Computer Vision and Image Understanding 157 (2017) 103–116

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Modeling large-scale indoor scenes with rigid fragments using RGB-D cameras

Diego Thomas[a,b,*], Akihiro Sugimoto[c]

[a] National Institute of Informatics, Tokyo 101-8430, Japan
[b] Kyushu university, Fukuoka, Japan
[c] National Institute of Informatics, Tokyo 101-8430, Japan

ARTICLE INFO

Article history:
Received 16 October 2015
Revised 10 August 2016
Accepted 29 November 2016
Available online 30 November 2016

Keywords:
3D Modeling
RGB-D Camera
Large-scale
Real-time
Loop closure
Bump image
Semi-global model

ABSTRACT

Hand-held consumer depth cameras have become a commodity tool for constructing 3D models of indoor environments in real time. Recently, many methods to fuse low quality depth images into a single dense and high fidelity 3D model have been proposed. Nonetheless, dealing with large-scale scenes remains a challenging problem. In particular, the accumulation of small errors due to imperfect camera localization becomes crucial (at large scale) and results in dramatic deformations of the built 3D model. These deformations have to be corrected whenever it is possible (when a loop exists for example). To facilitate such correction, we use a structured 3D representation where points are clustered into several planar patches that compose the scene. We then propose a two-stage framework to build in details and in real-time a large-scale 3D model. The first stage (the local mapping) generates local structured 3D models with rigidity constraints from short subsequences of RGB-D images. The second stage (the global mapping) aggregates all local 3D models into a single global model in a geometrically consistent manner. Minimizing deformations of the global model reduces to re-positioning the planar patches of the local models thanks to our structured 3D representation. This allows efficient, yet accurate computations. Our experiments using real data confirm the effectiveness of our proposed method.

1. Introduction

3D reconstruction of indoor scenes using consumer depth cameras has attracted an ever-growing interest in the last decade. Many applications such as robot navigation, virtual and augmented reality or digitization of cultural heritage can benefit from highly detailed large-scale 3D models of indoor scenes built using cheap RGB-D sensors. In general, the 3D reconstruction process consists of (1) generating multiple 2.5D views of the target scene, (2) registering (i.e., aligning) all different views into a common coordinate system, (3) fusing all measurements into a single mathematical 3D representation and (4) correcting possible deformations.

To obtain depth measurements of a target scene (i.e., 2.5D views), many strategies exist, which can be classified into either passive sensing or active sensing. A popular example for passive sensing is stereo vision (Lazaros et al., 2008). On the other hand, structured light (Rusinkiewicz et al., 2002) and time-of-flight (Hansard et al., 2012) are the most popular techniques for active sensing. Consumer depth cameras such as the Microsoft Kinect camera or the Asus Xtion pro camera use such techniques and are recent active sensors which produce low quality depth images at video rate and at low cost. These sensors have raised much interest in the computer vision community, and in particular for the task of automatic 3D modeling.

The video frame rate provided by consumer depth cameras brings several advantages for 3D modeling. One distinguished advantage is that it simplifies the registration problem. This is because the transformation between two successive frames can be assumed to be sufficiently small. As a consequence, well-known standard registration algorithms such as variants of the Iterative Closest Point (ICP) (Besl and McKay, 1992) work efficiently. Moreover, by accumulating many (noisy) depth measurements available for each point in the scene, it is possible to compensate for the low quality of a single depth image and construct detailed 3D models. A well known successful work for 3D modeling using RGB-D cameras is KinectFusion (Newcombe et al., 2011). In this system, a linearized version of Generalized ICP (GICP) (Segal et al., 2009) is used in the frame-to-global-model registration framework to align successive depth images, which are accumulated into a volumet-

[*] Corresponding author at: Kyushu University, Motooka, Fukuoka 819-0395, Japan.
E-mail addresses: diegot.thomas@gmail.com (D. Thomas), sugimoto@nii.ac.jp (A. Sugimoto).

http://dx.doi.org/10.1016/j.cviu.2016.11.008
1077-3142/© 2016 Elsevier Inc. All rights reserved.

ric Truncated Signed Distance Function (TSDF) (Curless and Levoy, 1996) using a running average. With using rather simple, well-established tools, impressive 3D models at video frame rate can be obtained, which demonstrates the potential of RGB-D cameras for fine 3D modeling.

Another new interesting property of consumer depth cameras is that they can be held by hands and thus they allow reconstructing large-scale scenes rather easily. With this new possibility, new challenges also arise: how to deal with error propagation and a large amount of data? In other words, how to minimize deformations of the produced 3D model while keeping fine details, even at large-scale?

In the last several years, a lot of research have been reported to allow large-scale 3D reconstruction (Chen et al., 2013; Henry et al., 2013; Meilland and Comport, 2013; Neibner et al., 2013; Roth and Vona, 2012; Thomas and Sugimoto, 2014; Whelan et al., 2012; Zeng et al., 2013; Zhou and Koltun, 2013; Zhou et al., 2013). Noticeable works employ hash tables for efficient storage of volumetric data (Neibner et al., 2013), patch volumes to close loops on the fly (Henry et al., 2013) and non-rigid registration of sub-models to reduce deformations (Zhou et al., 2013). Though recent works considerably improved the scale and the quality of 3D reconstruction using consumer depth cameras, existing methods still suffer from little flexibility for manipulating the constructed 3D models. This is because most of existing methods work at the pixel level with unstructured 3D representations (voxels with volumetric TSDF or 3D vertices with meshes). Modifying the whole 3D model then becomes difficult and computationally expensive, which precludes from closing loops or correcting deformations on-the-fly. Instead of unstructured representations, introducing the structured 3D representation overcomes this problem. In the work by Henry et al. (2013), indeed, patch volumes are used to manipulate structured 3D models, which allows simpler modifications of the 3D model. In this work, we push forward in this direction by taking advantage of a parametric 3D surface representation with bump image (Thomas and Sugimoto, 2013) that allows easy manipulations of the 3D model while maintaining fine details, real-time performance and efficient storage. Our experimental evaluation demonstrates that by using the structured 3D representation, loops can be efficiently closed on-the-fly and overall deformations are kept to a minimum, even for large-scale scenes.

The main contributions of this paper are: (1) the creation of a *semi-global model* to locally fuse depth measurements and (2) the introduction to *identity constraints* between multiple instances of a same part of the scene viewed at different time in the RGB-D image sequence, which enables us to run a fragment registration algorithm to efficiently close loops. Overall, we propose a method that segments the input sequence of RGB-D images both in time and space and that is able to build in real-time high fidelity 3D models of indoor scenes. After briefly recalling the parametric 3D surface representation with bump image in Section 3 we introduce our two-stage strategy in Section 4. We demonstrate the effectiveness of our proposed method through comparative evaluation in Section 5 before concluding in Section 6. We note that a part of this work appeared in Thomas and Sugimoto (2014).

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.

Weise et al. (2009) proposed to build a global 3D model using surface elements called Surfels (Pfister et al., 2000). They also proposed to handle loop closure by enforcing the global model as-rigidly-as-possible using a topology graph where each vertex is a Surfel and edges connect neighboring Surfels. However, maintaining such a topology graph for large-scale scenes is not practical because of the existence of potentially millions of Surfels. Keller et al. (2013) proposed a real-time point-based fusion algorithm for memory efficient and robust 3D reconstruction. However, how to correct drift errors that arise at large scale is not discussed, which is a crucial limitation when reconstructing large scale scenes.

Newcombe et al. (2011) proposed KinectFusion: a system to build implicit 3D representations of a scene from an RGB-D camera at an interactive frame-rate (higher than 30fps). The implicit representation consists of a TSDF that is discretized into a volume covering the scene to be reconstructed. From the volumetric TSDF and a given camera pose, dense depth images can be generated, which allows accurate and robust camera pose tracking. They also proposed a GPU framework to integrate live depth data into the volumetric TSDF at interactive frame-rate. The TSDF is recorded into a regular voxel grid, which requires a large amount of memory usage. This limits the practicability of the method for large-scale scenes.

Thereafter, much work has been done on extending KinectFusion for large-scale applications (Chen et al., 2013; Neibner et al., 2013; Roth and Vona, 2012; Whelan et al., 2012; Zeng et al., 2013). Roth and Vona (2012) and Whelan et al. (2012) proposed Kinect moving volumes and Kintinuous (respectively). In these methods, the volume is automatically translated and rotated in space as the camera moves, and remapped into a new one by the TSDF interpolation whenever sufficient camera movement is observed. The use of the non-regular volumetric grid has also been studied for compact TSDF representations. Zeng et al. (2013) and Chen et al. (2013) proposed an octree-based fusion method. Neibner et al. (2013) 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. Correcting deformations in the volumetric TSDF is tedious and, as a consequence, multiple passes over the whole sequence of RGB-D images are required. This also means that the whole sequence of RGB-D images needs to be recorded at run-time. This becomes a critical limitation for large-scale applications.

Meilland and Comport (2013) 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 visualization. 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 and Koltun (2013) proposed to use local volumes around points of interest, and proposed an offline reconstruction method with two passes. The first pass estimates a rough camera trajectory and segment the camera path into interest and connector segments. The second pass refines the camera path by employing KinectFusion in each interest segment. A pose-graph is built during the second pass that is optimized in a post-process for the final accurate estimation of the camera trajectory. Thereafter all RGB-D images are fused into a large volumetric TSDF (computation are done on CPU so that memory is not a problem anymore) to generate the final 3D model. The proposed technique allows for accurate 3D modeling of large-scale scenes but at the cost of huge computational time. Moreover, as pointed out in Zhou et al. (2013), even the ground truth camera trajectory is not sufficient to generate the ideal 3D model. This is because of the low frequency noise that

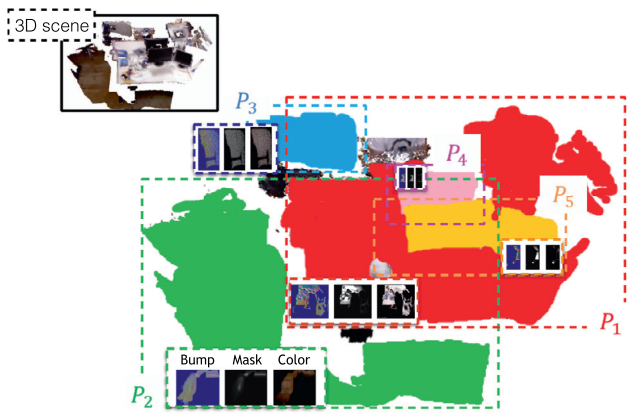


Fig. 1. The 3D representation for an indoor scene. The target scene is segmented into multiple coplanar parts. Each segment S is modeled by the equation of the plane P that best fit the set of 3D points in S, the 2D bounding box of the projection of all 3D points in S into the plane P, and three 2D images that encode local geometry (Bump), color (Color) and confidence of measurements (Mask). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

exists in each depth image (such noise may come from lens distortion or inexact camera models for example).

