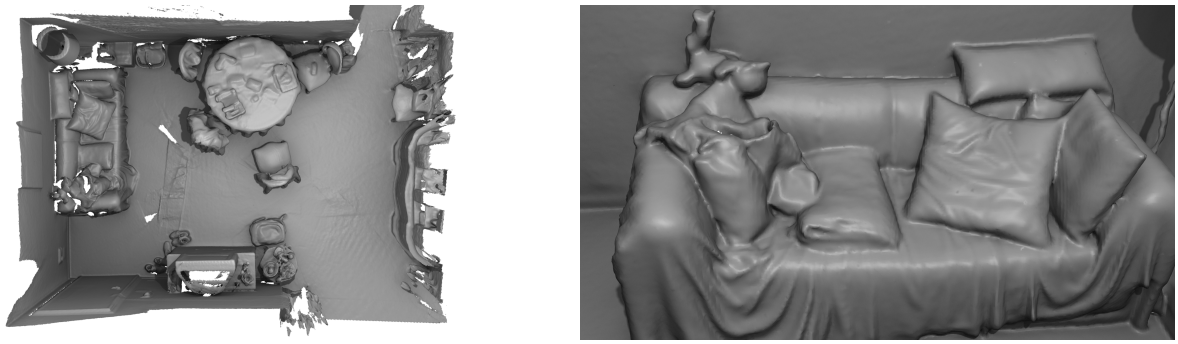
8. Conclusions And Future Work

We have presented a novel technique for addressing large-scale globally optimized surface reconstruction and tracking. Our tracking framework builds upon the well-known KinectFusion algorithm. However, while integrating new measurements, old frames are eroded, thus keeping a low-drift volume built from the last K frames only. The *erosion* process has a complexity dependent on the number of voxels, so the whole tracking algorithm still runs in real-time. The choice of K may turn out critical in a real-world scenario, as it defines what is to be considered a *low-drift* local reconstruction. A small K could hinder the noise attenuation brought in by integration of multiple frames, whereas a large K could introduce into the volume misaligned TSDF measurements. Clearly, the optimum value should be related to the camera motion and we plan to better investigate on this issue in our future work.

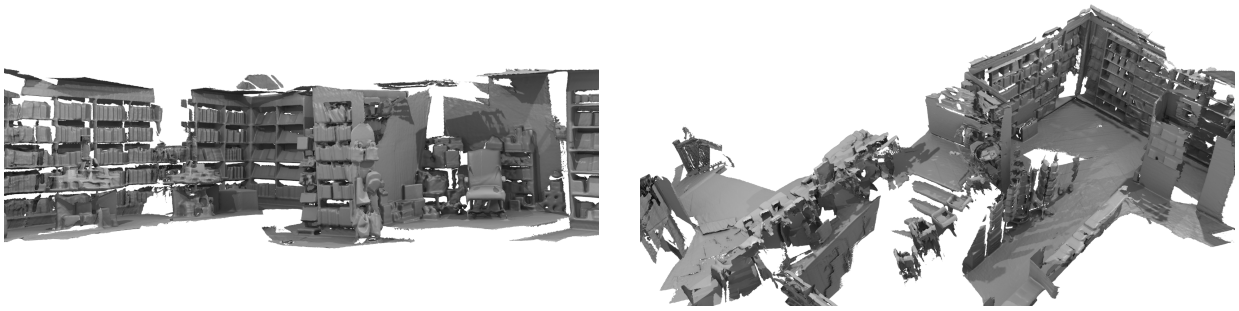
Online subvolumes are the second key concept of this paper. As shown, they provide a seamless and almost drift-free dense representation of the environment and enable large



(a) *Burghers* (left) and *lounge* (right) sequences.



(b) Top view (left) and a detail (right) from the *dark room* sequence.



(c) *Bookshop1* (left) and *bookshop2* (right) sequences.

Figure 10: Final reconstructions achieved by our method.

scale surface reconstruction through a simple but effective iterative solution. The proposed method works online, and allows fused volumes to continuously correct based on new measurements. However, each run requires an amount of time which increases as new subvolumes are spawned, because a global optimization is always performed. A valuable alternative in the future could be to solve only local problems, including building cost functions for a subset of the subvolumes' poses, thus limiting the maximum optimization time.

Finally, we have proposed to finely reconstruct the explored 3D models by blending together the subvolumes, instead of re-sampling a new global TSDF model. On one hand, we have shown how this technique yields state-of-art surface reconstruction at a fraction of the cost required by previous proposals. On the other hand, a blended subvolume could be used in the future to populate the active volume with reliable TSDF measurements which could then reduce the drift introduced by the camera tracking algorithm.

In all, this provides new capabilities for drift-free online corrected 3D surface representations at large scales, which link together the advantages of volumetric techniques for cheap and high quality depth map fusion, with more tractable global optimization methods, which remove the reliance on heavy offline computation, preprocessing of the raw depth data or RGB data.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] P. J. Besl and H. D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence (PAMI), IEEE Trans. on*, 14(2):239–256, 1992.

[3] 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.

[4] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transaction on Graphics (TOG)*, 32(4), July 2013.

[5] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10:145–155, April 1992.

[6] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH Intl. Conf.*, New Orleans, LA, USA, 1996.

[7] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3d mapping with an RGB-D camera. *IEEE Transactions on Robotics (T-RO)*, 2013.

[8] P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch volumes: Segmentation-based consistent mapping with rgb-d cameras. In *International Conference on 3D Vision - 3DV*, pages 398–405, June 2013.

[9] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *Experimental Robotics (ISER), Int'l Symp on*, 2010.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology*, Santa Barbara, CA, USA, October 2011.

[11] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.

[12] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Intl. Conf.*, 21(4):163–170, July 1987.

[13] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[14] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), IEEE and ACM Int'l Symp. on*, pages 127–136, Washington, DC, USA, 2011.

[15] M. Niessner, M. Zollhofer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6), November 2013.

[16] H. Roth and M. Vona. Moving volume KinectFusion. In *British Machine Vision Conference (BMVC)*, September 2012.

[17] F. Steinbruecker, J. Sturm, and D. Cremers. Volumetric 3d mapping in real-time on a cpu. In *Robotics and Automation (ICRA), IEEE Int'l Conf. on*, Hongkong, China, 2014.

[18] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *Computer Vision (ICCV), IEEE Int'l Conf. on*, pages 2352–2359, Los Alamitos, CA, USA, 2011.

[19] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[20] T. Whelan, M. Kaess, J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, November 2013.

[21] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, July 2012.

[22] M. Zeng, F. Zhao, J. Zheng, and X. Liu. A memory-efficient kinectfusion using octree. In *Computational Visual Media (CVM), Int'l Conf. on*, pages 234–241, Beijing, China, 2012.

[23] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transaction on Graphics (TOG)*, 32(4), July 2013.

[24] Q.-Y. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *Computer Vision (ICCV), IEEE Int'l Conf. on*, pages 473–480, Sydney, NSW, Australia, 1-8 December 2013.