3D modeling of large-scale indoor scenes using RGB-D cameras

Diego Thomas, Akihiro Sugimoto
National Institute of Informatics, Tokyo, Japan.

Hand-held consumer depth cameras have become a commodity tool for constructing 3D models of indoor environments in real time. Recently, much work has been done in developing methods to fuse low quality depth images into a single dense and high fidelity global 3D model. However, most techniques have difficulties in scaling up to large-scale scenes. These difficulties arise due to the need to manipulate a large amount of data. For example, parts of the 3D model often need to be re-positioned to close loops and keep consistency of the 3D model. The low level 3D representations (e.g. point-based or voxel-based) allow reasoning at the point-based or pixel-based level only and offer low flexibility for manipulating the 3D model that is being built live. Such representations are thus not convenient for large-scale reconstruction. In this paper, we present our recent work on object-based level 3D representation using planar patches with geometric textures (Bump images), confidence texture and color texture. We also present an algorithm to build such a representation in real-time using RGB-D cameras, even for large-scale scenes. We show that our 3D representation allows for detailed, compact and consistent 3D reconstruction of large-scale indoor scenes.

1. Introduction

In the computer vision community, the task of constructing a 3D model of a real object (i.e. a mathematical representation of its 3D surface) from images has attracted an ever-growing interest in a last few decades. Constructing such detailed 3D models of real objects is of great interest for many applications such as scientific simulations, digitization of cultural heritage or entertainment (e.g. special effects in movies). In general, the 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) integrating (i.e. fusing) all measurements into a single mathematical 3D representation and (4) correcting possible deformations (due to error propagation for example) and refining the output 3D model (e.g. filling holes).

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. On the other hand, structured light and time of flight are the most popular techniques for active sensing. Consumer depth cameras such as the Microsoft Kinect camera or the Asus Xtion pro camera are emerging active sensors that produce low quality depth images at video rate and at low cost. These sensors have raised much interest for many applications in computer vision, 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 small. As a consequence, well-known standard registration algorithms such as the variants of the Iterative Closest Point (ICP) [2] can be used efficiently. Moreover many measurements available for each point of the scene can be accumulated and merged to compensate for the low quality of a single depth image. A popular example of a system that takes advantage of consumer depth cameras for 3D modeling is KinectFusion [8]. In this system, a linearized version of GICP [10] is used in the frame-to-global-model registration framework to align successive depth images, which are accumulated into a Volumetric Truncated Signed Distance Function (TSDF) [5] using the running average. With using rather simple, well-established tools, impressive 3D reconstructions at interactive rate could be obtained, which demonstrates the potential of Kinect-like cameras for fine 3D modeling.

Another new interesting property of consumer depth cameras is that they can be held by hand and thus they allow users to move freely. As a consequence, users can easily capture long sequences of RGB-D images and aim at reconstructing large-scale scenes. With this new possibility, new challenges also arise: how to deal with error propagation and large amount of data. In other words, how can we minimize deformations of the produced 3D model while keeping fine details, even at large-scale?

In the last three years, there has been a lot of research on extending previous work to allow large-scale 3D reconstruction [4, 6, 7, 9, 13, 14, 15, 16]. Noticeable works employ hash tables for efficient storage of volumetric data [7], patch volumes to close loops on the fly [6] and non-rigid registration of sub-models to reduce deformations [16]. Though recent work considerably

improved the scale and quality of 3D reconstruction using consumer depth cameras, existing methods still offer little flexibility for manipulating the 3D model that is being built. This is because most of existing works reason at the pixel level (voxels with volumetric TSDF or 3D vertices with meshes). Modifying the whole 3D model then becomes difficult. The work by Henry et.al. [6] is exceptional. Namely, it reasons at the level of objects, which allows simpler modifications of the overall structure of the 3D scene. Our work push forward in this direction by using a new 3D representation that allows easy reasoning at the object level while maintaining fine details, real-time performance and efficient storage.

This paper gives our recent work on flexible 3D scene representation based on planar patches with attributes, and an algorithm to build such a representation [11, 12]. In real-time, we track the camera and accumulate depth images into a semi-global 3D model that consists of a pair of RGB-D and Mask images. From this semi-global model, we segment at run time the target scene both in time and space. The input sequence of RGB-D images is uniformly segmented in time to build multiple accurate and un-deformed local 3D models. Each local 3D model is segmented in space by fitting multiple planes to which attributes such as bounding box, Bump image, color image and confidence image are attached. Each planar patch represents an object in the scene and our closing loops reduces to re-organizing all the objects in the scene (without re-computing their local geometry). As a side effect, our representation allows easy and interactive editing and 3D rendering at multiple levels of details.

2. Parametric 3D scene representation

We reason that parametric surfaces can be used to describe a 3D scene and thus represent an indoor scene as a set of planar patches having attributes. To each planar patch detected in the scene, we attach as its attributes three 2D images in addition to information that identifies the planar patch.

The three images are a three-channel Bump image, a 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 corresponding to the pixel from the lower left corner of the pixel. The Mask image encodes the confidence for accumulated data at a point and the Color image encodes the color of each point.