To account for the low frequency noise in each depth image, Zhou et al. (2013) proposed to employ elastic registration between 3D meshes built from short subsequences of the whole RGB-D image sequence. The KinectFusion algorithm is applied to each short-time subsequence and then local meshes of the scene are generated. After all subsequences are processed, an elastic registration algorithm runs that combines rigid alignment and non-rigid optimization for accurate alignment, and correction of drift-errors and distortions caused by the low frequency noise. This method achieves the 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.

Henry et al. (2013) proposed to segment the scene into planar patches and to use 3D TSDF volumes around them to represent the 3D scene. In Henry et al. (2013), a pose-graph optimization for loop closure was proposed where each vertex of the graph is a planar patch and it is connected to one or multiple keyframe(s). When a loop is detected, it gives constraints on the relationship between patches and keyframes. Optimizing 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. In Thomas and Sugimoto (2013), the authors proposed a method that requires only three 2D images, called attributes (i.e., Bump image, Mask image and Color image), for each planar patch to model the scene, which allows a more compact representation of the scene. Though they achieved significant compression of the 3D scene representation, no discussion was given about how to deal with deformations that arise when applied to large-scale scenes.

Differently from the above methods, this paper uses the graph optimization framework with new rigidity constraints built between planar patches representing different parts of the scene and identity constraints built between planar patches representing the same part of the scene to accurately and efficiently correct deformations of the global model. (Figs. 1–14).

3. Parametric 3D scene representation

We construct 3D models of indoor scenes using a structured 3D scene representation based on planar patches augmented by Bump images (Thomas and Sugimoto, 2013). At run-time, the scene is segmented into multiple planar patches and the detailed geometry of the scene around each planar patch is recorded into the Bump images. Each Bump image (i.e., the local geometry) is assumed to be accurate because standard RGB-D SLAM algorithms work well locally (deformation problems arise at large scale). Therefore, correcting deformations at large scale in our built 3D model can be reduced to repositioning each planar patch consistently (the local geometry encoded in the Bump image do not need to be modified). This allows efficient computations. We choose to use the parametric representation with Bump image (Thomas and Sugimoto, 2013) rather than patch volumes (Henry et al., 2013) because (1) it is a more compact representation and (2) it is easier to compute dense point correspondences between multiple planar patches, which is required to run accurate registration (see Section 4.2.2). Below, we briefly review our used representation (Thomas and Sugimoto, 2013) to make the paper self-contained.

To each surface patch detected in the scene, three 2D images are attached as its attributes: a three-channel Bump image, an one-channel Mask image and a three-channel Color image. These three images encode geometric and color details of the scene. The Bump image encodes the local geometry around the planar patch. For each pixel, we record in the three channels the displacement of the 3D point position from the lower left-corner of its corresponding pixel. The Mask image encodes the confidence of accumulated data at a point and the Color image encodes the color of each point. Fig. 1 illustrates the 3D representation for an indoor scene.

3.1. Planar patches with attributes

Let us assume we are given a set of points $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ with $n > 2$ and there are at least 3 points that do not lie on the same line, and a planar patch Π of a finite size, with the equation determined by $(\mathbf{n}, d)$[1] and 2D bounding box $Bbox = (ll, lr, ul, ur)$ that best fits P. Any 3D point on Π (with normal $\mathbf{n}$) can be represented using parameters (t, h) with $0 \le t, h \le 1$ that run over Π.

$$\begin{aligned} \Pi : ([0,1])^2 &\to \mathbb{R}^3 \\ (t,h) &\longmapsto (x(t,h), y(t,h), z(t,h))^\top. \end{aligned}$$

Computation of the projected points. We define a projection operator π_Π for Π, such that for any $\mathbf{p} \in \mathbb{R}^3$,

$$\pi_\Pi(\mathbf{p}) = \underset{(t,h)\in([0,1])^2}{\text{argmin}} \; (\|\mathbf{p} - \Pi(t,h)\|_2).$$

$\pi_\Pi(\mathbf{p})$ represents the 2D parameters of the projection of $\mathbf{p}$ onto Π.

Discretisation of the planar patch. The planar patch Π is discretized in both dimensions into given m and l uniform segments. We define the operator *discr* that maps real coordinates $(t, h) \in ([0, 1])^2$ into discrete coordinates[2] $discr(t, h) \in [\![0, m]\!] \times [\![0, l]\!]$.

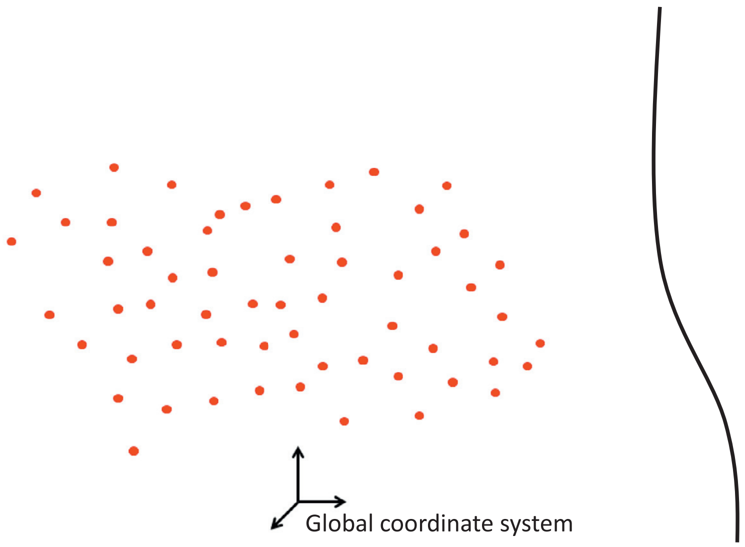
$$\begin{aligned} &\forall (t,h) \in ([0,1])^2, \\ &discr(t,h) = (\lfloor t \times m \rfloor, \lfloor h \times l \rfloor). \end{aligned}$$

3.2. Bump image

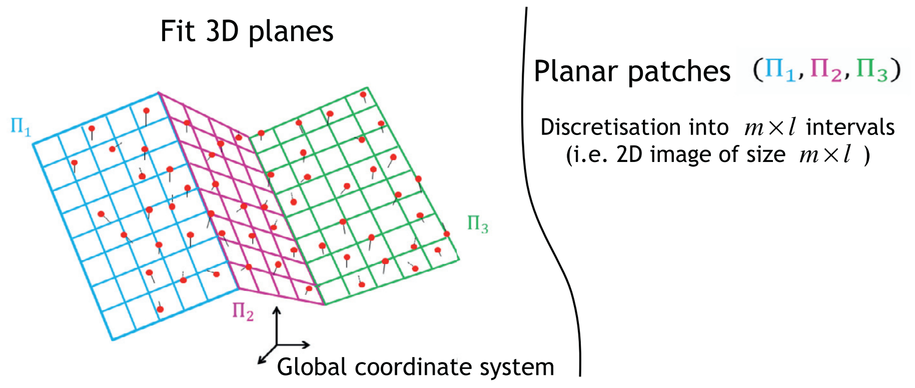
The Bump image, encoded as a three-channel 16 bits image, represents the local geometry of the scene around a planar patch.

[1] $\mathbf{n}$ is the normal of the plane Π and d is the distance of the plane Π from the origin.

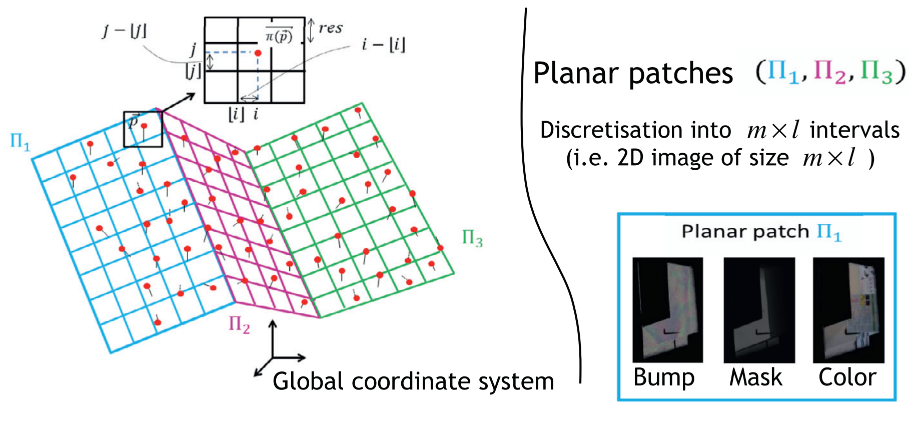
[2] The notation $[\![a, b]\!]$ denotes the integer interval between a and b.



(a) Input cloud of points.


Fit 3D planes
Π_1
Π_2
Π_3
Global coordinate system
Planar patches (Π_1, Π_2, Π_3)
Discretisation into $m \times l$ intervals
(i.e. 2D image of size $m \times l$)

(b) Planar patch segmentation.



(c) Compact and accurate 3D representation.

Fig. 2. Simple example of the 3D scene representation. The input cloud of points (a) is segmented into multiple coplanar segments to each of which is attached a planar patches (Π_1, Π_2, Π_3) that best fit the segment (b). Each planar patch is discretized in both dimensions to generate the 2D image attributes. For each point of each planar segment, its 3D position, color and measurement confidence are recorded into the Bump, Color and Mask images of the corresponding planar patch (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

For a given planar patch, the Bump image is obtained by identifying points around the planar patch and projecting them onto the discretized planar patch. The three values of a pixel in the Bump image encode the exact 3D position of the point corresponding to the pixel in the global coordinate system.

