A two-stage strategy for real-time dense 3D reconstruction of large-scale scenes

Diego Thomas and Akihiro Sugimoto

National Institute of Informatics, Tokyo, Japan.
{diego_thomas,sugimoto}@nii.ac.jp

Abstract. The frame-to-global-model approach is widely used for accurate 3D modeling from sequences of RGB-D images. Because still no perfect camera tracking system exists, the accumulation of small errors generated when registering and integrating successive RGB-D images causes deformations of the 3D model being built up. In particular, the deformations become significant when the scale of the scene to model is large. To tackle this problem, we propose a two-stage strategy to build in details a large-scale 3D model with minimal deformations where the first stage creates accurate small-scale 3D scenes in real-time from short subsequences of RGB-D images while the second stage re-organises all the results from the first stage in a geometrically consistent manner to reduce deformations as much as possible. By employing planar patches as the 3D scene representation, our proposed method runs in real-time to build accurate 3D models with minimal deformations even for large-scale scenes. Our experiments using real data confirm the effectiveness of our proposed method.

1 Introduction

The ability to build an accurate 3D model of a large-scale indoor scene from an RGB-D image sequence is of great interest because of various potential applications in the industry. Such 3D models can be used for, for example, visualisation tasks, robot navigation, or space organisation. With the recent advent of consumer RGB-D sensors such as the Microsoft Kinect camera, the task of 3D scanning in indoor environments became easy and a number of techniques to map all acquired RGB-D data into a single 3D model have been proposed (*e.g.* KinectFusion and extensions ([11], [12], [13], [16]). As a consequence, the quality of generated 3D models has been remarkably improved for relatively small scenes (*e.g.* a space of a few square meters). Nevertheless, handling large-scale scenes still remains a challenging research area.

While early work such as KinectFusion [11] strives to build detailed small-scale 3D scenes from noisy RGB-D data in real-time, significant effort ([6], [15], [17]) has been recently devoted to allow detailed reconstruction of large-scale scenes by employing more compact yet accurate 3D scene representations. However, though fine details can be locally obtained, the 3D model as a whole becomes deformed because of the accumulation of inevitable drift errors in camera pose estimation over a long sequence. To correct such deformations, elastic registration [19] between fragments of the whole scene can be used at the cost of significant computational overhead, which prevents from real-time application. On the other hand, for fast correction of deformations, graph optimisation [6] can be used where the scene is segmented into several (ideally) undeformed fragments that are connected to keyframes via visibility constraints. Keyframe pose constraints enforced over different keyframes are also used there. Once a loop is detected, the graph is optimised to re-position all fragments. Because one fragment is represented by a single vertex in the graph and the geometry of each fragment need not be modified, this process is executed quickly. By using visibility and keyframe pose constraints, drift errors are re-distributed uniformly (when all edges are weighted equally) along the estimated camera path to close the loop. However, this is not reasonable because the drift errors, in general, do not uniformly arise (parts of the scene with many features induce small errors while parts with less features do large errors).

In this paper, we also employ graph optimisation to correct deformations. One of our main contributions is the introduction to new geometric constraints for better geometrically consistent redistribution of drift errors. The new geometric constraints represent (ideally) exact relative positions in the real world between different objects in the scene. We reason that a 3D model is deformed when the relative positions between different objects that compose the scene do not match their exact relative positions. Therefore, to remove deformations from the 3D model, we strive to ensure that the relative positions of all objects in the 3D scene are as close as possible to their exact ones.

In order to generate such useful geometric constraints, two main difficulties arise: (a) how to ensure that each object itself is not deformed? And (b) how to obtain exact relative positions between different objects in the scene? One solution to reduce possible deformations is that we reconstruct each object in the scene from a short sequence of RGB-D images (as proposed in [19]). Multiple instances of a single object may then be generated for a long sequence, and how and when to merge them becomes crucial (as raised in [6]). We propose not to merge multiple instances of the same object but rather we introduce identity constraints that represent their relative positions. This allows more flexibility in re-positioning all objects in the 3D scene.

Overall, we propose a two-stage strategy for 3D modeling using RGB-D cameras where the 3D scene is represented using planar patches [15] for compact and structured, yet accurate, 3D modeling. The first stage, called local mapping, aims at generating, from short-time RGB-D image sequences, accurate small-scale structured 3D scenes with minimal deformations and geometric constraints in real-time. We here introduce a new model, called semi-global model, for tracking the camera pose and merging incoming RGB-D data in real-time with state-of-the-art accuracy. The second stage[1], called global mapping, on the other hand, aims at organising all the results obtained in the first stage to keep geometric consistency of the whole scene. To remove deformations from the global model

[1] The second stage runs in a parallel process to enable real-time 3D reconstruction.

as much as possible, we employ fragment registration and graph optimisation. Namely, we align an input local model to multiple rigid fragments of the global model, generate identity constraints between different instances of the same object and use the graph optimisation framework to satisfy geometric constraints, identity constraints, keyframe pose constraints and visibility constraints. Our proposed method enables us to (1) recover fine details of the scene thanks to employing the frame-to-global-model framework, (2) achieve real-time processing thanks to using our semi-global model for tracking and fusing RGB-D data, and (3) reconstruct large-scale indoor scenes without deformations thanks to the capability of re-positioning the different objects in the scene with respect to their exact relative positions.

