DOI: 10.1111/cgf.12495
Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

*Volume 33 (2014), Number 7*

# Automatic 3D Indoor Scene Updating with RGBD Cameras

Zhenbao Liu[1], Sicong Tang[1], Weiwei Xu[2,*], Shuhui Bu[1], Junwei Han[1], Kun Zhou[3]

[1] Northwestern Polytechnical University, China
[2] Hangzhou Normal University, China; [3] Zhejiang University, China. [*] Corresponding author.

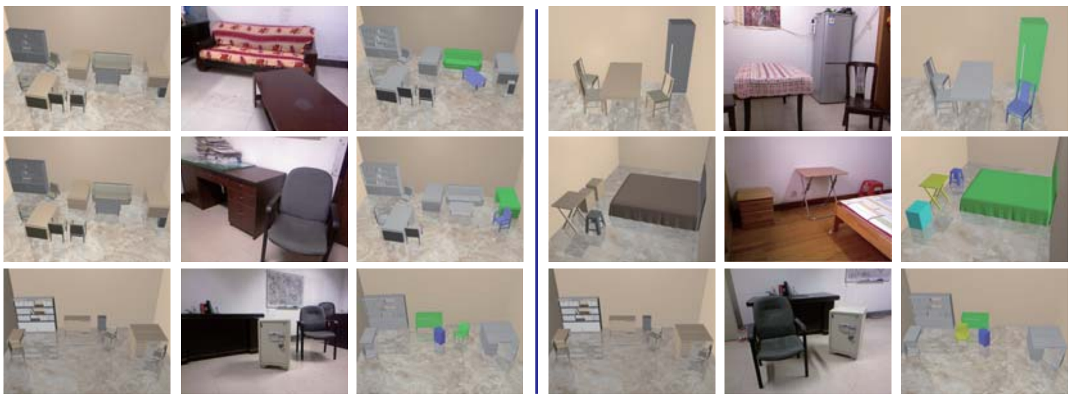

**Figure 1:** Six groups of scene update results, and comparison between original 3D scene (left) and automatically updated 3D scene (right) by capturing RGBD image (middle) with local variation. The objects in green are seen as static reference, relative to moved objects in other colors.

**Abstract**

*Since indoor scenes are frequently changed in daily life, such as re-layout of furniture, the 3D reconstructions for them should be flexible and easy to update. We present an automatic 3D scene update algorithm to indoor scenes by capturing scene variation with RGBD cameras. We assume an initial scene has been reconstructed in advance in manual or other semi-automatic way before the change, and automatically update the reconstruction according to the newly captured RGBD images of the real scene update. It starts with an automatic segmentation process without manual interaction, which benefits from accurate labeling training from the initial 3D scene. After the segmentation, objects captured by RGBD camera are extracted to form a local updated scene. We formulate an optimization problem to compare to the initial scene to locate moved objects. The moved objects are then integrated with static objects in the initial scene to generate a new 3D scene. We demonstrate the efficiency and robustness of our approach by updating the 3D scene of several real-world scenes.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

## 1. Introduction

In recent years, 3D scene modeling has been a hot research topic in computer graphics, which aims to simultaneously model single objects and the spatial layout between them in a scene. Its recent research includes sketch based scene modeling, inference based scene modeling, and scene reconstruction with RGBD cameras [SXZ*12] [NXS12]. Although such methods can reconstruct a static scene in 3D efficiently, it is still of heavy computational and interactive loads to apply them to a scene updated frequently. For instance, furniture in an indoor scene may be moved frequently, e.g. chairs, and it is tedious to apply the methods above to reconstruct a

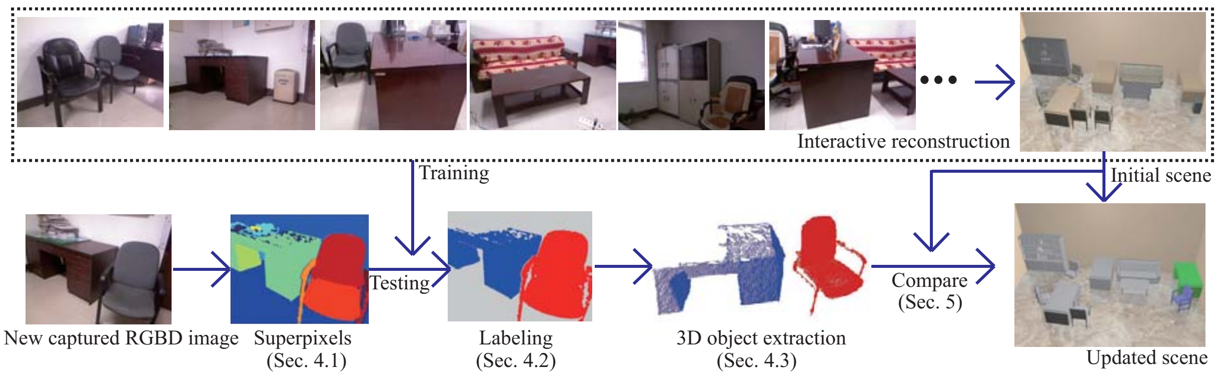

**Figure 2:** Flowchart of our algorithm.

3D scene from scratch. Thus, it is desirable to have such a solution make a reconstructed 3D indoor scene updated according to the change of its real scene.

In this paper, we propose a fully automatic modeling solution to scene update, which attempts to generate a new 3D scene by means of capturing local scene variation with an RGBD camera. We assume that a complete indoor scene has been scanned by a RGBD camera before its change. Its RGBD images are segmented into semantic regions manually or semi-automatically with user-guided interactive algorithm, and its 3D scene has been modeled according to the segmentation, which contains 3D mesh objects such as many pieces of furniture [SXZ*12]. In practice, such an interactive algorithm requires users to make a number of strokes unceasingly so as to adjust semantic segmentation until it reaches a satisfactory effect. The whole process takes a long time and it is time consuming for common consumers to use this tool to reconstruct a 3D scene even there are only minor changes of the scene.