For each point $\mathbf{p} \in P$, we obtain the pixel coordinates (i, j) of $\mathbf{p}$ in *Bump* as $(i, j) = discr(\pi_\Pi(\mathbf{p}))$. We encode in the Bump image the deviation caused by the discretisation process together with the signed distance of $\mathbf{p}$ from the planar patch Π (with respect to the normal vector $\mathbf{n}$) so that we can exactly recover the 3D position of $\mathbf{p}$. Namely, $Bump(i, j) = [\pi_\Pi(\mathbf{p})[0] \times m - i, \pi_\Pi(\mathbf{p})[1] \times l - j, (\mathbf{p} - \gamma(\pi_\Pi(\mathbf{p}))) \cdot \mathbf{n}]$, where $\cdot$ is the scalar product operator. Note that in the case where two points project onto the same pixel in *Bump*, we keep in *Bump* only the values for the point that is closest to Π.

On the other hand, computing the 3D positions of points is straightforward. Given a planar patch Π and its Bump image *Bump*, we can compute the exact position of a point corresponding to a pixel (i, j) in the global 3D coordinate system.

$$\mathbf{p}(i,j) = \mathbf{\Pi}\left(\frac{i + Bump(i,j)[0]}{m}, \frac{j + Bump(i,j)[1]}{l}\right) + Bump(i,j)[2] \times \mathbf{n}. \quad (1)$$

Fig. 2 illustrates the process of representing a cloud of points using planar patches. In this simple example, three planar patches are used to represent the input cloud of points. The process discussed above is applied independently to each planar segment. Note that 3D points in the scene that do not belong to any planar patch are not recorded in any Bump image, and are thus lost. However, we reason that indoor man-made scenes are mostly composed of roughly planar parts. Actually, as discussed in Thomas and Sugimoto (2013), in general the planar patch representation can encode about 90% of the number of points in the target scene.

3.3. Mask and color images

The Mask image encodes the confidence for 3D points. This allows us to perform a running average when fusing live measurements and also to eliminate erroneous measurements. The values of the Mask image are initialized when creating the Bump image. A pixel in the Mask image is set to 1 if a 3D point is projected onto the same pixel location in the Bump image, it is set to 0 otherwise. The Mask image is then updated with live measurements as explained in Section 4.1.1 and Section 4.1.2. The Color image takes the RGB values of the points that are projected into the corresponding pixel in the Bump image. We can thus recover color information of 3D points as well.

4. A two-stage strategy for 3D modeling

We employ the above mentioned 3D scene representation for constructing large-scale 3D models. With this representation, the target scene is structured into multiple planar patches with attributes. To achieve successful 3D reconstruction using such a representation one key assumption is that each planar patch is correctly reconstructed (i.e., no deformations). If this assumption is satisfied then drift errors in the camera pose estimation can be corrected by just repositioning different planar patches that compose the scene, which eases computations and enables real-time performance. Even though deformations inevitably appear after some time, we reason that (as in Zhou et al., 2013) deformations remain small in 3D models built from short sequences of RGB-D images. We thus 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. 3), the local mapping and the global mapping, to build a large-scale 3D model.

In the local mapping, on one hand, we build a local 3D model (with the 3D representation presented in Section 3) from each short subsequence of RGB-D images. We attach to each local 3D model a keyframe (i.e., an RGB-D image) and constraints that will be useful for the global mapping. Namely, we generate rigidity constraints that represent relative positions between different planar patches in the local 3D model, and visibility constraints that represent the position of all planar patches with respect to the keyframe.

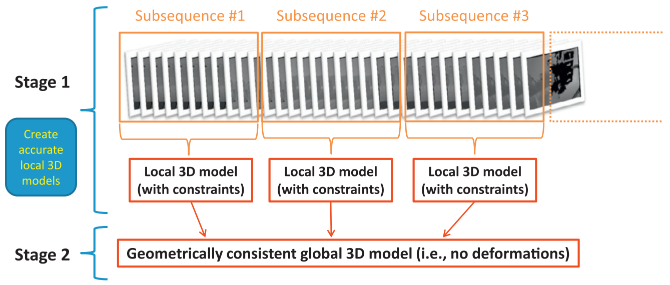

Subsequence #1
Subsequence #2
Subsequence #3
Stage 1
Create accurate local 3D models
Local 3D model (with constraints)
Local 3D model (with constraints)
Local 3D model (with constraints)
Stage 2
Geometrically consistent global 3D model (i.e., no deformations)

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

Locally, the frame-to-global-model framework has proven to be successful (Newcombe et al., 2011) to build accurate 3D models from short sequences of RGB-D images. However, as shown in Thomas and Sugimoto (2013) and Henry et al. (2013), 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 fusion. The attributes of all planar patches (i.e., Bump image, Color image, and Mask image) are then built on-line from the semi-global model. As we will show later, this allows us to keep real-time performance with the state-of-the-art accuracy. Rigidity constraints are generated from the first frame of each short subsequence, and they will be used to re-position all planar patches in the scene when loops exist. This is because we reason that the exact relative positions between different parts in the scene can be reliably estimated only from a single image (deformations usually arise after merging multiple RGB-D images).

In the global mapping, on the other hand, we build a graph where each vertex represents either a planar patch generated from the local mapping or a keyframe (the RGB-D image corresponding to the state of the semi-global model at the first frame of each subsequence), and edges represent our introduced rigidity constraints, identity constraints of patches over keyframes, keyframe pose constrains (Henry et al., 2013), or visibility constraints (Henry et al., 2013). Over the graph, we keep all similar planar patches (i.e., multiple instances of the same part of the scene) without merging, which allows more flexibility in re-positioning all the planar patches. Moreover, the rigidity 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.

Planar patches segmentation. We detect different planes in a depth image by using the gradient image of the normal image. From the gradient image, we identify contours and extract connected components in the depth image. For each connected component that represents one planar patch, we first compute the median normal vector and then the median distance to the origin to have the plane equation for the planar patch. Once the plane equation is computed, we identify the bounding box of all planar patches and initialize their attributes. Note that for each subsequence, we detect a set of planar patches only from the depth image corresponding to the keyframe and we fix the set over the subsequence.

4.1. Local mapping

The goal of the local mapping is to build accurate 3D models of planar patches in the scene that is scanned during a short sequence of RGB-D images. Each 3D model consists of a plane equation, a bounding box and three 2D image attributes. The set of all

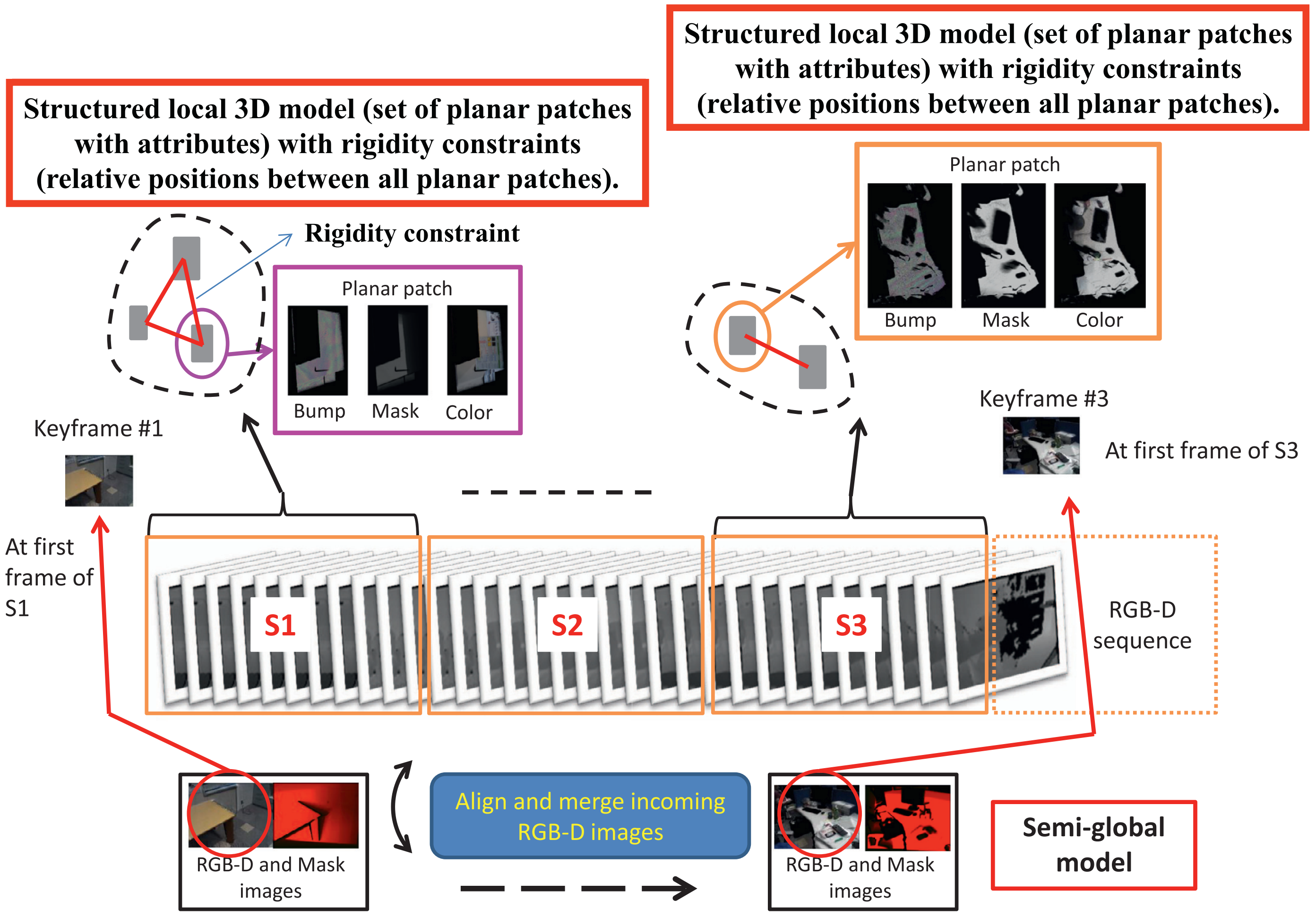

Structured local 3D model (set of planar patches with attributes) with rigidity constraints (relative positions between all planar patches).
Rigidity constraint
Planar patch
Bump
Mask
Color
Structured local 3D model (set of planar patches with attributes) with rigidity constraints (relative positions between all planar patches).
Planar patch
Bump
Mask
Color
Keyframe #1
Keyframe #3
At first frame of S3
At first frame of S1
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. 4. 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.

planar patches' 3D models is called a local 3D model. We also output geometric relationships (i.e., relative positions) between all different planar patches of the local 3D model. These relationships will be used as rigidity constraints in the global mapping to consistently re-position all patches of the local 3D models in the large-scale scene. To build local 3D models and generate their associated rigidity constraints, two challenges exist: (1) how to build the local 3D model in real-time and (2) how to generate "good" rigidity constraints (i.e., constraints that represent the exact relative positions between all planar patches). We tackle these challenges with keeping in mind the accuracy, real-time processing and minimal deformations. Fig. 4 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 meaningless). Therefore, we need to employ another 3D model to perform accurate, robust and fast camera tracking and data fusion. 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. Accordingly, we introduce a new model, called semi-global model, to track camera and integrate incoming RGB-D data. Note that fused RGB-D data are lively recorded into the planar patches 3D representation.