2 Related work

In the last few years, much work has been proposed to fuse input RGB-D data into a common single global 3D model.

Newcombe *et al.* [11] proposed KinectFusion: a system to build implicit 3D representations of a scene from an RGB-D camera at an interactive frame-rate. The implicit representation consists of a Truncated Signed Distance Function (TSDF) that is discretized into a volume covering the scene to be reconstructed. The TSDF is recorded into a regular voxel grid, which requires large amount of memory usage. This limits the practicability of the method for large-scale scenes. Then, much work has been done on extending the method for large-scale applications [13], [16]. On the other hand, the use of non-regular volumetric grid has been studied for compact TSDF representations. Zeng *et al.* [17] and Chen *et al.* [5] proposed an octree-based fusion method. Neibner *et al.* [10] proposed to use hash tables to achieve significant compression of the volumetric TSDF. However, because of the accumulation of errors generated when registering and integrating incoming RGB-D data, the global volumetric TSDF inevitably becomes deformed when applied to a large-scale scene. Moreover, correcting deformations in the volumetric TSDF is tedious and, in general, multiple passes over the whole sequence of RGB-D images are required. This becomes a critical limitation for large-scale applications.

Zhou *et al.* [18] proposed to use local volumes around points of interest, and Henry *et al.* [6] proposed to segment the scene into planar patches and to use 3D TSDF volumes around them to represent the 3D scene. In [6], a pose-graph optimisation for loop closure was proposed where each vertex of the graph is a planar patch connected to one or multiple keyframe(s). When a loop is detected, it gives constraints on the relationship between patches and keyframes. Optimising the graph to meet the constraints reduces deformations. When the 3D scene represented by old patches is deformed due to drift errors, however, the rigid alignment is not reliable anymore. Moreover, a critical question about when to merge overlapping patches in a loop is left open. Also, the processing time drops drastically due to procedures for maintaining planar patches. Thomas *et al.* [15], on the other hand, proposed a method that requires only three 2D images, called attributes, for each planar patch to model the scene, which allows more compact representation of the scene. Though they achieved significant compression of the 3D scene, no discussion was given about how to deal with deformations that arise when applied to large-scale scenes.

Meilland *et al.* [9] proposed a method to reconstruct large-scale scenes by using an image-based keyframe method. Multiple keyframes are recorded along the camera path and merged together to produce RGB-D images from any viewpoint along the camera path that can be used for robust camera tracking or visualisation. Once the whole image sequence is processed, a cloud of point or a mesh is generated by running a voxel-based reconstruction algorithm over the set of keyframes. Loops may be closed using keyframes, and drifts in the estimated camera pose can be corrected accordingly. However, uniformly redistributing drift errors over the camera path is not reasonable at large scale because the distribution of the errors is, in general, not uniform.

Zhou *et al.* [19] proposed to break the whole RGB-D image sequence into short-time subsequences. The KinectFusion algorithm is applied to each short-time subsequence and then local meshes of the scene are generated. After all subsequences are processed, they run an elastic registration algorithm that combines rigid alignment and non-rigid optimisation for accurate alignment and drift-error correction. This method achieves state-of-the-art accuracy in 3D reconstruction. However, the computational cost is expensive and the elastic registration is a post-process, which prevents the method from real-time applications.

Differently from the above methods, this paper uses graph optimisation framework with new geometric constraints built between planar patches representing different objects and identity constraints built between planar patches representing the same object to accurately and efficiently correct deformations of the global model.

3 Proposed method

We break the whole sequence of RGB-D images into short subsequences (in our experiments we used subsequences of 100 frames) and take a two-stage strategy (Fig. 1), local mapping and global mapping, to build a large-scale 3D model.

We use planar patches [15] as the 3D scene representation. As shown in [15] and [6], however, using sets of planar patches in the frame-to-global-model framework significantly drops computational speed. In the local mapping, we thus introduce a new model, called semi-global model, for camera tracking and data integration. The attributes of all planar patches (*i.e.* Bump image (that represents local geometric disparity), Color image and Mask image (that represents confidence of measurements)) are then built on-line from the semi-global model. As we will show later, this allows us to keep real-time performance with state-of-the-art accuracy. Geometric constraints are generated from the first frame of each short subsequence. This is because the exact relative positions between different objects in the scene are reliably estimated only from a single image (deformations usually arise after merging multiple RGB-D images).

