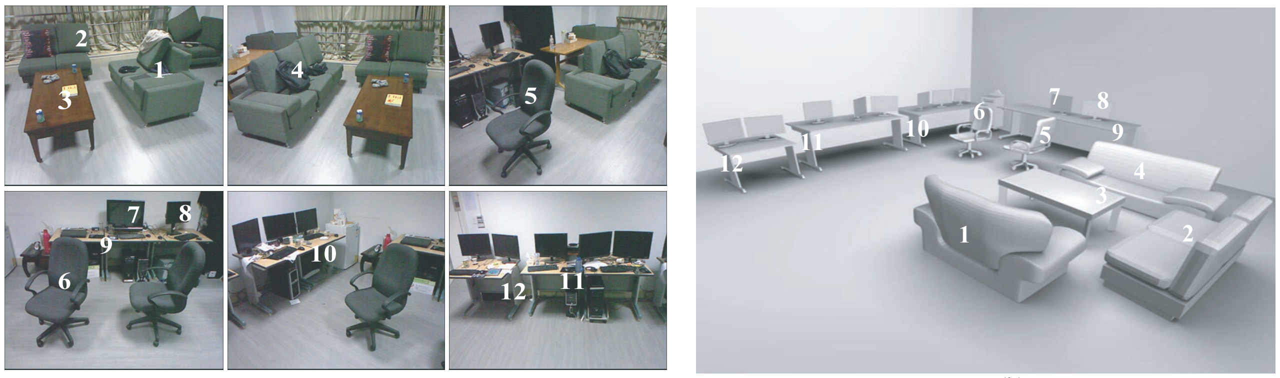
An Interactive Approach to Semantic Modeling of Indoor Scenes with an RGBD Camera

Tianjia Shao * Weiwei Xu † § Kun Zhou ‡ Jingdong Wang § Dongping Li‡ Baining Guo §
Tsinghua University * Hangzhou Normal University † Zhejiang University‡ Microsoft Research Asia§



(a) (b)

Figure 1: *Lab scene. (a) Captured images (Only RGB data are shown). (b) Reconstruction result. Only 6 RGBD images captured by Microsoft Kinect camera are enough for our system to reconstruct its prototype scene with 20 objects of semantic labels (eight monitors are not numbered to avoid possible clutter). The numbers in white indicate the correspondence between objects in the image and their reconstruction results and the overall modeling time of this lab scene is less than 18 minutes.*

Abstract

We present an interactive approach to semantic modeling of indoor scenes with a consumer-level RGBD camera. Using our approach, the user first takes an RGBD image of an indoor scene, which is automatically segmented into a set of regions with semantic labels. If the segmentation is not satisfactory, the user can draw some strokes to guide the algorithm to achieve better results. After the segmentation is finished, the depth data of each semantic region is used to retrieve a matching 3D model from a database. Each model is then transformed according to the image depth to yield the scene. For large scenes where a single image can only cover one part of the scene, the user can take multiple images to construct other parts of the scene. The 3D models built for all images are then transformed and unified into a complete scene. We demonstrate the efficiency and robustness of our approach by modeling several real-world scenes.

Keywords: Indoor scene, Depth images, Segmentation, Labeling, Random Regress Forest

Links: DL PDF

[†]email:weiwei.xu.g@gmail.com, corresponding author

[‡]Kun Zhou and Dongping Li are with the State Key Lab of CAD&CG of Zhejiang university.

1 Introduction

With the popularization of commercial RGBD cameras such as Microsoft's Kinect, now everyone can have a low-cost and easy-to-use device to digitalize 3D objects at home [Clark 2011]. It is naturally of great interest to use RGBD cameras to scan and build 3D scenes which closely match the indoor environments where people live or work. Such 3D scenes can be used in applications for many purposes, such as providing an interface for user interaction [Izadi et al. 2011] and rearranging furniture for interior design [Yu et al. 2011; Merrell et al. 2011].