To tackle the second challenge, a standard approach is to incrementally build the local 3D model by adding new planar patches on the fly as they are being detected (as done in Thomas and Sugimoto, 2013 or Henry et al., 2013) and to generate rigidity constraints at the end of each subsequence by computing their relative positions. The relative positions (i.e., rigidity constraints) between planar patches detected at different times can be significantly different from the exact relative positions due to drift errors. Therefore we compute the rigidity constraints only from a single RGB-D image (the keyframe of the 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 sufficiently close to the exact ones.

4.1.1. Semi-global model.

Our semi-global model is defined as a pair of RGB-D and Mask images. The semi-global model is initialized with the first frame of the whole RGB-D image sequence and is updated with each aligned input RGB-D image. For efficient and robust camera tracking, the frame-to-global-model registration strategy is the best solution. That is why our semi-global model is updated at every frame and used as a reference RGB-D image for camera pose estimation. From the first camera viewpoint, however, the semi-global model has a limited field of view and many parts of the scene

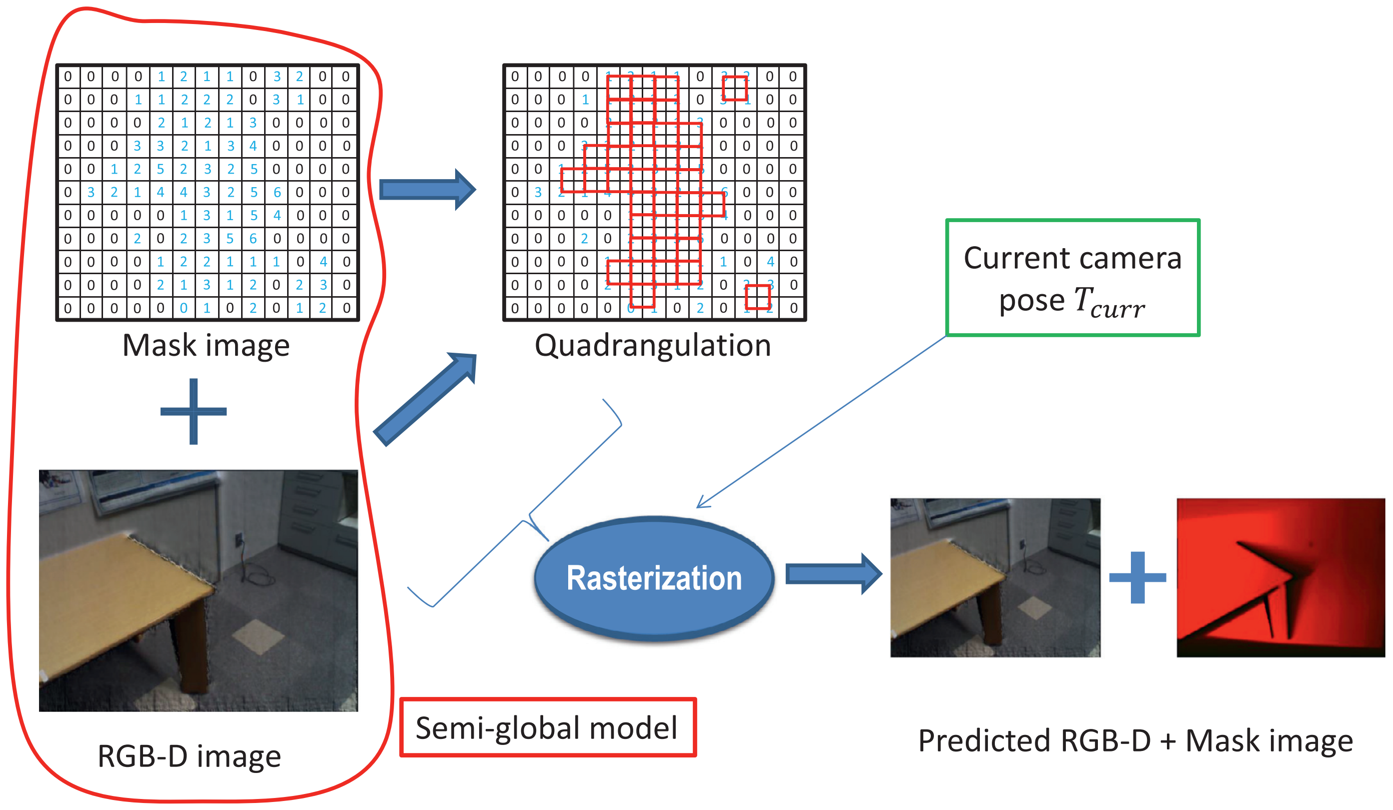


Fig. 5. Generating a pair of predicted RGB-D and Mask images. We use a rasterizing rendering pipeline to render RGB-D and mask images corresponding to the estimated current camera pose.

are occluded. Therefore, in addition to fusing new data, the semi-global model is also re-rendered with each camera pose to maximize the overlap with respect to the current RGB-D image. By doing so, the semi-global model represents the field of view of the current camera viewpoint (note that points dropped from the semi-global model during this process are recorded into the local model). To render predicted RGB-D and Mask images, we use a rasterizing rendering pipeline with the natural quadrangulation given by the organization of points in the 2D image.

For each pixel $(u, v) \in [\![1, n-1]\!] \times [\![1, m-1]\!]$, a quad is created if the Mask value 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 5 cm 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. 5 illustrates how to obtain the predicted RGB-D and Mask images. The rendered RGB-D image is used to align the incoming RGB-D image using the technique described in Henry et al. (2013), which combines both geometric and color information. The predicted Mask image is used to merge aligned data with a running average, as proposed in Thomas and Sugimoto (2013). The semi-global model is then renewed to a 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.

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).

4.1.2. Local 3D model.

For each subsequence, we segment the keyframe (i.e., the first predicted RGB-D image) into several planar patches that form (together with their attributes) our local 3D model. As proposed in Thomas and Sugimoto (2013), 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 larger than that at the projected pixel (as explained in Thomas and Sugimoto (2013)).

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 rigidity constraints on the planar patches.

4.2. Global mapping

The objective of the second stage is to fuse all local 3D models generated in the first stage into a single global 3D model with minimal deformations. The main problem here comes from the accumulation of registration errors, which causes dramatic deformations in the global model (in particular, for large-scale scenes). If the global model becomes deformed, the rigid alignment of new local 3D models to the (deformed) global model becomes ineffective. In such a case, merging wrongly aligned planar patches (that represent the same entity in the scene) will irremediably corrupt the global model. Then, loop closure correction by re-positioning planar patches (Henry et al., 2013) becomes meaningless, because

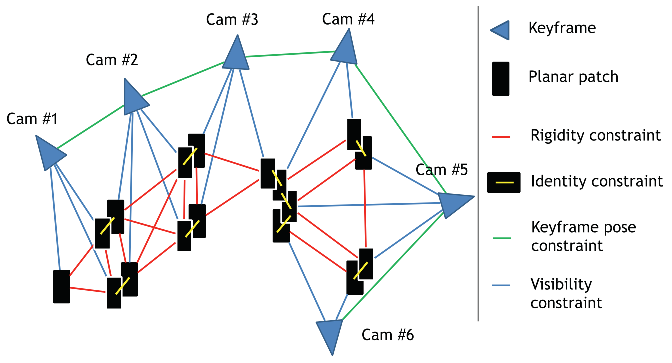


Fig. 6. The global graph with constraints. Vertices represent either planar patches or keyframes. Edges represent either rigidity constraints, identity constraint, keyframe pose constraints or visibility constraints. Note that similar patches are not merged together but rather connected together with identity edges (that represent identity constraints).

planar patches themselves have already become deformed. Applying non-rigid alignment of local 3D models to the global model (Zhou et al., 2013) is not good either, because of the 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, instead of merging planar patches referring to the same entity in the scene. This allows more flexibility in re-organizing the global model (a mistake in alignment may be corrected later when additional constraints become available). An identity constraint represents the relative position between planar patches representing the same part in the scene that come from different local 3D models. A graph optimization framework (namely, the g2o framework Cameral et al., 2011) is used to guarantee geometric consistency of the global model (thus reducing deformations). Once the 3D scanning of the target scene is finished, planar patches that are connected with identity constraints are merged to output the final 3D model. The shape and boundary of the merged planar patch are computed using redundant information from all the planar patches that refer to the same entity in the scene (i.e., linked by identity constraints). We note that the merging can be executed on request even during 3D scanning because the local mapping and the global mapping run independently.