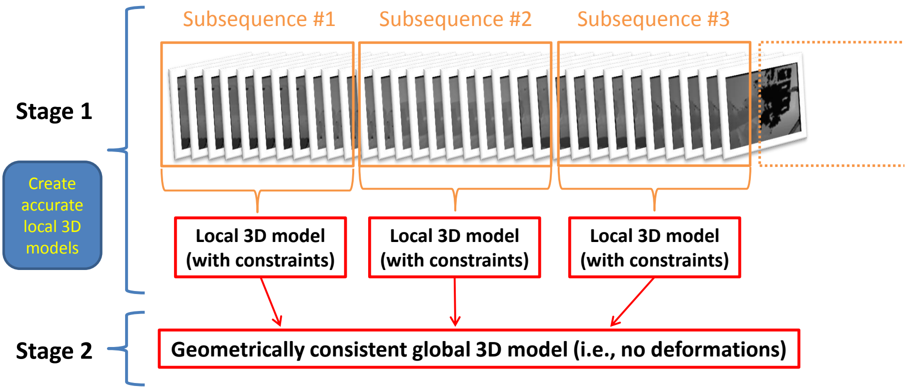
Stage 1
Create accurate local 3D models
Subsequence #1
Subsequence #2
Subsequence #3
Local 3D model (with constraints)
Local 3D model (with constraints)
Local 3D model (with constraints)
Stage 2
Geometrically consistent global 3D model (i.e., no deformations)

Fig. 1: Overview of our proposed strategy. The first stage generates local structured 3D models with geometric constraints from short subsequences of RGB-D images. The second stage organises all local 3D models into a single common global model in a geometrically consistent manner to minimise deformations.

In the global mapping, on the other hand, we build a graph where each vertex represents either a patch generated from the local mapping or a keyframe, and edges represent keyframe pose constrains [6], visibility constraints [6], our introduced geometric constraints, or identity constraints of patches over keyframes. Over the graph, we keep all similar planar patches (*i.e.* multiple instances of the same object) without merging, which allows more flexibility in re-positioning all the planar patches. Moreover, the geometric constraints enable us to redistribute errors more coherently with respect to the 3D geometry of the scene. This is crucial as in general, drift errors derived from camera tracking do not uniformly arise.

3.1 Local mapping

To build local 3D models and generate their associated geometric constraints, two challenges exist: (1) how to build the structured 3D model in real-time and (2) how to generate "good" geometric constraints (*i.e.* constraints that represent the exact relative positions between all planar patches). We tackle these challenges with keeping in mind accuracy, real-time processing and minimal deformations. Fig. 2 illustrates the pipeline of the local mapping.

To tackle the first challenge, the frame-to-global-model framework has proven to be successful. However, the set of planar patches itself cannot be directly used as the global model. This is because (1) planar patches representation may not be as dense as the input RGB-D images (non co-planar parts of the scene are not represented), and (2) rendering all planar patches at every incoming frame is time consuming and unwise (as many points on a patch may disappear from the current viewing frustum and rendering such points is useless). Therefore, we

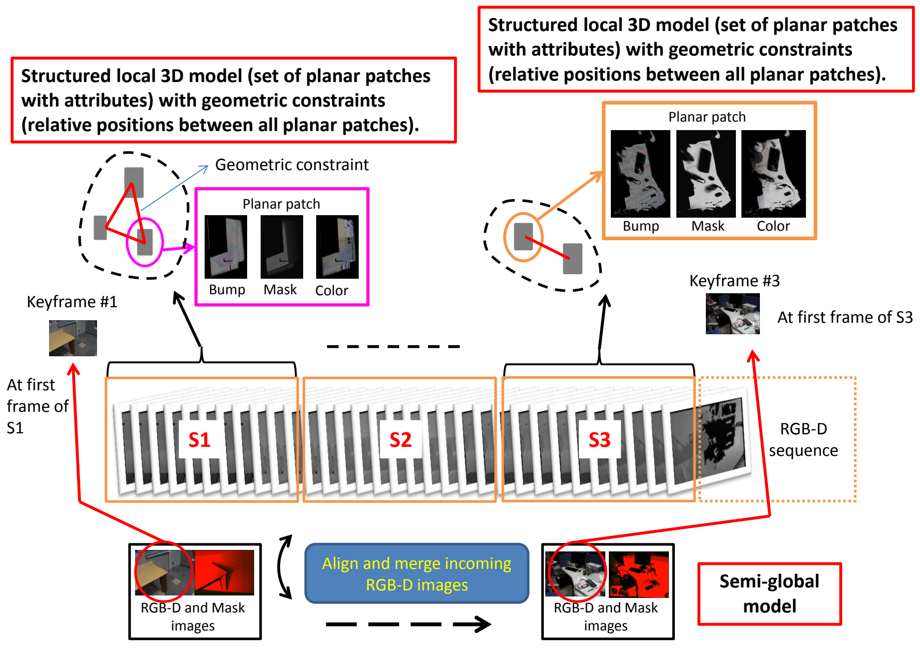
Structured local 3D model (set of planar patches with attributes) with geometric constraints (relative positions between all planar patches).
Geometric constraint
Planar patch
Bump
Mask
Color
Keyframe #1
At first frame of S1
Structured local 3D model (set of planar patches with attributes) with geometric constraints (relative positions between all planar patches).
Planar patch
Bump
Mask
Color
Keyframe #3
At first frame of S3
S1
S2
S3
RGB-D sequence
RGB-D and Mask images
Align and merge incoming RGB-D images
RGB-D and Mask images
Semi-global model

Fig. 2: Local mapping. A semi-global model is used for tracking the camera and merging incoming RGB-D images. Local 3D models are built dynamically by projecting (at every new incoming frame) the current semi-global model onto the attributes of different planar patches that are detected from the semi-global model at the first frame of each subsequence.

need to employ another 3D model to perform accurate, robust and fast camera tracking and data integration. The standard TSDF volumetric representation is not good because it requires too much amount of GPU memory to update the attributes of all planar patches.