The Bump image encodes the local geometry, which allows us to accurately represent the geometry of the 3D

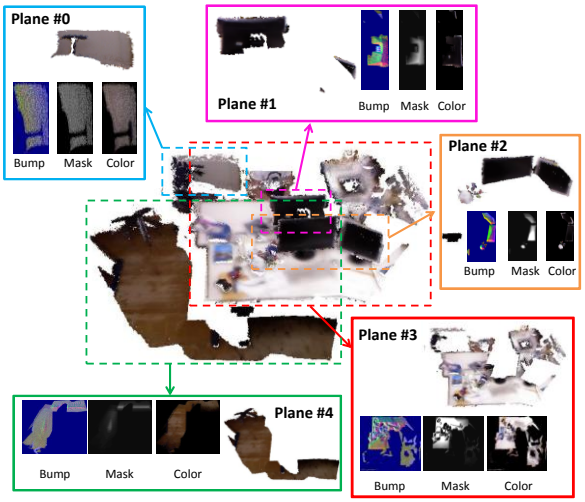


Fig.1. 3D representation for an indoor scene [11]. The target scene is segmented into multiple coplanar parts. Each segment **S** is modeled by the equation of the plane **P** that best fits the set of 3D points in **S**, the 2D bounding box of the projection of all 3D points in **S** into the plane **P**, and three 2D images that encode local geometry (Bump), color (Color) and confidence of measurements (Mask).

scene while using less memory. In addition, adding, removing or updating points is executed easily and efficiently in our representation, because we are manipulating 2D images. Figure 1 illustrates our proposed 3D representation for an indoor scene. Details on the 3D representation can be found in [11].

3. A two-stage algorithm for 3D scene modeling

We build a large-scale 3D model by breaking the whole sequence of RGB-D images into short subsequences (in our experiments we used subsequences of 100 frames) and taking a two-stage strategy (Fig. 2). The two stages are called local mapping and global mapping. Details of this algorithm can be found in [12].

3.1. Local Mapping

We use our 3D scene representation to build local 3D models of a target scene. To build local 3D models from short subsequences of RGB-D images in real-time we employ the frame-to-global-model framework. Rather than using the set of planar patches for the global model, we employ an additional semi-global model, which consists of a pair of RGB-D and Mask images. 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 inefficient (as many points on a patch may disappear from the current viewing frustum and rendering such points is useless).

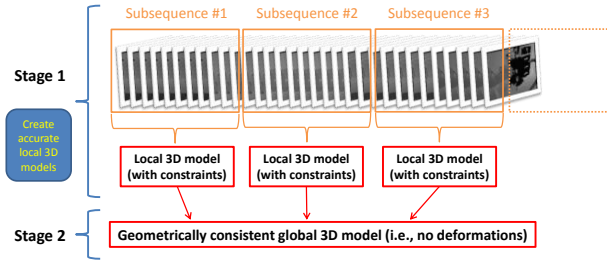


Fig.2. Overview of our two-stage strategy [12]. The first stage generates local structured 3D models with geometric constraints from short subsequences of RGB-D images. The second stage organizes all local 3D models into a single common global model in a geometrically consistent manner to minimize deformations.

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. This allows us to keep real-time performance with state-of-the-art accuracy.

The semi-global model is initialized 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 organization of points in the 2D image. At every input frame, attributes of each planar patch are updated using the semi-global model as follows. Each point **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 **p** if the mask value of **p** is higher than that at the projected pixel.

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

3.2. Global Mapping

The objective of the second stage (i.e. global mapping) 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.

In order to compensate for the accumulation of registration errors, we build a graph where each vertex represents either a planar patch generated from the local mapping or a keyframe; and edges represent keyframe pose constrains [6], visibility constraints [6], geometric constraints, or identity constraints of patches over keyframes. The graph is maintained geometrically consistent with newly generated local 3D models, which allows us to build a global 3D model with minimal deformations (details can be found in [12]).

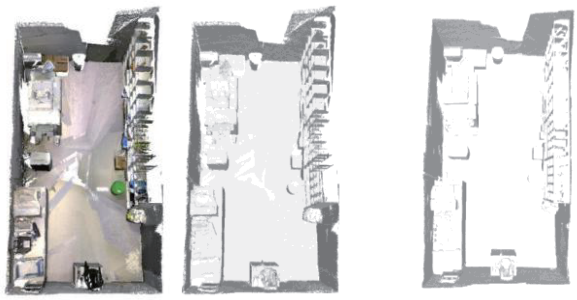
The global mapping is capable of (1) always aligning new local 3D models to un-deformed subsets (called fragments) of the global model and (2) introducing new constraints (called identity constraints) at each successful rigid alignment, rather than merging planar patches representing the same object. This capability allows more flexibility in re-organizing the global 3D 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 optimization framework (namely, the g2o framework [4]) is then used to guarantee geometric consistency of the global model (thus reducing deformations).

Note that the geometric constraints enable us to redistribute errors more coherently with respect to the 3D geometry of the scene. This is crucial because in general, drift errors derived from camera tracking do not uniformly arise. 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).