4.2.1. Global graph.

We organize 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 rigidity 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 and by $Pset_i = \{P^i_j\}_{j\in[1:m_i]}$ the set of m_i planar patches built with Seq_i. For each planar patch P^i_j we build a local Cartesian coordinate system $(\mathbf{\Omega}^i_j, \mathbf{e}^{i,j}_1, \mathbf{e}^{i,j}_2, \mathbf{n}^{i,j})$, where $\mathbf{\Omega}^i_j$ is the origin of the coordinate system and $(\mathbf{e}^{i,j}_1, \mathbf{e}^{i,j}_2, \mathbf{n}^{i,j})$ is the orthonormal basis of the coordinate system. The vector $\mathbf{n}^{i,j} = (n_x, n_y, n_z)$ is the normal vector of the plane that corresponds to P^i_j. $\mathbf{e}^{i,j}_1$ and $\mathbf{e}^{i,j}_2$ are two orthonormal vectors lying in the plane P^i_j that are computed as follows:

$$\begin{aligned}
\mathbf{e}^{i,j}_1 &= (-n_y, n_x, 0) \text{ if } n_x \neq 0 \text{ and } n_y \neq 0,\\
\text{else } \mathbf{e}^{i,j}_1 &= (-n_z, 0, n_x) \text{ if } n_x \neq 0 \text{ and } n_z \neq 0,\\
\text{else } \mathbf{e}^{i,j}_1 &= (0, -n_z, n_y) \text{ if } n_y \neq 0 \text{ and } n_z \neq 0,\\
\text{else } \mathbf{e}^{i,j}_1 &= (1, 0, 0) \text{ if } n_y \neq 0 \text{ or } n_z \neq 0,\\
\text{else } \mathbf{e}^{i,j}_1 &= (0, 1, 0).\\
\mathbf{e}^{i,j}_2 &= \mathbf{n}^{i,j} \wedge \mathbf{e}^{i,j}_1,
\end{aligned}$$

where $\wedge$ denotes the cross product operator. The origin $\mathbf{\Omega}^i_j$ is defined as the 3D position (in the global coordinate system) of the lower left corner of the bounding box for P^i_j (with the oriented

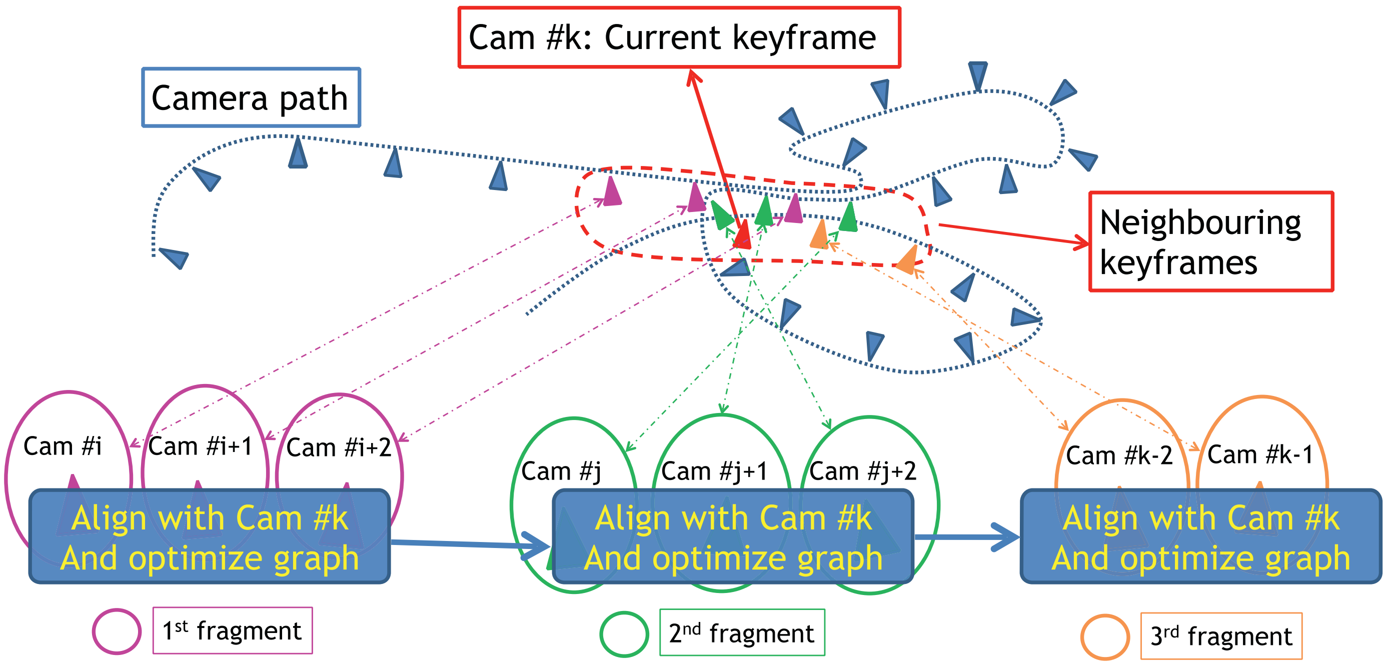


Fig. 7. Fragment registration. The current local 3D model (that corresponds to the current keyframe) is successively aligned to multiple rigid fragments of the global 3D model. One rigid fragment is identified as a set of neighboring local 3D models that correspond to keyframes with successive timestamps. After each rigid alignment, additional constraints are added to the global graph and the graph is optimized. This allows integrating new local 3D models into the global 3D model using rigid registration only, even when the global 3D model is deformed (due to multiple loops for example). Employing only dense rigid registration enables fast processing.

coordinate system). Each planar patch P_j^i is then represented as a vertex in the graph to which the matrix of basis V_j^i is assigned:

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

Matrix V_j^i transforms 3D points from the world coordinate system to the local coordinate system of P_j^i.

Each keyframe K_i is represented as a vertex in the graph to which its pose (4×4 matrix) T_i (computed during the local mapping) is assigned. Note that V_j^i and T_i are the variables in our graph optimization problem. Our objective is to find the optimal V_j^i and T_i with respect to the observations (i.e., the set of constraints).

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. In the remaining of this paper, we will replace T_{edge} by $T_{i,j,k}^{Rig}$, $T_{i,k,j,l}^{Id}$, $T_{i,i+1}^{Key}$ or $T_{i,j}^{Vis}$ when referring to rigidity constraints, identity constraints, keyframe pose constraints or visibility constraints (respectively). Note that once created, the value of each edge remains fixed during the global mapping (while the values of T_a and T_b are optimized).

1. **Rigidity 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 $T_{i,j,k}^{Rig} = V_k^i (V_j^i)^{-1}$ (i.e., the relative position) that defines the rigidity constraint between P_j^i and P_k^i. These edges are created at the end of each local mapping. Note that P_j^i and P_k^i are built from the same subsequence Seq_i.
2. **Identity constraints.** Identity constraints are defined by the relative positions between planar patches representing the same part of 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 $\mathrm{overlap}(P_k^i, P_l^j) > \tau_3$, where $\cdot$ is the scalar product, $\mathrm{overlap}(P_k^i, P_l^j)$ 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 $T_{i,k,j,l}^{Id} = V_l^j (V_k^i)^{-1}$ that defines the identity constraint. Note that $i \neq j$.
3. **Keyframe pose constraints** (Henry et al., 2013). 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** (Henry et al., 2013). 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_j^i), we assign to it matrix $T_{i,j}^{Vis} = V_j^i T_i^{-1}$ that defines the visibility constraint between P_j^i and K_i.

4.2.2. On-line fragment registration with 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 corresponding to each planar patch P_j^i and a vertex corresponding to the keyframe K_i. We then add edges so that all planar patches in

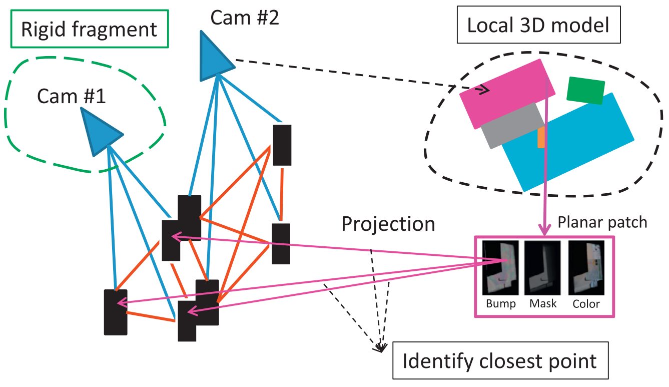


Fig. 8. Efficient registration of the current local 3D model using GICP with projective data association. For each point in the current local 3D model, its corresponding point (i.e., approximated closest point) in the global 3D model is estimated in a few comparisons by projecting it into the different planar patches that compose the reference rigid fragment (which corresponds to the local 3D model of a single keyframe (the previous one) in this simple example). Closest point association is run efficiently on the GPU for fast GICP alignment.

$Pset_i$ are connected with each other, K_i and K_{i-1} (if $i > 1$) are connected, and K_i and any entry in $Pset_i$ are connected (they represent the rigidity constraints, keyframe pose constraint and visibility constraints respectively).

Second, we perform the fragment registration of $Pset_i$ with multiple fragments (i.e., undeformed subsets) of the global graph to include $Pset_i$ into the global model while minimizing 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 (see Fig. 7).

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

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

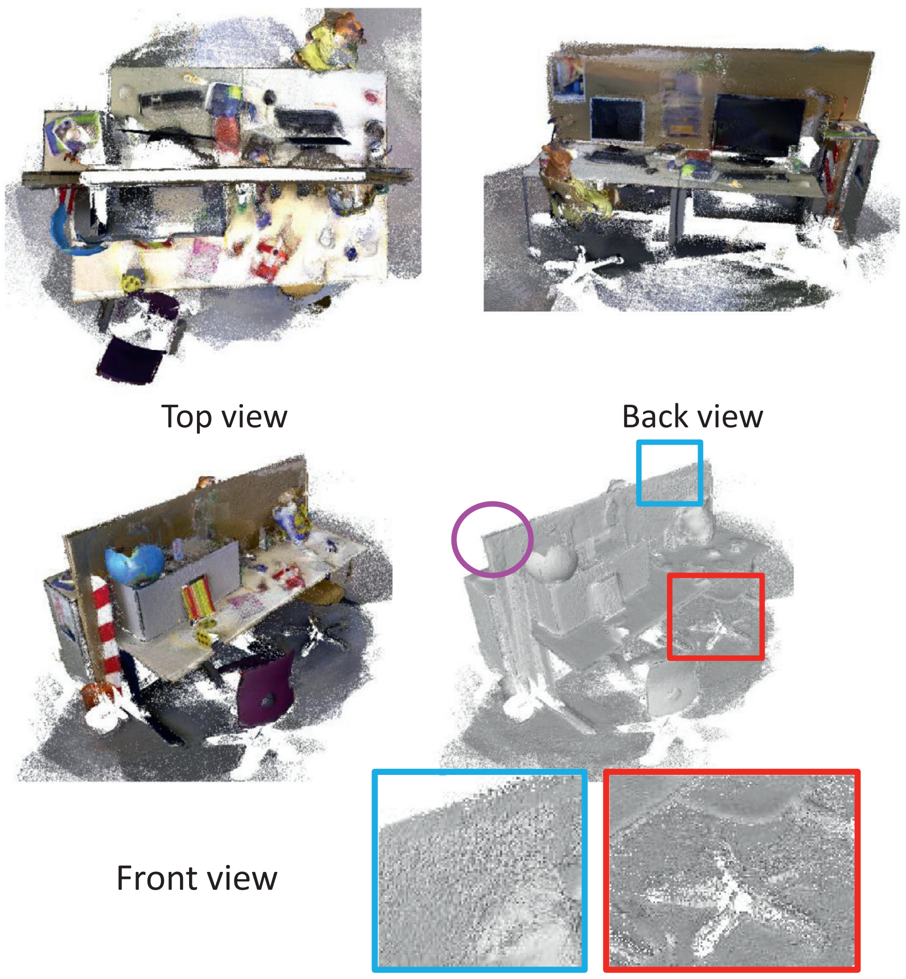
where $d(K_i, K_j)$ and $\alpha(K_i, K_j)$ are the Euclidean distance between the centres of two cameras, and the angle between the two viewing directions of the two cameras (respectively) for the i^{th} and j^{th} keyframes (in the experiments, we set $\tau_d = 3\text{m}$ and $\tau_\alpha = 45°$).