Our approach only requires the user to take RGBD images of changed scene to update an already reconstructed 3D scene. The RGBD image is first automatically segmented into superpixels, which are labeled based on reconstructed scene information after recognizing wall and floor and spatial continuity check. After obtaining the changed scene with extracted 3D objects, we formulate an optimization problem for comparison between the changed scene and the original scene to automatically determine moved objects and their new positions and orientations in the updated scene. Fig. 2 illustrates the pipeline of our approach.

The key technical challenge in our approach is how to determine where the scene updating occurs. Because the original scene and new partial scene are captured in different global coordinates, we first find one or more static objects in two scenes as common reference. Rotating and translating them can make new partial scene register to the original complete scene. When two scenes align, moved objects are automatically placed in the new positions and orientations according to the relative relation to static objects kept in the scanned scene. In order to build right correspondences between static objects, we match two scenes and formulate an objective function, which is maximized only when all the static objects rightly overlap with each other in two different scenes. Each static or moved object in partial scene is replaced with its matching 3D model in the original scene, and the 3D scene update is accomplished. Therefore, our work supports the movement or removal of objects in the original scene, but does not support adding new object into the original scene.

**Contributions**. We have implemented the whole modeling pipeline for scene update in a prototyping system and demonstrated its usefulness with several real-world scenes. Our approach is made possible with the following two technical contributions:

1. An optimization problem for scene comparison and update is proposed. Given candidate object pairs between the original scene and changed local scene captured with RGBD camera, our method considers both similarity term and geometric consistency term to optimize an objective function. The moved objects and their new positions and orientations are obtained and the scene is updated.
2. To provide more accurate initial solution to the above optimization, object pairs are selected only in the same class. Differently from previous RGBD labeling works based on a large indoor scene data set, we only utilize the historical scene information including original labeled RGBD images and 3D reconstruction, to train a random forest classifier for labeling new captured RGBD images in the changed scene.

## 2. Related works

**RGBD semantic segmentation**. As an early work, Silberman and Fergus [SF11] put forward a method by using Conditional Random Field (CRF) model, in which a data term and compatibility term are defined according to the geometry and texture feature of the given training samples from a huge database they built. Koppula et al. [KAJS11] present an algorithm based on the appearance, geometry, and geometrical context of patches generated by over-segmentation of RGBD images, and an important improvement is that the labeling accuracy is increased significantly by introducing geometrical context. Gupta et al. [GAM13] combine the depth information with texture information to extract initial contours of RGBD images and generate hierarchical segmentation. After that, they introduce a SVM classifier with additive kernels to yield probability of contour consistent with training set, and select high quality contours to form close regions. Shao et al. [SXZ*12] present a semi-automatic segmentation method, and add user interaction to the CRF-based model. Nan et al. [NXS12] over-segment the input point cloud into patches by an interesting search-classify strategy. In our work, we will partially depend on some state-of-the-art segmentation techniques, such as the training of contour extraction [GAM13]. In our labeling method, the whole process requires no manual interaction, which benefits from accurate labeling training on a small quantity of initial scenes, rather than on a large set of indoor scene data.

**Indoor scene modeling with RGBD camera**. Kim et al. [KMYG12] present an efficient method to acquire 3D indoor environments with variability and repetition. Their work segments single 3D point cloud scanned via real-time SLAM technique, classifies it into plausible objects, recognizes them using primitive fitting and connected component analysis, and extracts their pose parameters. They also introduce a hierarchical structure composed of super-points, parts, and objects, which can easily support depth data segmentation, fitting, and its match with learned models. With the similar objective, Nan et al. [NXS12] present another different solution to indoor scene modeling. They introduce template fitting to segmented point cloud, and use non-rigid ICP algorithm to minimize Euclidean distances between point cloud and its candidate templates. These selected templates after transformation form a whole 3D scene. Shao et al. [SXZ*12] present an interactive approach to modeling indoor scene by semi-automatically segmenting RGBD images. Our purpose is different from theirs. We focus on scene update problem based on the assumption that the indoor scene has been reconstructed in advance.

**Furniture layout**. Some researchers look 3D indoor scene modeling from another point of view. Yu et al. [YYT*11] stress the quality of furniture arrangements for indoor scene, with the prior condition that all the furniture models have already existed in the given scene. Their method optimizes furniture arrangements in an automatic way, by considering the ergonomic factors, visibility, and accessibility as the measurable indicator of specific placement of furniture models. The work from Merrell et al. [MSL*11] allows user to modify the arrangement when multiple placement suggestions are provided, so that user can add some user-specified constraints on their designed scene. In their objective function both the functional criteria and visual criteria are held. Xu et al. [XCF*13] propose a sketch-based scene generation algorithm, which can retrieve the furnitures and their placements at the same time by discovering the spatial and structural relationships among candidate models. Compared with the above works, we solve the modeling problem of a real indoor scene by capturing its RGBD data. We do not require optimization of reconstructed scene because it has already been reasonable by means of real placements from human.

**Shape matching**. In our work, the challenge we face is not global shape matching but partial shape matching [LBZ*13]. Recent representative methods include Shape Google [BBGO11] and sketch based retrieval [ERB*12]. In order to match single-view Kinect scan to high-quality 3D models, Shen et al. [SFCH12] propose to recover the underlying structure of a scanned object by assembling suitable parts obtained from the repository models. Our problem is different: we require the comparison between depth image and 3D model and furthermore the new placement position must be determined by obtaining transformation parameters between partial object and 3D model. We design an optimization framework to reach the two objectives simultaneously.

## 3. Overview

We denote the 3D indoor scene reconstructed from real scene before its change by $P$. The $P$ is obtained using the interactive approach in [SXZ*12]. It not only contains reconstructed 3D objects but also the captured RGBD images, their segmentation, and labeling information. The captured RGBD image of a changed scene is denoted by $Q$. $P$ and $Q$ are the inputs of our pipeline, illustrated in Fig. 2. The updated 3D scene according to these images is denoted by $U$, the final output of our pipeline. We denote those constructed 3D objects not moved in the both scenes by static objects. Those objects are moved to form the updated scene are denoted by moved objects.