Semi-global model. Our semi-global model is defined as a pair of RGB-D and Mask images. The semi-global model is initialised with the first frame of the whole RGB-D image sequence. To generate predicted RGB-D and Mask images, we use OpenGL capability with the natural quadrangulation given by the organisation of points in the 2D image.

For each pixel $(u, v) \in [1 : n-1] \times [1 : m-1]$, a quad is created if Mask values of its 4-neighbor pixels are all positive and if the maximum Euclidean distance between all vertices at these pixels is less than a threshold (we used a threshold of 5cm in the experiments). We render the mesh three times with (1) depth information, (2) color information and (3) mask information, to generate the predicted RGB-D and Mask images. Fig. 3 illustrates how to obtain the predicted RGB-D and Mask images. The predicted RGB-D image is used to

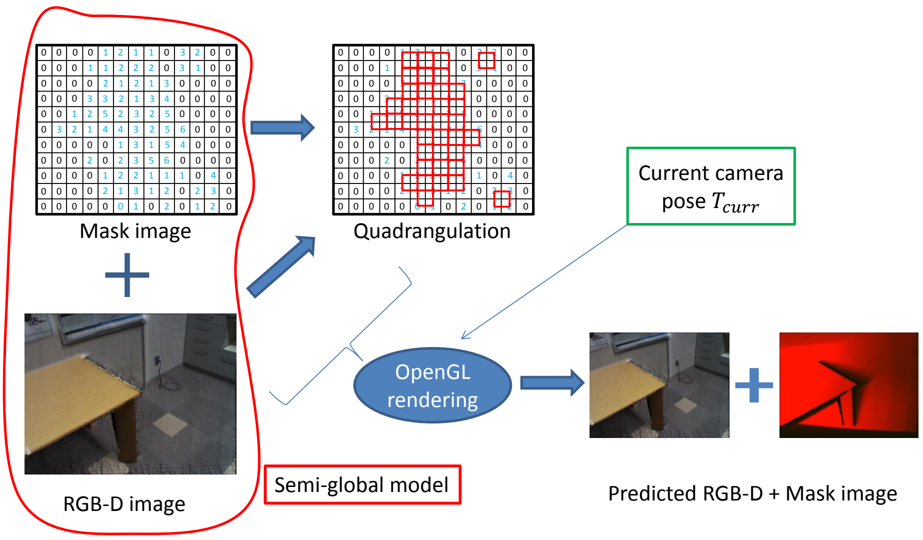

Fig. 3: Generating of predicted RGB-D and Mask images. We use OpenGL capacities to render RGB-D and mask images corresponding to the estimated current camera pose.

align the incoming RGB-D image using the technique described in [6], which combines both geometric and color information. The predicted Mask image is used to merge aligned data with a running average, as proposed in [15]. The semi-global model is then renewed to the newly obtained pair of merged RGB-D and Mask images. For every subsequence, the RGB-D image of the semi-global model at the first frame of the subsequence is selected as a keyframe and recorded on the hard drive.

Using the semi-global model enables us to keep real-time and accurate performance because it quickly generates high-quality dense predicted RGB-D images. Our proposed 3D model is semi-global in that all invisible points from the current viewing frustum of the camera are lost (but recorded in the set of planar patches that form the local 3D model).

As for the second challenge, a standard approach would be to incrementally build the local 3D model by adding new planar patches on the fly as they are being detected (as done in [15] or [6]) and to generate geometric constraints at the end of each subsequence by computing their relative positions. The relative positions (*i.e.* geometric constraints) between planar patches detected at different times can be significantly different from the exact relative positions due to drift errors. Therefore we propose to compute the geometric constraints only from a single RGB-D image (the first of each subsequence). This is justified by the fact that a single RGB-D image has few deformations and thus the relative positions between objects detected in a single image are close to the exact ones. We detect the set of planar patches from the first RGB-D image of each subsequence and fix it over the subsequence.

Local 3D model. For each subsequence, we segment the first predicted RGB-D image into several planar patches (together with their attributes) that form our local 3D model. As proposed in [15], for each planar patch we use as its attributes three 2D images: a Bump image, a Color image and a Mask image. At every input frame, the attributes of each planar patch are updated using the semi-global model as follows.

Each point $\mathbf{p}$ of the semi-global model is projected into its corresponding planar patch. The values of Bump, Color and Mask images at the projected pixel are all replaced by those of $\mathbf{p}$ if the mask value of $\mathbf{p}$ is higher than that at the projected pixel (as explained in [15]).

Whenever all the process for a short subsequence of RGB-D images is finished, we record a generated local 3D model (*i.e.* the set of planar patches with their attributes), as well as the current keyframe (*i.e.* the first predicted RGB-D image) and geometric constraints on the planar patches.

3.2 Global mapping

The objective of the second stage is to fuse all local 3D models generated in the first stage into a single geometrically consistent global 3D model with minimal deformations. The main problem here comes from the accumulation of registration errors. If the global model becomes deformed, the rigid alignment of new local 3D models to the (deformed) global model does not work. Moreover, merging planar patches representing the same object will drastically corrupt the global model. Even loop closure correction ([6]) does not work anymore, because planar patches themselves become deformed. Applying non-rigid alignment of local 3D models to the global model ([19]) is not good either, because of expensive computational cost.