We then break the set S_i into p fragments: $S_i = \{F_i^1, F_i^2, \ldots, F_i^p\} = \{\{K_{s_1}, K_{s_1+1}, K_{s_1+2}, \ldots, K_{s_1+t_1}\}, \{K_{s_2}, K_{s_2+1}, K_{s_2+2}, \ldots, K_{s_2+t_2}\}, \ldots, \{K_{s_p}, K_{s_p+1}, K_{s_p+2}, \ldots, K_{s_p+t_p}\}\}$ 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 that are close to K_i 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 denote by $Pset_i^j$ the set of all planar patches connected to a keyframe in F_i^j ($j \in [1{:}\, p]$).

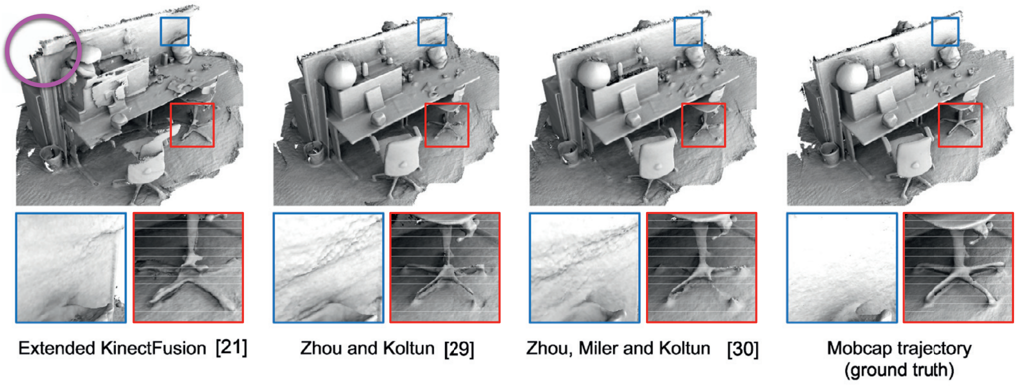
Next, we align $Pset_i$ with each of $\{Pset_i^j\}_{j\in[1:p]}$. The (fragment) registration process of $Pset_i$ with a fragment $Pset_i^j$ ($j \in [1{:}\, p]$) consists of an initialization step using sparse visual features detected in each keyframe followed by the dense GICP. We detail this procedure below.

The transformation that aligns $Pset_i$ with $Pset_i^j$ is initialied by using matches of SIFT features (Lowe, 1999) between K_i and K_{s_j} ($\in F_i^j$). 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 initialized by the matched features; it is set to the identity transformation otherwise.

After the initialization, we apply the GICP algorithm (Segal et al., 2009) to accurately align $Pset_i$ and $Pset_i^j$. Because of millions



(a) Results obtained with our method with real-time performance.


Extended KinectFusion [21]
Zhou and Koltun [29]
Zhou, Miler and Koltun [30]
Mobcap trajectory (ground truth)

(b) Results shown in [30].

Fig. 9. Results obtained with data LONG_OFFICE_HOUSEHOLD. The circled areas show the advantage of using loop-closure algorithms. Without fragment registration, 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

of points in $Pset_i^j$, 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 (Blais and Levine, 1995), which runs 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 $Pset_i^j$. To identify the closest point of $\mathbf{pt}(u, v)$, we search for the point at the projected location that minimizes the Euclidean distance to $\mathbf{pt}(u, v)$ and that has sufficiently small angle between the two normals (we used a threshold of 40º 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. Fig. 8 illustrates the projective data association.

After aligning $Pset_i$ with $Pset_i^j$ as seen above, we generate edges that represent the identity constraints, and then optimize the global graph using the g2o framework (Cameral et al., 2011). We fix the values of vertices corresponding to all planar patches in $Pset_i$ and those corresponding to all planar patches in $Psets_j$. We also fix the values of vertices corresponding to K_{s_j} and K_i. The values of the other vertices are then optimized with respect to all the constraints before proceeding to align $Pset_i$ with the next fragment (if any). After each optimization, all planar patches are re-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 part of the scene are close to the relative positions obtained with fragment registration (this reduces the deformations within each part (the same objects) in the scene).

5. Experiments

We evaluated our proposed method in several situations using real data. All scenes were captured at 30 fps and their dimensions are presented in Table 1. Note that datasets OFFICE, LIBRARY, LIBRARY-2 and LIBRARY-3 were captured by ourselves using a Microsoft Kinect for Windows V1 sensor. We used a resolution of 0.4 cm 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.

Table 1
Dimensions of datasets used to produce our experimental results.

Name of dataset	(length × width × height)	# Frames
LONG_OFFICE_HOUSEHOLD RGBData	4 m × 4 m × 2.5 m	2510
COPYROOM 3D Scene Dataset	8 m × 3 m × 2.5 m	5490
LOUNGE 3D Scene Dataset	8 m × 4.5 m × 2.5 m	3000
OFFICE	10 m × 5 m × 2.5 m	8500
LIBRARY	60 m × 5 m × 2.5 m	19,900
LIBRARY-2	20 m × 6 m × 2.5 m	7800
LIBRARY-3	30 m × 6 m × 2.5 m	9700

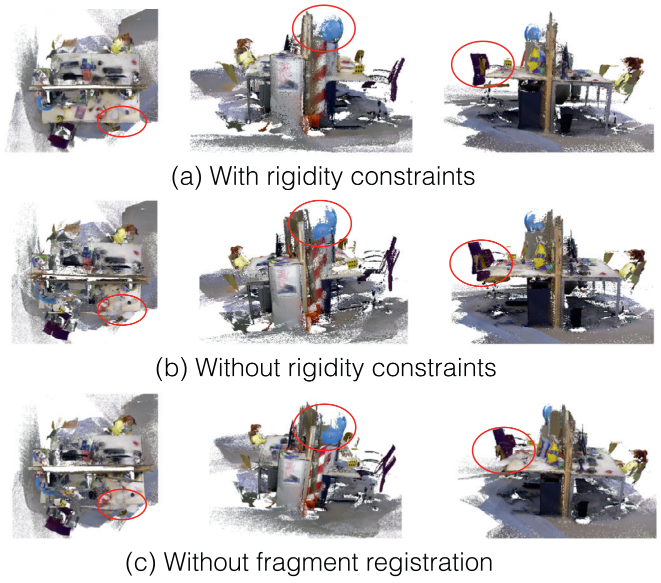

(a) With rigidity constraints
(b) Without rigidity constraints
(c) Without fragment registration

Fig. 10. Results obtained with the dataset LONG_OFFICE_HOUSEHOLD. The improvement of the obtained results using rigidity constraints for the fragment registration is significant, as shown in the circled parts.

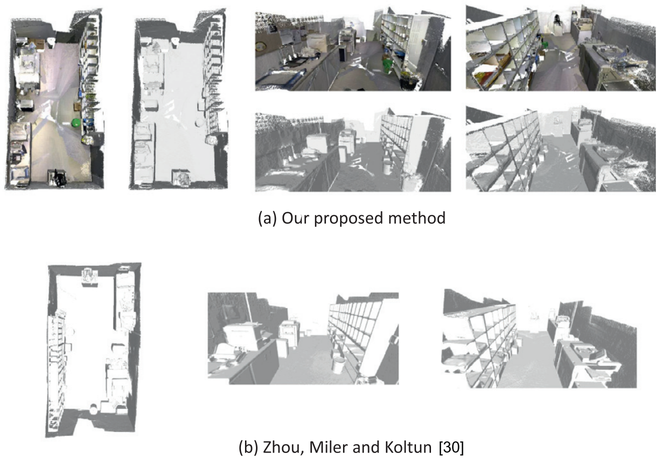


Fig. 11. Top views of two reconstructed scenes for the dataset COPYROOM. Our method can build geometrical consistent large-scale 3D scenes in real-time. Our proposed method also allowed us to obtain color information of the 3D scene.

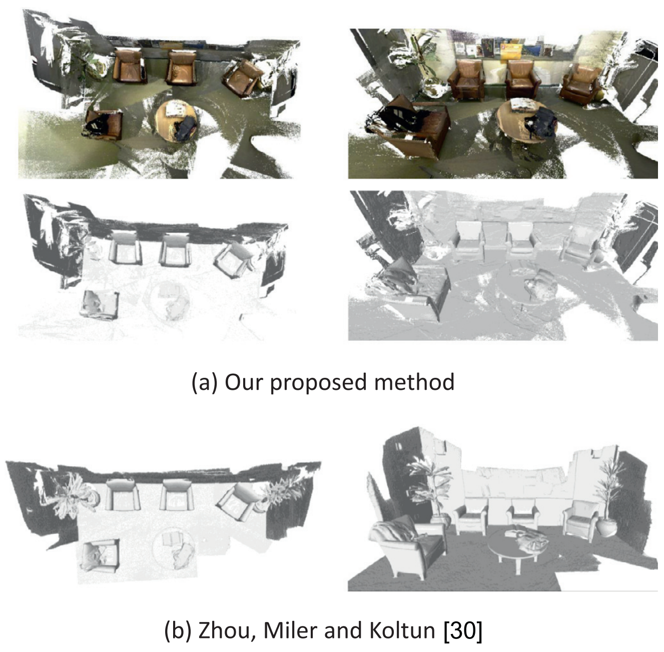


Fig. 12. Top views of two reconstructed scenes for the dataset LOUNGE. Our method can build geometrical consistent large-scale 3D scenes in real-time. Our proposed method also allowed us to obtain color information of the 3D scene.

5.1. Comparative results

Fig. 9(a) shows results obtained by our method using the dataset LONG_OFFICE_HOUSEHOLD RGBData. In this dataset, the camera turns around a small scale scene, which requires loop-closure operations for 3D reconstruction. We compared results by our method with the state-of-the-art results shown in Zhou et al. (2013). Note that we used the mocap camera trajectory in Fig. 9(b) (Henry et al., 2012) as the ground truth. The circled parts on the right side of Fig. 9(a) and on the left side of Fig. 9(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.