Our method first performs automatic superpixel segmentation and labeling for $Q$, which is based on classification models trained on existent segmented and labeled RGBD images of $P$. Objects in $Q$ are extracted by label consistency of neighbor superpixels to form a 3D point cloud scene, which is compared to the original 3D scene with mesh models. We set up a discrete optimization subjective function for scene matching and update. It is maximized only when all the static objects rightly overlap with each other in two different scenes. The optimized result is the correspondence between those static objects. The static objects can make new partial scene to register to the original scene. Moved objects are placed in the new scene according to the relative position and orientation to static objects. Each static or moved object is replaced with its matching 3D model in $P$ to accomplish update and obtain $U$.

## 4. Automatic object extraction of a captured scene

The RGBD image of $Q$ is segmented into superpixels, which then are labeled according to training data of $P$. 3D objects are extracted by label consistency of neighbor superpixels.

### 4.1. Superpixel generation of RGBD images

**Over-segmentation of RGBD images**. We first over-segment $Q$ into superpixels. Many related works can be found to reach the similar objective [BLHW13]. In our work, we first adopt local features to detect contours, and form many regions surrounded by these contours with watershed transform. Color and texture gradients in the image captured by RGB camera are extracted as monocular features, while the depth gradient and concavity represented with dihedral angle are obtained to be geometric features. They are computed in multi-scale manner. Features of RGBD images in original scenes $P$ are used to train support vector machines with additive kernels [GAM13]. For new captured images, color, texture, and geometric features are inputted into the classifier, and the probability of each pixel belonging to real contour can be looked as its strength. Because a large number of contours lead to many small close regions, a hierarchical segmentation can be formed by merging these regions according to the strength of contour features. We set the number of superpixels to 30. The second column of Fig. 3 shows the over-segmentation results for three captured RGBD images.

**Recognition of wall and floor**. Before labeling RGBD images of $Q$, we first judge the floor and wall individually because they are easily recognized without any training. A simple way to recognize floor, is to use the height of each pixel. The region is looked as floor when the height values are below a very small threshold. For wall recognition our algorithm supposes that a wall always locates at the farthest place in the horizontal direction. Based on that, we project all the points scanned by RGBD sensor onto the plane parallel to ground, and set the projection location of camera as the original point. Then the rectangular coordinate system in the projection plane is converted into the polar coordinate system. For all the points between an uniform angle interval $[\theta, \theta + \Delta\theta]$, we normalize their distances to the origin between 0 and 1. The points with distance larger than 0.95 are regarded as points belonging to wall. For each superpixel of the given image, all the pixels vote for wall. The third column of Fig. 3 shows the wall and floor detection results.

**Spatial continuity check**. In order to make each superpixel more reasonable and consistent with human perception, we design a re-clustering algorithm to check internal spatial continuity of each superpixel according to the depth values of pixels. The superpixel with more than one spatially discontinuous parts needs more attention. Specifically, for all the pixels in one superpixel, we re-cluster them into hierarchical clusters, and the distance metric between any two clusters is their nearest spatial distance. When the inter-cluster error exceeds a specified threshold, the clustering will be stopped. Finally, some small isolated areas and scattered noisy points are filtered out. Superpixels after spatial continuity check are shown in the right column of Fig. 3.

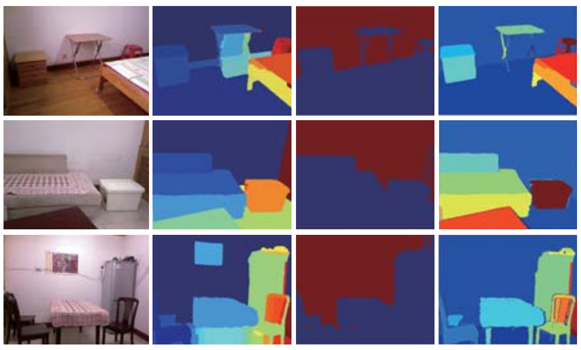
**Figure 3:** Superpixel generation of RGBD images.

### 4.2. Automatic labeling

Our objective here is to make superpixels semantically labeled with known class labels. These classes are defined in advance according to the original scenes and their RGBD images, which have been labeled along with 3D reconstruction. We use these information generated during the historical reconstruction process of the original scene to guide the labeling of the new captured RGBD images.

**Training set**. Existent object labels of the original scene contain 6 key classes, sofa, table, chair, bed, cabinet, and background. The background class consists of some small objects such as books. Considering the consistency between the changed scene and the original scene, we choose labeled RGBD images of the original scene as training samples. Suppose there are $t$ training images with labels. For each training image, we generate superpixels by means of over-segmentation mentioned in Sec. 4.1. The label of each superpixel is estimated according to the majority of pixels labeled in advance. In order to enhance generalization ability and reduce the influence of small sample number $t$, we randomly sample 50 small blocks in each superpixel, and the block size is fixed to $10 \times 10$ pixels. These blocks constitute a large training set.

**Classification**. Similarly, many blocks are extracted from RGBD images captured in the changed scene. We estimate the labels of these blocks with trained classifiers on the training set. Random forest classifier is designed to determine the label of each block represented with features. Specifically, we adopt the average HSV value of pixels and its histogram, gradient histogram, and several geometric properties. They are the average height, the average normalized depth value, the average of fitting plane area, and the angle between normal and the vertical direction. For each pixel, fitting plane area in its neighborhood is computed by extracting plane primitive using RANSAC algorithm [SWK07]. In the testing stage, each tree in the forest yields a label for an input block at the last leaf node after passing its parent nodes with a series of binary classifications. All the trees vote to decide the final label for the block. Finally, the label of each superpixel is also voted by the majority of their sampled blocks. Fig. 4 shows a group of labeling results.