Our main ideas for the global mapping are (1) to always align new local 3D models to undeformed subsets (called fragments) of the global model and (2) to introduce new constraints (called identity constraints) at each successful rigid alignment, rather than merging planar patches representing the same object. This allows more flexibility in re-organising the global model. An identity constraint represents the relative position between planar patches representing the same object in the scene that come from different local 3D models. A graph optimisation framework (namely, the g2o framework [4]) is then used to guarantee geometric consistency of the global model (thus reducing deformations). Note however that to obtain the final global 3D model in real-time, merging planar patches is executed separately depending on the object and updated on-line as the relative positions between all planar patches are being optimised.

Global graph. We organise all planar patches and keyframes into a global graph that represents all constraints. Each vertex of the global graph represents either a planar patch or a keyframe, and each edge represents either a geometric constraint, an identity constraint, a keyframe pose constraint or a visibility constraint.

For each i^{th} subsequence Seq_i, we denote by K_i the keyframe of Seq_i, by $Pset_i = \{P_j^i\}_{j \in [1:m_i]}$ the set of m_i planar patches built with Seq_i. For each planar patch P_j^i we denote by $BBox_j^i$ the projected 3D bounding box of the 3D points in P_j^i into the plane equation $(\mathbf{e}_1^{i,j}, \mathbf{e}_2^{i,j}, \mathbf{n}^{i,j}, d^{i,j})$ of P_j^i, where $\mathbf{n}^{i,j}$ is the normal of the plane, $d^{i,j}$ the distance of the plane from the origin and $(\mathbf{e}_1^{i,j}, \mathbf{e}_2^{i,j}, \mathbf{n}^{i,j})$ form an orthonormal basis. $\mathbf{pt}_j^i$ is the lower left corner of $BBox_j^i$. Each planar patch P_j^i is represented by a vertex in the graph to which a 3D transformation matrix V_j^i is assigned such that:

$$V_j^i = \begin{bmatrix} \mathbf{e}_1^{i,j} & \mathbf{e}_2^{i,j} & \mathbf{n}^{i,j} & \mathbf{pt}_j^i \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Each keyframe K_i is represented by a vertex in the graph to which its pose (matrix) T_i (computed during the local mapping) is assigned.

To each edge $e = (a, b)$ that connects two vertices a and b, with transformation matrices T_a and T_b (respectively), we assign a 3D transformation matrix T_{edge} that defines the following constraint:

$$T_b T_a^{-1} T_{edge}^{-1} = \mathrm{Id},$$

where Id is the 4×4 identity matrix. We detail in the following how to compute T_{edge} for each type of edges.

1. **Geometric constraints.** For each subsequence Seq_i we generate edges that connect all planar patches with each other. For each edge (P_j^i, P_k^i) we assign matrix $T_{i,j,k}^{Geo} = V_k^i (V_j^i)^{-1}$ (*i.e.* the relative position) that defines the geometric constraint between P_j^i and P_k^i.
2. **Identity constraints.** Identity constraints are defined by the relative positions between planar patches representing the same object in the scene (by abuse we will say that the planar patches are identical). Every time a set of patches $Pset_i$ is registered to another set of patches $Pset_j$, we generate edges that represent identity constraints. We first identify identical planar patches as follows. Two planar patches P_k^i and P_l^j are identical if and only if $\|d^{i,k} - d^{j,l}\| < \tau_1$, $\mathbf{n}^{i,k} \cdot \mathbf{n}^{j,l} > \tau_2$ and overlap$(P_k^i, P_l^j) > \tau_3$, where $\cdot$ is the scalar product, overlap(P_k^i, P_l^j) is a function that counts the number of overlapping pixels between P_k^i and P_l^j and τ_1, τ_2 and τ_3 are three thresholds (*e.g.* 10cm, 20° and 3000 points respectively in the experiments). For every pair of identical planar patches (P_k^i, P_l^j) we generate an edge, and assign to it matrix $T_{i,k,j,l}^{Id} = V_l^i (V_k^j)^{-1}$ that defines the identity constraint.
3. **Keyframe pose constraints [6].** For every two successive subsequences Seq_i and Seq_{i+1}, we generate an edge (K_i, K_{i+1}), and assign to it matrix $T_{i,i+1}^{Key} = T_{i+1} T_i^{-1}$ that defines the keyframe pose constraint between K_i and K_{i+1}.

4. **Visibility constraints [6].** For each subsequence Seq_i we generate edges so that K_i is connected with any planar patch in $Pset_i$. To each edge (K_i, P^i_j), we assign to it matrix $T^{Vis}_{i,j} = V^i_j T^{-1}_i$ that defines the visibility constraint between P^i_j and K_i.

On-line update of global graph. The global graph grows every time a local 3D model is generated. Once a local model $Pset_i$ comes, we first add vertices for each planar patch P^i_j and a vertex for the keyframe K_i. We then add edges so that K_i and K_{i-1} (if $i > 1$) are connected, K_i and any entry in $Pset_i$ are connected and all planar patches in $Pset_i$ are connected with each other (they represent the keyframe pose constraint, visibility constraint and geometric constraint, respectively).