Fig. 9(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 accu-

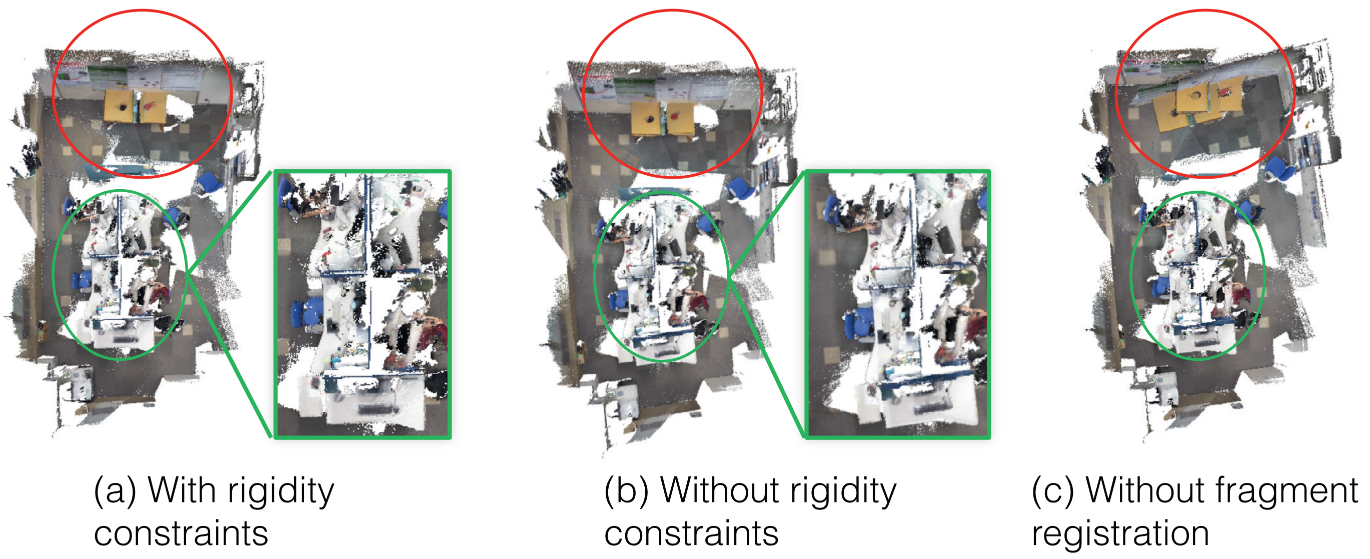

(a) With rigidity constraints
(b) Without rigidity constraints
(c) Without fragment registration

Fig. 13. Comparative results obtained with the dataset Office. The improvement of the obtained results using rigidity constraints for the fragment registration is significant, as shown in the circled parts.

racy the Extended KinectFusion method (Roth and Vona, 2012): to the part where there is only one corner, two corners of the wall are incorrectly reconstructed by Roth and Vona (2012) while ours reconstructed it as one corner. This is because Roth and Vona (2012) 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 Zhou and Koltun (2013) or the ground truth (the mocap trajectory). On the other hand, Zhou et al. (2013) achieved better accuracy than ours: the surface is smoother on the wall (blue box). This is because Zhou et al. (2013) employs non-rigid registration, which is able to correct deformations inside objects that are generated from short subsequences. We remark that our method assumes that each object generated from a short subsequence is correctly constructed (i.e., we cannot correct deformations inside object instances built from short subsequences). Due to employing the planar patches representation, some parts of the scene are missing in the results obtained by our method: parts of the chair in the red box are missing with our method. However, we remark that the results by Zhou and Koltun (2013) or Zhou et al. (2013) are off-line while ours are on-line.

Fig. 10 shows results obtained with our method with (a) and without (b) using our introduced rigidity constraints (Section 4.2.1), and with ((a), (b)) and without (c) using the fragment registration. As expected, the results obtained without using the fragment registration are catastrophic: objects like the chair or the sphere (circled in red) appear twice. Moreover, we can see that to accurately close the loop, rigidity constraints that link different objects in the scene or different instances of the same objects are useful. This is because the rigidity constraints enable us to redistribute errors more coherently with respect to the 3D geometry of the scene, while they are redistributed uniformly along the camera path if not using rigidity constraints.

Fig. 11(a) and Fig. 12(a) show results obtained by our method using the datasets copyroom and lounge respectively. Both datasets copyroom and lounge contain loops. We compared the results obtained by our method with the state-of-the-art results (Zhou et al., 2013) on these two datasets. We displayed top-views of the reconstructed scene to attest the amount of deformations of the reconstructed scenes. From these results we can see that our method is able to reconstruct the 3D scene in details at large scale without deformations, similarly as in Zhou et al. (2013). We remark that our results are produced on-line, while those by Zhou et al. (2013) are off-line. Moreover, with our method we can generate textured 3D models while texture is not available in the results by Zhou et al. (2013).

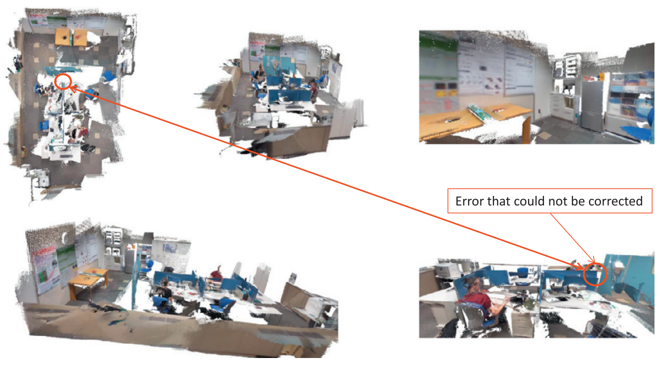


Fig. 14. Limitations of our proposed method observed with the dataset Office. When no constraints are built between different faces of one object, some deformations may not be corrected.

Fig. 13 shows results obtained by our method for the dataset Office. 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 rigidity constraints can be highlighted. Fig. 13 shows the results by our method with/without using rigidity constraints ((a) v.s. (b)) and with/without applying the fragment registration ((a), (b) v.s. (c)). The improvement of the obtained results using our rigidity constraints for the fragment registration is significant, as shown in the circled parts. Fig. 13 (c) shows that, without handling deformations (i.e., without the fragment registration), results are catastrophic.

Unfortunately, as we can see in the circled parts of Fig. 14, we still observe small inconsistencies in the reconstructed 3D scene that could not be corrected. This is because there are no rigidity constraints available between the two opposite faces of the central wall. As this example shows, our method has limitation when dif-

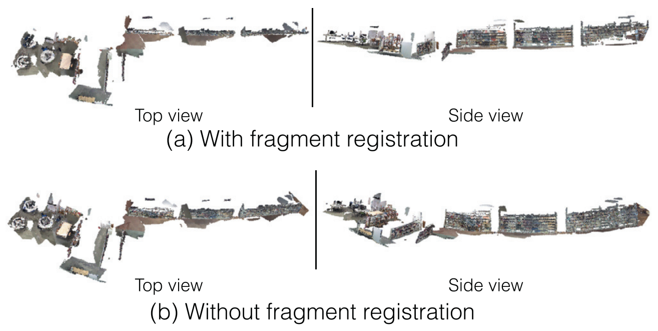

Top view
Side view
(a) With fragment registration
Top view
Side view
(b) Without fragment registration

Fig. 15. Results obtained using the dataset Library with our method with/without the fragment registration.

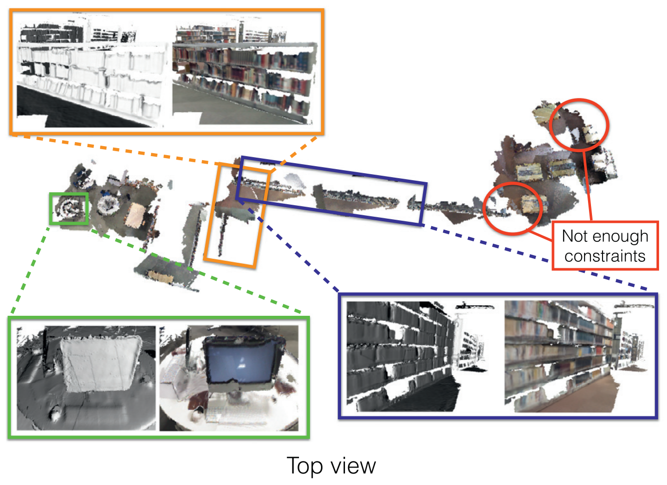


Fig. 16. Detailed results obtained with the dataset Library with our method and with zoomed views. At large scale, our method is able to reconstruct a 3D scene with less deformations, while keeping details of the order of 1 cm: the keyboard on a desk or books in shelves. However, if constraints are insufficient as in the red circles some drift errors cannot be corrected. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ferent parts of the same object are not connected by rigidity constraints.

We also applied our method to our three captured large scale scene datasets: Library, Library-2 and Library-3 (cf. Table 1). The results are shown in Figs. 15–18.

As we can see from Figs. 16 to 18, our proposed method could reconstruct large-scale 3D models in details. In Fig. 15, we compare results obtained with our method with/without employing the fragment registration. As we can see, our proposed method could significantly reduce deformations caused by the drift in camera pose estimation. In Fig. 16 we show zoomed views of the reconstructed scene (in the square boxes) to confirm that our proposed method could retrieve geometric details of the scene. Note that, actually, there is no loop in the data Library because the camera is always moving forward. That is why, as shown in Fig. 16 in the circled area, some deformations could not be corrected. The reduced deformations observed in Fig. 15 come from alignment of redundant information from planar patches that refer to the same parts of the scene detected in successive short-time sub-sequence.