### 4.3. 3D object extraction

Each object is extracted by merging superpixels according to label consistency of neighbor superpixels, which can be further converted into 3D point cloud according to its depth values and camera coordinate system. For neighbor superpixels belonging to the same body but with different labels, for example, the swivel chair (bottom right) in Fig. 4 where upper part and lower part are labeled inconsistently, we introduce a judgment rule based on plane projection. Considering that the objects in an indoor scene are usually placed vertically to ground, we orthogonally project the point cloud of superpixel onto the ground to compute the overlap between neighbor superpixels. They will be merged if the overlap area is larger than 50% of the smaller one between convex hulls of superpixels. It should be noted that although 3D object can be extracted, its real label can not be accurately determined. It may have multiple labels temporarily, which will assist in selecting candidate correspondence pairs for objects in two different scenes.

## 5. Scene update

When the scene changes, only local variation is taken with an RGBD camera. Given $P$ containing $n_P$ 3D mesh objects, and $Q$ having $n_Q$ objects only with one partial view, the problem of scene comparison and update is formulated into combinatorially mapping from $n_Q$ objects to $n_P$ objects. $Q$ is represented with automatically segmented and labeled RGB and depth images, and point cloud generated for separate partial objects. $P$ is composed of semi-automatically segmented and labeled RGB and depth images, and reconstruction with 3D furniture models. The mapping is based on features extracted from appearance, color, geometry, and orientation. A mapping constraint is imposed to allow one object from $Q$ to match at most one object from $P$. We define a matching score term for each candidate matching pair $m = \{p, q\}$, which measures how well object $q$ in $Q$ matches object $p$ in $P$. Furthermore, we take into account that the relative relation of two objects should be added to infer whether object moves. Another term named pairwise consistency degree is introduced to measure how compatible a pair of objects in $Q$ is with another pair of objects in $P$. For example, a sofa is oriented toward a tea table and they are close in the original scene. The pairwise consistency degree judges whether the relation keeps invariable in the changed scene and how much it changes.

**Optimization problem**. Given each candidate match $m$, we introduce an optimization formulation to select static matching ones from $n$ candidate pairs by maximizing score of a certain quadratic function. The one-to-one constraints are imposed on each pair of objects to make sure that one object in $Q$ corresponds with at most one object in $P$.

$$\begin{aligned} \text{maximize} \;\; & \mathbf{m}^T W_s \mathbf{m} + \mathbf{m}^T W_c \mathbf{m}, \\ s.t. \;\; & A\mathbf{m} = \mathbf{1}, \\ & \mathbf{m} \in \{0,1\}^n. \end{aligned} \tag{1}$$

where $\mathbf{m}$ is an indicator vector such that $m_i = 1$ if the $i$-th pair of objects can match and zero otherwise. $W_s$ is a diagonal matrix with positive elements $\exp\{-S\}$ containing similarity functions, such that each function $S$ measures how similar two counterparts are in several aspects such as appearance, color, geometric shape, and placement orientation. $W_c$ is a non-diagonal matrix with positive elements $\exp\{-C\}$ expressed with geometric consistency functions $C$, which measure the compatibility degree of relative geometric relation of a pair of objects in $Q$ with another pair in $P$.

**One-to-one constraint**. One-to-one constraint is imposed on $\mathbf{m}$ by the above constraint equation, and the constraint matrix $A$ is set to a sparse matrix whose entry is either 0 or 1 based on correspondence between each row and the indicator vector.

**Candidate pair**. We only select pairs with the same class labels as candidate ones. How to obtain these labels is described in detail in Sec. 4.2. For object with uncertain multiple labels, for example, the swivel chair with two labels in the bottom right of Fig. 4, its candidate pairs will be selected in more than one classes.

**Solution to optimization**. The above objective function can be efficiently maximized using an integer projected fixed point method [LH09]. The optimization result is right correspondence only between static objects.

### 5.1. Similarity function

For each candidate pair, the similarity function $S$ measures its similarity by four terms. They are appearance similarity term $S_a$, color similarity term $S_c$, geometric shape similarity term $S_g$, and placement orientation similarity term $S_o$.

$$\begin{aligned} S(p,q) = \; & \omega_a S_a(p,q) + \omega_c S_c(p,q) \\ & + \omega_g S_g(p,q) + \omega_o S_o(p,q). \end{aligned} \tag{2}$$

The coefficients $\omega$ determine the relative weighting among the similarity terms, and balance the influence from several factors mentioned below.

**Appearance similarity term**. In order to quantify the

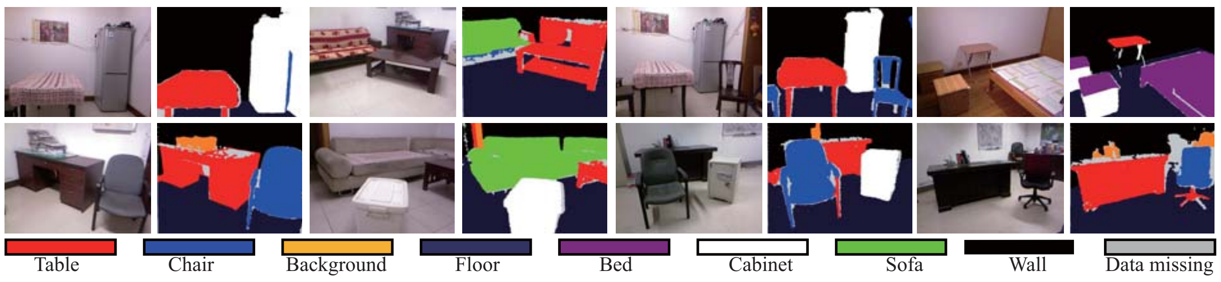

**Figure 4:** Labeling results of a group of captured RGBD images. The algorithm is tested on different types of indoor scenes, and effectively labels all the RGBD images of these changed indoor scenes into 9 classes.