Second, we perform fragment registration of $Pset_i$ with multiple fragments of the global graph to include $Pset_i$ into the global model while minimising deformations as much as possible. We first identify a set of keyframes each of which is sufficiently close to K_i, and divide the set into fragments so that each fragment consists of only successive keyframes.

We define the set S_i of the neighboring keyframes of K_i in the global graph as follows:

$$S_i = \{K_j \mid d(K_i, K_j) < \tau_d \text{ and } \alpha(K_i, K_j) < \tau_\alpha\},$$

where $d(K_i, K_j)$ and $\alpha(K_i, K_j)$ are the Euclidean distance between the centres of two cameras, and the angle between the two viewing directions of the two cameras (respectively) for the i^{th} and j^{th} keyframes (in the experiments, we set $\tau_d = 3$m and $\tau_\alpha = 45°$).

We then break the set S_i into p fragments: $S_i = \{F^1_i, F^2_i, ..., F^p_i\} = \{\{K_{s_1}, K_{s_1+1}, K_{s_1+2}, ..., K_{s_1+t_1}\}, ..., \{K_{s_p}, K_{s_p+1}, K_{s_p+2}, ..., K_{s_p+t_p}\}\}$ where for all $j \in [1 : p-1]$, $s_{j+1} > s_j + t_j + 1$. We reason that the local 3D models corresponding to successive keyframes are registered together in a sufficiently correct (*i.e.* undeformed) manner to perform rigid alignment with $Pset_i$. This is not the case if the set of keyframes contains non-successive keyframes.

We then align $Pset_i$ with each of $\{F^j_i\}_{j\in[1:p]}$. We align $Pset_i$ with a fragment F^j_i ($j \in [1 : p]$) as follows. Let us denote by $Pset^j_i$ the set of all planar patches connected to a keyframe in F^j_i.

We first initialise the transformation by using matches of SIFT features [8] between K_i and K_{s_j}. We use the RANSAC strategy here to have a set of matched features. If the number of matched features is greater than a threshold (we used a threshold of 30 in our experiments), then the transformation is initialised by the matched features, it is set to the identity transformation otherwise.

After the initialisation, we applied the GICP algorithm [14] to align $Pset_i$ and $Pset^j_i$. Because of millions of points in $Pset^j_i$, searching for the closest points in a standard manner (using k-d trees for example) is not practical at all. Instead, we borrow the idea of the projective data association algorithm [3], which can be run efficiently on the GPU. Namely, for each planar patch $P_l \in Pset_i$ and for each pixel (u, v) in the Bump image of P_l, we project the 3D point $\mathbf{pt}(u, v)$ into all planar patches in $Pset_i^j$. We then identify the closest point of $\mathbf{pt}(u, v)$ as the point at the projected location with the minimum Euclidean distance to $\mathbf{pt}(u, v)$ and with angle between the two normals sufficiently small (we used a threshold of 40° in the experiments). If the minimum distance is greater than a threshold (we used a threshold of 5cm in the experiments), then we regard that $\mathbf{pt}(u, v)$ has no match.

After aligning $Pset_i$ with $Pset_i^j$ as seen above, we generate edges that represent the identity constraints, and then optimise the global graph using the g2o framework [4]. We fix the poses of all planar patches in $Pset_i$ and those of all planar patches in $Pset_{s_j}$. We also fix those of K_{s_j} and K_i. The poses of the other vertices are then optimised with respect to all constraints before proceeding to align $Pset_i$ with the next fragment (if there is one). After each optimisation, all planar patches are positioned such that (1) the relative positions between the planar patches in the same local model are close to the exact ones (this reduces the deformations between different objects in the scene), and (2) the relative positions between planar patches representing the same object in the scene are close to the relative positions obtained with fragment registration (this reduces the deformations within each object in the scene).

4 Experiments

We evaluated our proposed method in several situations using real data. All scenes were captured at 30 fps. We used a resolution of 0.4cm for attribute images in all cases. The CPU we used was an Intel Xeon processor with 3.47 GHz and the GPU was a NVIDIA GeForce GTX 580. Our method runs at about 28 fps with a live stream from a Kinect camera.

Figure 4 (a) shows results obtained by our method using data RGBD_DATASET _FREIBURG3_LONG_OFFICE_HOUSEHOLD [2]. In this data-set, captured with a Kinect camera, the camera turns around a small scale scene, which requires loop-closure operations for 3D reconstruction. We compared results by our method with those shown in [19]. Note that we used the mocap camera trajectory in Fig. 4 (b) [7] as the ground truth. The circled parts on the the right side of Fig. 4 (a) and on the left side of Fig. 4 (b) focus on the corner of the central wall to attest the ability of each method to correct deformations. In the squared boxes, we zoomed in a smooth surface (blue box) and thin details (red box) to attest the capability of each method to generate fine details of the 3D scene.