4. Experiments

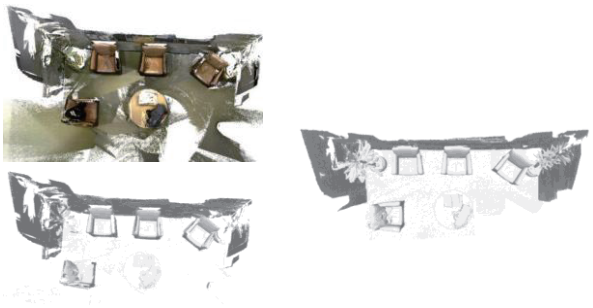
We evaluated our method in several situations using real data. All scenes were captured at 30 fps. We used a resolution of 0.4 cm for attribute images in all cases. The CPU we used was an Intel Xeon processor with 3.47 GHz and the GPU was a NVIDIA GeForce GTX 580. Our method runs at about 28 fps with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

Figures 3 and 4 show results obtained by our method using data COPYROOM and data LOUNGE, respectively, (captured with an Xtion Pro Live camera) available at [1]. We compared the results obtained by our method with results [16] 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 [16]. We remark that our results were produced on-line, while those by [16] were off-line. Moreover, with our method we could generate textured 3D models while texture is not available in the results by [16].



Our proposed method [12] Zhou et.al [16]

Fig.3. Data COPYROOM. Our method was able to successfully close the loop on the fly. The result obtained with our method in real-time were comparable to those obtained by [16].



Our proposed method [12] Zhou et.al [16]

Fig.4. Data LOUNGE. Our method was able to build accurate 3D models with fine details and overall consistent geometry. Results obtained by our method in real-time were comparable to those obtained in off-line by [16].

One interesting side effect of using our 3D scene representation is the capability of easily editing the generated 3D model (this reduces to editing 2D images) as well as the capability of performing rendering at multiple levels of details quite straightforwardly. Since we are manipulating texture images, it is possible to render the 3D model at multiple levels of details using normal and texture images in different resolutions mapped to a coarse 3D mesh.

5. Conclusion

We presented our recent work on parametric 3D scene representation based on planar patches with attributes. We also introduced an algorithm that allows building a 3D model using our 3D scene representation in real-time. Our 3D reconstruction algorithm employs a two-stage strategy where local 3D models are build in the first stage (local mapping) from successive short-time subsequences of RGB-D images. To achieve real-time performance, an additional semi-global model is used. In the second stage (global mapping) all local 3D models are successively integrated into a global graph with constraints that allow repositioning all planar patches in a consistent geometric manner.

References

1) 3D Scene Dataset: http://www.stanford.edu/qianyizh/projects/scenedata.html.
2) P. Besl and N. McKay: A method for registration of 3-d shapes, *IEEE Trans on PAMI*, 14(2):239–256, 1992.
3) R. Cameral, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard: g20: A general framework for graph optimization, Proceeding of ICRA, 2011.
4) J. Chen, D. Bautembac, and S. Izadi: Scalable real-time volumetric surface reconstruction, ACM transaction on Graphics, 32(4):113:1-113:16, 2013.
5) B. Curless and M. Levoy: A volumetric method for building complex models from range images, In *ACM Transactions on Graphics (SIGGRAPH)*, 1996.
6) P. Henry, D. Fox, A. Bhowmik, and R. Mongia: Patch volumes: Segmentation based consistent mapping with RGB-D cameras, Proceeding of 3DV'13, 2013.
7) M. Neibner, M. Zollhofer, S. Izadi, and M. Stamminger: Real-time 3D reconstruction at scale using voxel hashing, ACM Transaction on Graphics, 32(6):169:1-169:11, 2013.
8) R. Newcombe, S. Izadi, O. Hillinges, D. Molyneaux, d. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon: KinectFusion: Real-time dense surface mapping and tracking, Proceeding of ISMAR'11, 127-136, 2011.
9) H. Roth, and M. Vona: Moving volume kinectfusion, Proceeding of BMVC, 2012.
10) A. Segal, D. Haehnel, S. Thrun: Generalized-ICP, Robotics: Science and Systems, 2009.
11) D. Thomas, and A. Sugimoto: A flexible scene representation for 3D reconstruction using RGB-D camera, Proceeding of ICCV, 2013.
12) D. Thomas, and A. Sugimoto: A two-stage strategy for real-time dense 3D reconstruction of large-scale scenes, Proceeding of CDC4CV (ECCV workshop), 2014.
13) T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johansson, and J. Leonard: Kintinuous: Spatially extended kinectfusion, Proceeding of RSS Workshop on RGB-D: Advanced Reasoning with Depth Camera, 2012.
14) 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.
15) Q.-Y. Zhou and V. Koltun: Dense scene reconstruction with points of interest, ACM Transaction on Graphics, 32(4):112:1–112:8, 2013.
16) Q.-Y. Zhou, S. Miller and V. Koltun: Elastic fragments for dense scene reconstruction, Proceeding of ICCV, 2013.