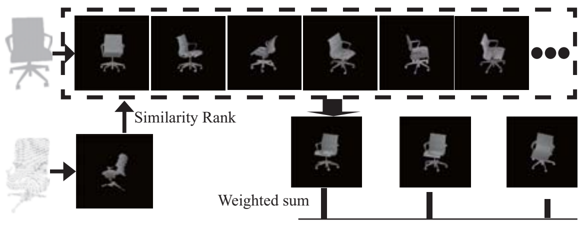

**Figure 5:** Appearance comparison between scanned partial object and reconstructed 3D object.

similarity, the depth image of scanned object compares to the appearances of 3D models. The appearances of 3D model are represented with its orthogonal projection onto virtual cameras uniformly distributed on spatial positions. Specifically, for each 3D model, we uniformly sample $N$ viewpoints on its bounding sphere centered at the origin, and set one virtual camera on each viewpoint which points towards the original center. We compare the region of depth image captured from object in $Q$ to all the projected images of the 3D model. Figure 5 shows the comparison process. In order to speed up the process, these projected views are first clustered into $K$ clusters. The scanned depth image matches each cluster center, and the most similar projection cluster is selected. The visual distance is defined as the weighted mean distance from the depth image to $R$ projected views in the selected cluster, which is expressed with the following equation.

$$S_a(p,q) = \sum_{r=1}^{R} \frac{d(I(q), I_r(p))}{\log_2(r+1)}. \tag{3}$$

where the weighting relies on the similarity degree between the depth image $I(q)$ and each projected view $I_r(p)$. While comparing two images, Zernike moment is employed to represent region images and a simple $L_1$ distance is adopted to measure their difference $d$.

**Color similarity term**. Not only appearance of object but also its color can be employed to distinguish it from other objects in the scene. Because the RGB image is known for 3D mesh model $p$, which can give some hints to assist in building the relation between $q$ and $p$. A color histogram $CH$ is introduced to represent the distributions in the images of objects $q$ and $p$, which are compared via the earth mover's distance (EMD). The term is defined as follows.

$$S_c(p,q) = EMD(CH(p), CH(q)). \tag{4}$$

**Geometric shape similarity term**. 3D partial point cloud $q$ compares with a complete 3D shape $p$ by geometric shape similarity and their alignment error. Coarse alignment is first performed by means of feature correspondence based on 3D key point detection [BWdBP13] and fast tree pruning based on Euclidean distance threshold, following the idea of [ZSCO*08]. We choose a recent Signature of Histograms of Orientations (SHOT) [TSDS10] as feature descriptor, which is robust to incomplete data, clutter, and sensor noise. By fast tree pruning method, initial correspondence is built to rotate and translate $q$ for alignment with $p$. Fine alignment starting from initial correspondence is realized by Expectation Maximization ICP [GP02], which is robust against outliers and tolerates partial data scarcity in the case of single view. Inspired by a recent work [TAR*10], we employ CUDA to enhance its computation efficiency, which can be sped up about 100 times on GPU of Nvidia GTX 750 than common CPU. The final alignment and its error are obtained, which is seen as the cost of geometric shape similarity term in the following equation. The error is normalized to $[0,1]$ by the largest error in all the candidate pairs. We obtain rigid transformation parameters $(R,T)$ including rotation matrix and translation vector, which will be used to move the 3D model in $P$ to the new position in $U$.

$$\begin{aligned} S_g(p,q) &= ICP(p,q), \\ (R,T) &= argmin\ ICP(p,q). \end{aligned} \tag{5}$$

**Placement orientation similarity term**. An important criterion used to determine whether an object in $Q$ moves, is its relative orientation $\theta$ to its nearest wall. We first use the transformation parameters $(R,T)$ computed above to transform the 3D model $p$. The point cloud of partial object $q$ in the changed scene is substituted with $p$ after transformation. The orientation of $p$ is estimated as the angle $\theta$ of principal component vector and the wall after the 3D model $p$ is projected onto ground plane. The term of placement orientation similarity between two objects is defined as follows.

$$S_o(p,q) = \sin(\frac{|\theta_p - \theta_q|}{2}). \tag{6}$$

## 5.2. Geometric consistency function

Given one pair of objects $u = (p_1, p_2)$ in the original scene, in order to compare its consistency with another pair of objects $v = (q_1, q_2)$ in the changed scene, we define a geometric consistency function $C$. The geometric consistency function is composed of two terms, pairwise orientation consistency term and pairwise distance consistency term. The function is defined as follows, and two terms are weighted using two parameters.

$$C(u,v) = \omega_{oc}C_o(u,v) + \omega_{dc}C_d(u,v). \tag{7}$$

**Pairwise orientation consistency term**. Each object is first projected onto the ground to form 2D image, and its principal component vector is computed. The angle $\phi$ between principal component vectors of a pair of objects describes the relative direction. The cosine difference of angles between one pair $u$ in $P$ and another pair $v$ in $Q$ is used to measure the orientation consistency degree $C_o(u,v)$.

**Pairwise distance consistency term**. This term describes whether the distance between one pair $u$ is consistent with that between $v$. We compute the mass center for 2D projection of each object. The distance is represented with the plane distance between their projections. The difference degree between pairwise distances $d_u$ and $d_v$ constitutes the pairwise distance consistency term $C_d(u,v)$, which is defined as

$$C_d(u,v) = \min\ \{\frac{|d_u - d_v|}{d_u}, 1\}. \tag{8}$$

To make computation robust against partial occlusion, $q$ is substituted with $p$ after transformation while computing its principal direction and center.

## 6. Experiment results

We have implemented the prototype system using Kinect on 3.4GHz 4-core Intel PC with 16G memory and GPU of Nvidia GTX 750, and evaluated the efficiency of automatic modeling update on different types of scenes, such as office and home. While capturing the local variation, RGBD camera should be directly oriented towards the changed scene so that the captured areas of objects are as large as possible. The capture distance should be appropriate. Note that dark surfaces may not reflect enough light and the specular reflection effect of mirrors or metal also causes data missing. Table 1 shows the statistics and timing of the modeling update of each scene. The second and third columns indicate the number of objects.