Figure 4 (a) shows that our method succeeded in building a geometrically consistent and accurate 3D model. As we can see in the circled parts, our method significantly outperformed in accuracy the Extended KinectFusion method [13]: to the part where there is only one corner, two corners of the wall are incorrectly reconstructed by [13] while ours reconstructed it as one corner. This is because [13] does not employ any loop closure. In the squared boxes, we can observe that the amount of noise (spiky effects in the blue box) obtained by our method is similar to that by [18] or the ground truth (the mocap trajectory). On the other

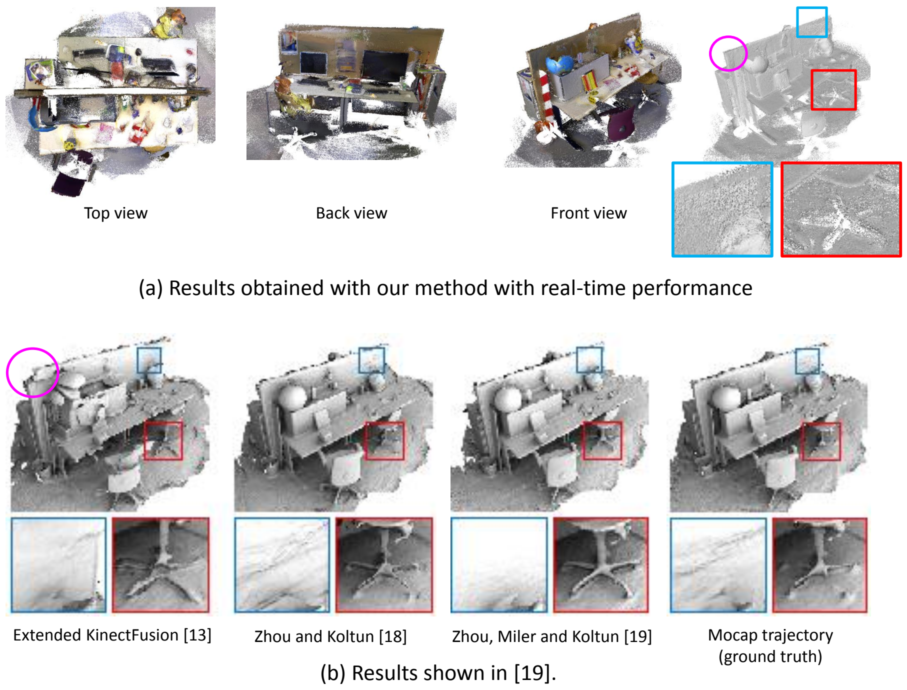
Top view
Back view
Front view
(a) Results obtained with our method with real-time performance
Extended KinectFusion [13]
Zhou and Koltun [18]
Zhou, Miler and Koltun [19]
Mocap trajectory (ground truth)
(b) Results shown in [19].

Fig. 4: Results obtained with data LONG_OFFICE_HOUSEHOLD. The circled areas show the advantage of using loop-closure algorithms. Without loop-closure, the scene was deformed. The blue boxes show smoothness of the reconstructed surface. The image should be uniformly white when illuminated (spiky effects comes from noise). The red boxes show ability to reconstruct thin details. Unfortunately, when using planar patches representation, some details were lost.

hand, [19] achieved better accuracy than ours: the surface is smoother on the wall (blue box). Note that due to employing planar patches representation, our method generated a 3D model that is less dense than that obtained by [18] or [19]: parts of the chair in the red box are missing with our method. We remark that, the results by [18] or [19] are off-line while ours are on-line.

Figure 5 (a) shows results obtained by our method using data COPYROOM and data LOUNGE (captured with an Xtion Pro Live camera) available at [1]. We compared the results obtained by our method with the state-of-the-art results [19] on these two datasets. The dataset COPYROOM consists of 5490 RGB-D images and contains a loop while the dataset LOUNGE consists of 3000 RGB-D images and does not contain any loop. We displayed top-views of the obtained 3D models to attest the amount of deformations of the reconstructed scenes. From these results we can see that our method was able to reconstruct the 3D models in details at large scale without deformations, similarly as in [19]. We

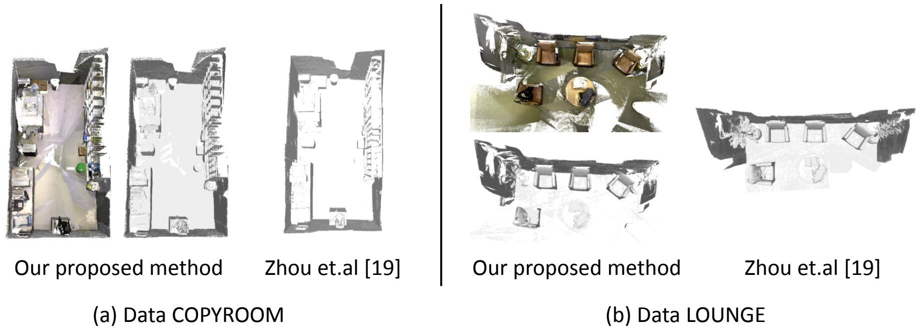

Fig. 5: Top views of two reconstructed scenes. Our method can build geometrical consistent large-scale 3D scenes in real-time. Our proposed method also allowed us to obtain color information of the 3D scene.

remark that our results were produced on-line, while those by [19] were off-line. Moreover, with our method we could generate textured 3D models while texture is not available in the results by [19].