In Figs. 17 and 18 we also compared results obtained by our proposed method with/without using our introduced rigidity constraints into the fragment registration. On these two datasets we obtained similar results with or without using the rigidity constraints. However, we can observe that, as expected, the way of redistributing the errors along the camera path differs when using or not the rigidity constraints. In the circled regions in Fig. 17,

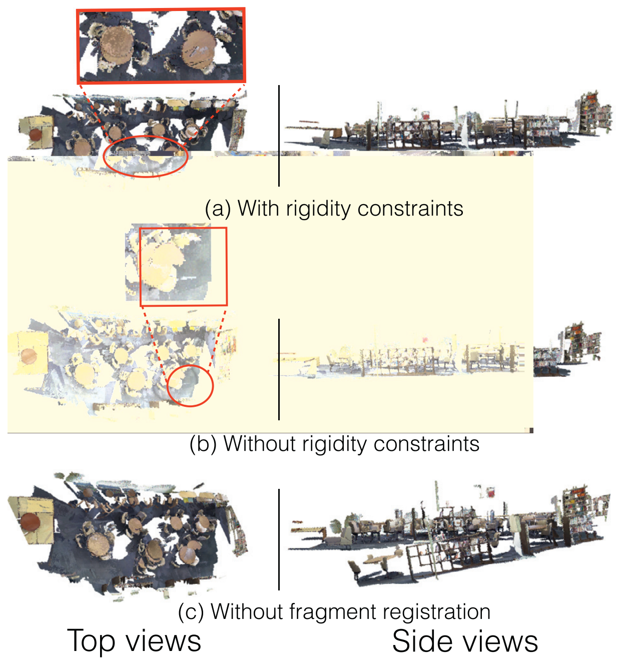

(a) With rigidity constraints
(b) Without rigidity constraints
(c) Without fragment registration
Top views
Side views

Fig. 17. Results obtained with the dataset Library-2 with our method, with/without using the rigidity constraints, and without using the fragment registration. Not using the fragment registration produces poor results. When using or not our introduced geometric constraints, re-distribution of errors in the graph differ. As we can see in the red circles, when not using the geometric constraints the error appears to be concentrated in some parts of the scene. On the contrary, when using geometric constraints errors remain in more objects but with smaller magnitude. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

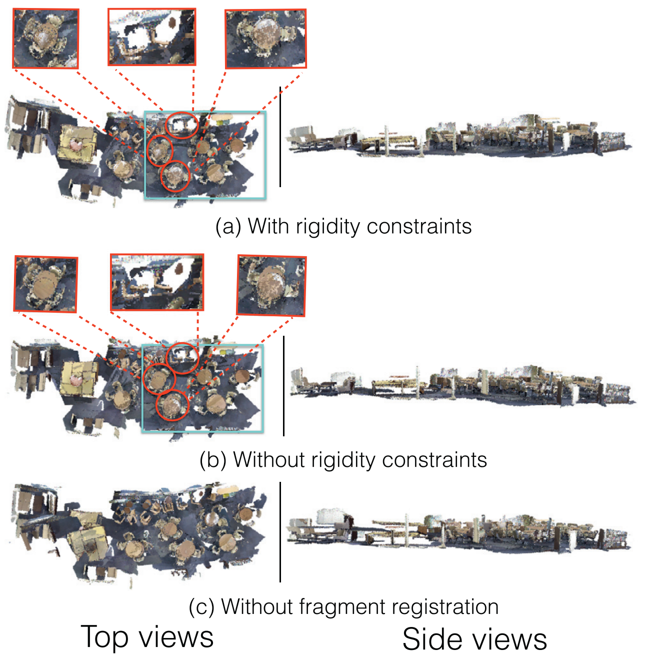

(a) With rigidity constraints
(b) Without rigidity constraints
(c) Without fragment registration
Top views
Side views

Fig. 18. Results obtained with dataset Library-3 with our method, with/without using the rigidity constraints, and without using the fragment registration. In this dataset, inside the blue squared region, there are more loops than with data Library-2. Therefore, we could have more geometric constraints, which resulted in less deformations when using geometric constraints than when not using geometric constraints (as observed in the red circled). Not using the fragment registration produces poor results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

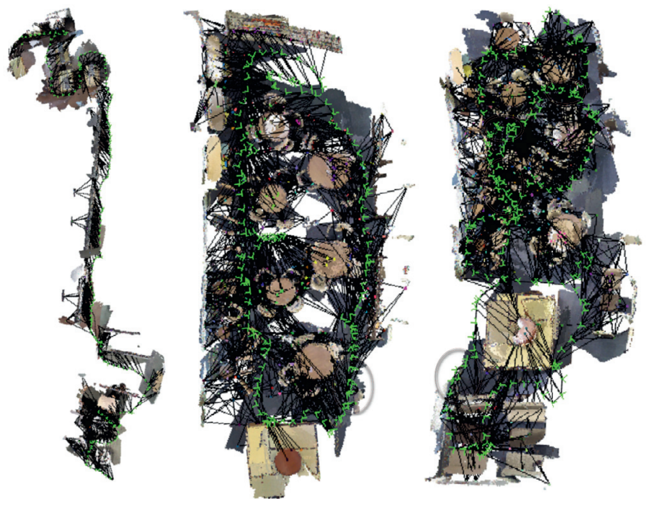


Fig. 19. The final graph of our constructed 3D model superimposed over the 3D model. Green spheres represent keyframe vertices. Black lines represent rigidity constraints and visibility constraints. Red circles indicate loops: there are no loops in the dataset LIBRARY, one loop in the dataset LIBRARY-2 and multiple loops in the dataset LIBRARY-3. Our proposed method produces consistent 3D models with consistent camera paths (the loops are closed). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

we can see that when using rigidity constraints the optimization tries to distribute the errors in a way that minimizes the geometric deformations of each object (but then, the errors remain in more objects). On the contrary, when not using the rigidity constraints the errors appear to be concentrated in some parts of the scene. Therefore the number of objects with significant deformations is reduced, but the deformations in these objects are larger than when using the rigidity constraints. With the dataset LIBRARY-3, inside the blue squared region of Fig. 18, there are more loops in the camera path than with the dataset LIBRARY-2 (as we can also see in Fig. 19). Therefore, we could build more rigidity constraints, which resulted in less deformations when using rigidity constraints than when not using rigidity constraints (as observed in the red circled).

Fig. 19 shows the global graph obtained by our method with the datasets LIBRARY, LIBRARY-2 and LIBRARY-3. In this figure, green spheres represent keyframe vertices, black lines represent rigidity and visibility constraints. We can see that our method generates many rigidity constraints between different planar patches that compose the scene, which enables us to consistently correct deformations.

6. 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 global mapping organizes all the local 3D models into a global model in an undeformed way using fragment registration in the graph optimization framework. Introducing rigidity 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.

References

3D Scene Dataset:,. http://www.stanford.edu/~qianyizh/projects/scenedata.html.
Besl, P.J., McKay, N.D., 1992. A method for registration of 3-d shapes. IEEE Trans. PAMI Vol. 14 (No. 2), 239–256.
Blais, G., Levine, M.D., 1995. Registering multi vie range data to create 3d computer objects. IEEE Trans. PAMI Vol. 17 (No. 8), 820–824.
Cameral, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W., 2011. G2O: a general framework for graph optimisation. In: Proceedings of ICRA.
Chen, J., Bautembach, D., Izadi, S., 2013. Scalable real-time volumetric surface reconstruction. ACM Trans. Graph 32 (4), 1132:1–113:8.
Curless, B., Levoy, M., 1996. A volumetric method for building complex models from range images. In: Proceedings of SIGGRAPH, pp. 303–312.
Hansard, M., Lee, S., Choi, O., Horaud, R.P., 2012. Time-of-flight cameras: principles, methods and applications. Springer Brief in Computer Science. Springer Science & Business Media.
Henry, P., Fox, D., Bhowmik, A., Mongia, R., 2013. Patch volumes: segmentation-based consistent mapping with RGB-D cameras. In: Proceedings of 3DV'13.
Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D., 2012. RGB-D mapping: using kinect-style depth cameras for dense 3d modelling of indoor environments. Int. J. Rob. Res. 31 (5), 647–663.
Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., Kolb, A., 2013. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: Proceedings of International Conference on 3D Vision (3DV).
Lazaros, N., Sirakoulis, G.C., Gasteratos, A., 2008. Review of stereo vision algorithms: from software to hardware. Int. J. Optomech. 2, 435–462.
Lowe, D.G., 1999. Object recognition from local scale-invariant features. In: Proceedings of ICCV, pp. 1150–1157.
Meilland, M., Comport, A., 2013. On unifying key-frame and voxel-based dense visual SLAM at large scales. In: Proceedings of IROS.
Neibner, M., Zollhofer, M., Izadi, S., Stamminger, M., 2013. Real-time 3d reconstruction at scale using voxel hashing. ACM Trans. Graph 32 (6), 169:1–169:11.
Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A., 2011. Kinectfusion: real-time dense surface mapping and tracking. In: Proceedings of ISMAR'11, pp. 127–136.
Pfister, H., Zwicker, M., Baar, J., Gross, M., 2000. Surfels: surface elements as rendering primitives. In: ACM Transactions on Graphics (Proceedings of SIGGRAPH'00) RGB-d SLAM dataset and benchmark. http://vision.in.tum.de/data/datasets/rgbd-dataset.
Roth, H., Vona, M., 2012. Moving volume kinectfusion. In: Proceedings of BMVC.
Rusinkiewicz, S., Hall-Holt, O., Levoy, M., 2002. Real-time 3d model acquisition. ACM Trans. Graph 21 (3), 438–446.
Segal, A., Haehnel, D., Thrun, S., 2009. Generalized-ICP. In: Robotics: Science and Systems.
Thomas, D., Sugimoto, A., 2013. A flexible scene representation for 3d reconstruction using an RGB-D camera. Proceeding of ICCV.
Thomas, D., Sugimoto, A., 2014. A two-stage strategy for real-time dense 3d reconstruction of large-scale scenes. In: Proceedings of ECCV Workshops'14 (CDC4CV).
Weise, T., Wismer, T., Leibe, B., Gool, L., 2009. In-hand scanning with online loop closure. In: Proceedings of ICCV Workshops'09, pp. 1630–1637.
Whelan, T., McDonald, J., Kaess, M., Fallon, M., Johansson, H., Leonard, J., 2012. Kintinuous: Spatially Extended Kinectfusion. In: Proceedings of RSS Workshop on RGB-D: Advanced Reasoning with Depth Camera.
Zeng, M., Zhao, F., Zheng, J., Liu, X., 2013. Octree-based fusion for realtime 3d reconstruction. Trans. Graph. Models 75 (3), 126–136.
Zhou, Q.-Y., Koltun, V., 2013. Dense scene reconstruction with points of interest. ACM Trans. Graph. 32 (4), 112:1–112:8.
Zhou, Q.-Y., Miller, S., Koltun, V., 2013. Elastic fragments for dense scene reconstruction. In: Proceedings of ICCV.