**Table 1:** The run time (in seconds) of our algorithm.

| Scene | Total | Moved | Labeling | Update |
|---|---|---|---|---|
| Office A | 11 | 2 | 3.7 | 21 |
| Dining Room | 5 | 1 | 3.1 | 16 |
| Office B | 9 | 2 | 3.5 | 25 |
| Living Room | 5 | 2 | 3.2 | 10 |
| Bedroom | 5 | 3 | 3.1 | 13 |

**Parameter Setting**. All the parameters in our tests, including their values and locations, are listed in Table 2. We will focus on two groups of important parameters. *The first group* includes two classification parameters used in automatic labeling (Sec. 4.2). They are random tree number and depth of random forest classifier, affecting labeling results and further object extraction. If the number of trees is set to a small value (e.g. less than 10), the classifier becomes unstable because the votes from a few trees are easily dispersed. It is gradually increased to make the classification reach a steady state. Hence the number is appropriately set to 20. In the same time, the depth of each tree is set to 5 because a deeper tree may over-fit to a small quantity of training samples adopted in our method. *The second group* is composed of six balancing weights in the optimization framework. The sum of weights in each function is naturally set to 1. Color similarity weight is used to discriminate objects in different colors, and commonly set low because we notice that the furniture at home and office are intentionally arranged in the same color to obtain a unified visual effect. Placement orientation similarity term prefers the nearest wall to determine the object orientation, and a large weight may make it hard to distinguish objects such as many desks and cabinets regularly placed along wall or parallel to wall. Therefore, the two weights are tuned to lower values, e.g., 0.1. Appearance similarity weight favors the effective comparison between different categories of objects (e.g., desks and chairs) based on human visual perception, and it is accordingly with relatively high confidence. Nevertheless, only partial appearance of scanned objects is available and its discrimination power is greatly reduced while comparing to multiple views of 3D models, especially among objects in the same category. Heavily depending on it may cause wrong match between two similar office chairs. Hence its value is set to a moderate magnitude, e.g., 0.3. Geometric shape similarity term plays two important roles, one of which is to accurately determine registration error, and the other is to produce rigid transformation used to align a local changed scene to the original scene. We set its weight to be one half of the gross weights. In geometric consistency function, orientation consistency is regarded more important than distance consistency because the orientation variation relative to local transfer is generally significant in office and home scenes. They are empirically set to 0.8 and 0.2, respectively.

**Scenes, superpixel generation, and labeling results**. Figure 6 shows superpixel results for a series of RGBD images captured in real scenes. The boundaries of superpixels are visually reasonable, which explains that contour cues

**Table 2:** Parameters used in our algorithm.

| Parameters | Values | Sec. |
|---|---|---|
| Superpixel number | 30 | 4.1 |
| Distance threshold | 0.95 | 4.1 |
| Angle interval | 1/50 | 4.1 |
| Inter-cluster error threshold | 5 | 4.1 |
| Training image number | 51 | 4.2 |
| Block number | 50 | 4.2 |
| Random tree number | 20 | 4.2 |
| Random tree depth | 5 | 4.2 |
| Overlap threshold | 50% | 4.3 |
| Appearance similarity weight | 0.3 | 5.1 |
| Color similarity weight | 0.1 | 5.1 |
| Geometric shape similarity weight | 0.5 | 5.1 |
| Placement orientation similarity weight | 0.1 | 5.1 |
| Viewpoint number | 120 | 5.1 |
| View cluster number | 10 | 5.1 |
| EMD histogram size | 10 | 5.1 |
| Key point percent | 1% | 5.1 |
| SHOT descriptor dimension | 352 | 5.1 |
| Variance of EM-ICP | 0.1 | 5.1 |
| Pairwise orientation consistency weight | 0.8 | 5.2 |
| Pairwise distance consistency weight | 0.2 | 5.2 |

play a significant role. Wall and floor are removed from depth images to show that they are recognized effectively. From the labeling results in Fig. 4, it is seen that most objects are rightly labeled while there are some misclassified parts such as leg of chair. These errors come from noise, discontinuous points, and missing depth, which can not be avoided because of the condition limitation of low-cost depth sensor. Moreover, these errors are also related to discriminative power of descriptors.

**Scene rearrangement**. Figure 1 shows that our algorithm has ability of automatically rearranging different types of scenes, which can avoid any possible conflict with original scenes and yield consistent modeling with real scenes. Moreover, our algorithm allows to deal with multiple moved objects at the same time, e.g., desk, chair, and cabinet, as illustrated in the bedroom scene (right in the second row).

**Optimization term analysis**. We adopt an office updating experiment to demonstrate the effects of different terms, illustrated in Fig. 7. The experiment is to update the upper left scene with a RGBD image of chair and safe box movement (upper middle), and the updated result is shown in the upper right of Fig. 7. The case of adopting the similarity function only is shown in (a). The position and orientation of desk is wrongly updated: it is placed outside the wall. In addition, the safe box and chair are in wrong orientations. Without geometric consistency function to take their relative relationship into consideration, such objects are wrongly placed although they rightly match models. This validates the necessity of geometric consistency function. We also try to neglect appearance similarity term. This causes a false match between safe box and bookcase (see Fig. 7(b)). Figure 7(c)

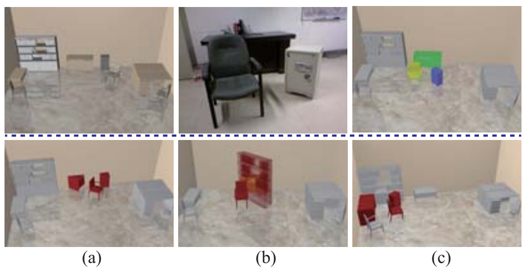

**Figure 7:** Optimization term analysis. (a) Without geometric consistency function. (b) Without appearance similarity term. (c) Without geometric shape similarity term.

presents that it is difficult to differentiate between the desk close to the safe box and another one along left wall without geometric shape similarity term, since their color, appearance, and relative orientation to wall are all similar. Moreover, neighboring chair and safe box refer to the desk, and hence they are also wrongly displaced.