We scanned an indoor scene of 10m by 5m, called Office, with a Kinect camera. The dataset contains 8500 images. This scene is challenging in that deformations become evident at some parts (inside the green circle) in the middle of the scene due to unconnected objects (*e.g.* opposite faces of the central wall), in addition to complex camera motion. As a consequence, deformations of the generated 3D model can be easily observed, and thus the advantage of introducing geometric constraints can be highlighted. Fig. 6 shows the results by our method with/without using geometric constraints ((a) v.s. (b)) and with/without applying fragment registration ((a), (b) v.s. (c)). The improvement of the obtained results using our geometric constraints for the fragment registration is significant, as shown in the circled parts. Fig. 6 (c) shows that, without handling deformations (*i.e.* without fragment registration), results are catastrophic.

5 Conclusion

We proposed a two-stage strategy, local mapping and global mapping, to build in details large-scale 3D models with minimal deformations in real time from RGB-D image sequences. The local mapping creates accurate structured local 3D models from short subsequences while the global mapping organises all the local 3D models into a global model in an undeformed way using fragment registration in the graph optimization framework. Introducing geometric and identity constraints facilitates repositioning planar patches to remove deformations as much as possible. Our method produces 3D models of high quality, without deformations and in real-time, even for large-scale scenes.

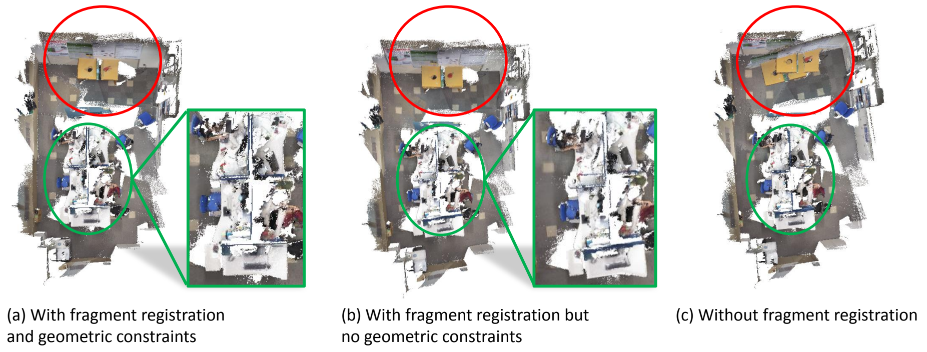
(a) With fragment registration and geometric constraints
(b) With fragment registration but no geometric constraints
(c) Without fragment registration

Fig. 6: Comparative results obtained with data OFFICE. The improvement of the obtained results using geometric constraints for the fragment registration is significant, as shown in the circled parts.

References

1. 3D Scene Dataset: `http://www.stanford.edu/~qianyizh/projects/scenedata.html`.
2. RGB-D SLAM Dataset and Benchmark: `http://vision.in.tum.de/data/datasets/rgbd-dataset`.
3. G. Blais and M.D. Levine. Registering multi vie range data to create 3D computer objects. *IEEE Trans. on PAMI*, Vol. 17, No. 8, pp. 820-824, 1995.
4. R. Cameral, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimisation. *Proc. of ICRA*, 2011.
5. J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics*, 32(4):1132:1–113:8, 2013.
6. P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch volumes: Segmentation-based consistent mapping with RGB-D cameras. *Proc. of 3DV'13*, 2013.
7. P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modelling of indoor environments. *International Journal of Robotics Research*, 31(5):647–663, 2012.
8. D.G. Lowe. Object recognition from local scale-invariant features. *Proc. of ICCV*, 1150-1157, 1999.
9. M. Meilland and A. Comport. On unifying key-frame and voxel-based dense visual SLAM at large scales. *Proc. of IROS*, 2013.
10. M. Neibner, M. Zollhofer, S. Izadi and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):169:1–169:11, 2013.
11. R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *Proc. of ISMAR'11*, 127–136, 2011.
12. C. Nguyen, S. Izadi, and D. Lovell. Modeling kinect sensor noise for improved 3D reconstruction and tracking. *Proc. of 3DIM/PVT'12*, 524–530, 2012.
13. H. Roth and M. Vona. Moving volume kinectfusion. *Proc. of BMVC*, 2012.
14. A. Segal, D. Haehnel, S. Thrun. Generalized-ICP. *Robotics: Science and Systems*, 2009.

15. D. Thomas and A. Sugimoto. A flexible scene representation for 3D reconstruction using an RGB-D camera. *Proc. of ICCV*, 2013.
16. T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johansson, and J. Leonard. Kintinuous: Spatially extended kinectfusion. *Proc. of RSS Workshop on RGB-D: Advanced Reasoning with Depth Camera*, 2012.
17. M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based fusion for realtime 3D reconstruction. *Transaction of Graphical Models*, 75(3):126–136, 2013.
18. Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transaction on Graphics*, 32(4):112:1–112:8, 2013.
19. Q.-Y. Zhou, S. Miller and V. Koltun. Elastic fragments for dense scene reconstruction. *Proc. of ICCV*, 2013.