**Multiple scene variations**. Our approach can accommodate continuous changes of a scene, as the living room shown in Fig. 8(a). The first movement is to change the position and orientation of the box in blue and the table in yellow. Two sofas in green are the static references in this case. The second variation of scene is based on the first one, in which the box is removed from the scene. When the local changed scene is captured, the reference objects are two sofas and one tea table in green. We show that the two variations of scene are both accurately modeled.

**Object removal**. Besides a previous example in Fig. 8(a), we supplement two extra examples about object removal as shown in Fig. 8(b). Objects in blue are removed from two original scenes while desks remain static as references. Each desk to be updated can correspond with the desk in the original scene, and its rigid transformation parameters are used to register the local changed scene into the whole original scene. Removed chairs cannot be captured so that the changed scene without the chair will substitute the local part of the original scene. This is why they disappear in two updated 3D scenes.

**Large scene variation**. In Fig. 9, we demonstrate two updating examples (a) and (b) for an office scene with 9 objects. In the case (a), 6 objects (upper right) except two desks and a chair (upper middle) are moved. We compare the updated 3D scene (middle right) to its expected result (bottom right) and find that only three static objects in green are rightly reconstructed while all other objects in red are wrongly placed. It is because that the geometric consistency function becomes invalid when it is lack of neighboring reference objects. In the case (b) that all the objects are moved, it can be seen that their positions and orientations (in red) are completely wrong after updating. Although each object

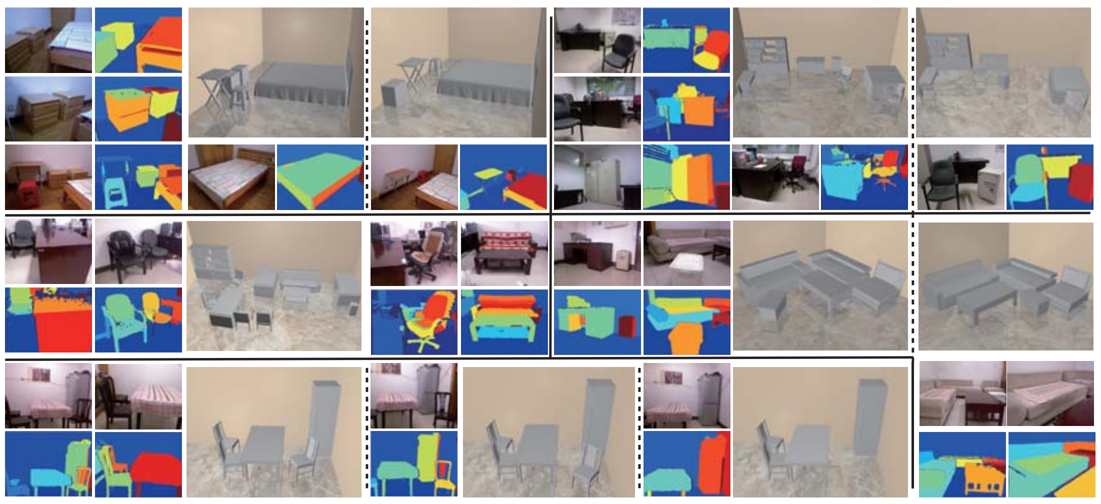

**Figure 6:** The test scenes separated by black solid lines are bedroom, office B, office A, living room, and dining room. The superpixel segmentation results neighboring to them are generated from RGBD images captured in these scenes.

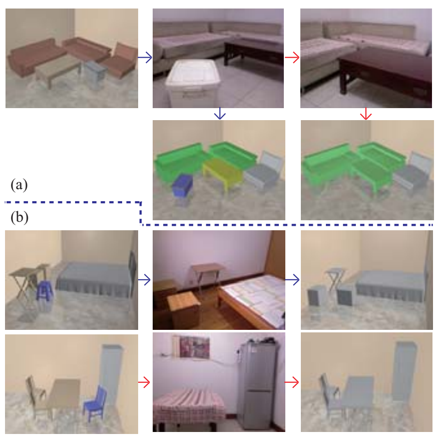

**Figure 8:** The tests on continuous changes of scene and the removal of original objects.

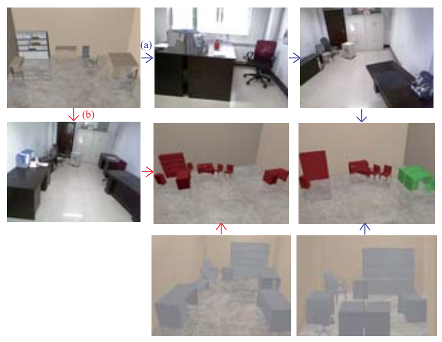

**Figure 9:** The tests on large scene variation.

can rightly match its 3D model, its new position is unknown since each object is unable to find its neighboring reference.

## 7. Conclusion

We presented an automatic approach to modeling update of indoor scenes with a consumer RGBD camera. It helps user to conveniently obtain alterable reconstruction without extra assistance, which significantly reduces modeling workload.

**Limitations**. Our method depends on a strong assumption that static objects should exist as references to moved objects when taking a local changed scene. If all the objects move, the optimization will not converge. Moreover, we observed two main failure modes:

*Cluttered scene update*. We perform an experiment to validate the update of a cluttered scene with smaller objects, as illustrated in Fig. 10. The original scene (upper left in (a)) is clean whereas in the changed scene (bottom left), many toys, tools, cups, bottles, food, books, and so on, are deposited on the sofa and table. The table moves, and the cabinet is placed behind table to form occlusion. Labeling results (b) show that the original image (upper middle) is rightly labeled while there are large amounts of errors in the image (bottom middle) of changed scene. In the updated scene (c), the cabinet is mostly occluded by table and becomes completely unrecognizable, which makes it disappear (bottom right). The left sofa is partitioned into two parts by various small objects, and in the matching stage each of the two parts matches one sofa model. As a result, the two sofa models overlap. The middle sofa is labeled as table and substituted with a table model. The right sofa is removed from the scene since its data is mostly missing in the presence of large occlusion.

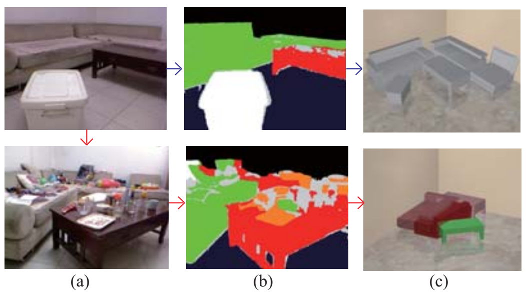

**Figure 10:** Cluttered scene update.

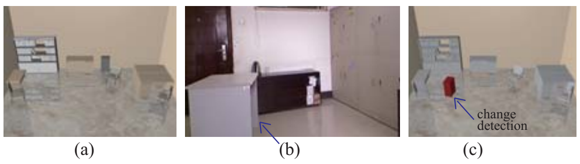

**Figure 11:** New object insertion and change detection.

*New object insertion*. Unlike recent methods such as [KMYG12], that accommodate new object insertion, our method fails to update scenes when new objects are inserted, as shown in Figure 11. While updating the scene (a) after a new desk is inserted (b), it is mistakenly substituted with a cabinet in red (c). Our method does not support new object insertion currently because our training samples are only composed of existent models in the original scene.

**Applications**. Our technique can be employed to assist in 3D indoor scene modeling. Compared to previous solutions to scene reconstruction with consumer RGBD camera, which produces a static scene in 3D, our method avoids reconstructing the whole 3D scene from scratch when the scene partially changes. The full 3D scene can be automatically and quickly generated again by only taking one RGBD image of local variation. These scenes can also be provided for not only digitalizing different schemes while rearranging furniture in interior designs but also building frequently changed indoor 3D map beforehand for robots.

**Acknowledgements**. The work is partially supported by NSFC (No. 61003137, 61202185, 61272392, 61322204, 61005018, and 91120005), NWPU Basic Research Fund 310201401-(JCQ01009, JCQ01012), and Fund of National Engineering Center for Commercial Aircraft Manufacturing.

## References

[BBGO11] Bronstein A. M., Bronstein M. M., Guibas L. J., Ovsjanikov M.: Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph. 30*, 1 (Feb. 2011), 1:1–1:20. 3

[BLHW13] Bu S., Liu Z., Han J., Wu J.: Superpixel segmentation based structural scene recognition. In *ACM International Conference on Multimedia* (2013), pp. 681–684. 4

[BWdBP13] Berretti S., Werghi N., del Bimbo A., Pala P.: Matching 3D face scans using interest points and local histogram descriptors. *Computers and Graphics 37*, 5 (2013). 6

[ERB*12] Eitz M., Richter R., Boubekeur T., Hildebrand K., Alexa M.: Sketch-based shape retrieval. *ACM Trans. Graph. 31*, 4 (July 2012), 31:1–31:10. 3

[GAM13] Gupta S., Arbelaez P., Malik J.: Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR* (June 2013), pp. 564–571. 3, 4

[GP02] Granger S., Pennec X.: Multi-scale EM-ICP: A fast and robust approach for surface registration. In *European Conference on Computer Vision* (2002), pp. 418–432. 6

[KAJS11] Koppula H. S., Anand A., Joachims T., Saxena A.: Semantic labeling of 3D point clouds for indoor scenes. In *Neural Information Processing Systems* (2011). 3

[KMYG12] Kim Y. M., Mitra N. J., Yan D.-M., Guibas L.: Acquiring 3D indoor environments with variability and repetition. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 138:1–138:11. 3, 10

[LBZ*13] Liu Z., Bu S., Zhou K., Gao S., Han J., Wu J.: A survey on partial retrieval of 3D shapes. *Journal of Computer Science and Technology 28*, 5 (2013), 836–851. 3

[LH09] Leordeanu M., Hebert M.: An integer projected fixed point method for graph matching and MAP inference. In *Neural Information Processing Systems* (2009). 5

[MSL*11] Merrell P., Schkufza E., Li Z., Agrawala M., Koltun V.: Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30*, 4 (July 2011), 87:1–87:10. 3

[NXS12] Nan L., Xie K., Sharf A.: A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 137:1–137:10. 1, 3

[SF11] Silberman N., Fergus R.: Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops)* (2011), pp. 601–608. 2

[SFCH12] Shen C.-H., Fu H., Chen K., Hu S.-M.: Structure recovery by part assembly. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 180:1–180:11. 3

[SWK07] Schnabel R., Wahl R., Klein R.: Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum 26*, 2 (2007), 214–226. 5

[SXZ*12] Shao T., Xu W., Zhou K., Wang J., Li D., Guo B.: An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 136:1–136:11. 1, 2, 3

[TAR*10] Tamaki T., Abe M., Raytchev B., Kaneda K., Slomp M.: CUDA-based implementations of softassign and EM-ICP. In *CVPR* (June 2010). 6

[TSDS10] Tombari F., Salti S., Di Stefano L.: Unique signatures of histograms for local surface description. In *European Conference on Computer Vision* (2010), pp. 356–369. 6

[XCF*13] Xu K., Chen K., Fu H., Sun W.-L., Hu S.-M.: Sketch2scene: Sketch-based co-retrieval and co-placement of 3D models. *ACM Trans. Graph. 32*, 4 (July 2013), 123:1–123:15. 3

[YYT*11] Yu L.-F., Yeung S.-K., Tang C.-K., Terzopoulos D., Chan T. F., Osher S. J.: Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph. 30*, 4 (July 2011), 86:1–86:12. 3

[ZSCO*08] Zhang H., Sheffer A., Cohen-Or D., Zhou Q., van Kaick O., Tagliasacchi A.: Deformation-driven shape correspondence. In *Symposium on Geometry Processing* (2008), pp. 1431–1439. 6