Several techniques have been proposed recently for modeling indoor environments with a depth camera, representing the scene geometry either as point clouds [Du et al. 2011] or signed distance fields defined over a 3D volume grid [Izadi et al. 2011]. While such geometry representations meet application requirements to a certain extent, they do not provide the necessary flexibility required by many other applications due to the lack of semantics. For example, in the context of rearranging furniture, the input scene to state-of-the-art algorithms consists of a set of independent, semantic objects, i.e., each scene object can be manipulated independently and belongs to a category (e.g., sofa, chair, bed, etc.). The algorithms need such semantics to calculate an optimal layout for all objects and transform each object to its target position. As far as we know, none of the existing modeling techniques aim to generate 3D semantic models for indoor environments.

In this paper we present an interactive approach to semantic modeling of indoor scenes with a consumer-level RGBD camera. Using our approach, the user first takes an RGBD image of an indoor scene, which is automatically segmented into a set of regions with semantic labels. If the segmentation is not satisfactory, the user can draw some strokes to guide the algorithm to achieve better results. After the segmentation is finished, the depth data of each semantic region is used to retrieve a matching 3D model from a database. Each model is then transformed according to the image depth to yield the scene. In cases where the scene is too large and a single image can only cover one part of the scene, the user can take multiple images to construct other parts of the scene. To this end, we require that two images taken successively have sufficient overlap for feature matching, which ensures that the camera transformation between the image pair can be computed. The 3D models built for all images are then transformed and unified into the complete scene. Fig. 2 illustrates the pipeline of our approach.

The efficiency and robustness of our approach are warranted by three design choices. First, instead of asking the user to acquire a RGBD video of the scene as in the Kinect Fusion or other interactive indoor scene modeling systems [Izadi et al. 2011; Du et al. 2011], we let the user capture only a sparse set of RGBD images to reconstruct a medium-scale indoor scene, resulting in less user interaction as well as reduced storage and computational cost. The requirement of sufficient overlap between successively captured images is similar to that in image stitching algorithms for panorama construction, which is very easy to satisfy in practice. Second, when segmenting an RGBD image into semantic regions, we allow the user to improve the result generated by the automatic algorithm through a stroke-based interface. According to our experiments, very simple user interactions can significantly improve the accuracy of segmentation, which is difficult to achieve by automatic algorithms. Moreover, the improved accuracy for the current image can be propagated to successive images automatically by integrating the segmentation information into the learned discriminative model. Finally, instead of reconstructing the precise scene geometry, we search a 3D model database to find models that best approximate the scene geometry. This avoids the difficulty of acquiring complete and precise depth data due to severe occlusions in indoor scenes. Although our modeling result is not exactly the same geometry as the indoor scene, it contains most useful information required by scene editing and other high-level applications, i.e., the semantic objects and their layout in the scene.

We have implemented the whole modeling pipeline in a prototyping system and demonstrated its usefulness with several real-world scenes. Besides the general approach, the paper also makes two technical contributions:

- An interactive context-aware image segmentation and labeling algorithm to extract semantic regions from RGBD images. Our image labeling algorithm first applies the automatic indoor scene segmentation algorithm in [Silberman and Fergus 2011] to get an initial image segmentation result, and dynamically adapts its learned discriminative appearance and geometry models to reflect the context of the current scene to improve labeling accuracy. User interaction is minimized since the user only needs to edit the segmentation result when automatic segmentation is not satisfactory.

- A random regression forest based algorithm for matching the depth data of the segmented regions to the models in our database. It is achieved by learning a mapping from patches in the depth image to the model instance labels, i.e., the indices of models in the database. We found that patch-level regression is robust to noise and can handle partial data that frequently occur in the captured depth images well. In addition, we also learn a mapping from patches to the object orientation in the image to facilitate the subsequent model placement in constructing the indoor scene.

In the rest of the paper, we first briefly review related work in Sec. 2. In Sec. 3, we introduce the details of the image segmentation and labeling algorithm. Sec. 4 discusses the data-driven construction of indoor scenes from segmented images. Experimental results and statistics are described in Sec. 5, and the paper concludes with discussions about future work in Sec. 7.

2 Related Work

Indoor scene image segmentation and labeling. Many research works have been devoted to segmentation and labeling of indoor scene images. Automatic indoor scene labeling algorithms usually learn a conditional random field (CRF) model from a large amount of labeled training data and then optimize the learned CRF model for scene labeling [Xiong and Huber 2010; Anand et al. 2011; Silberman and Fergus 2011; Koppula et al. 2011]. Silberman et al. [2011] learned the data term and pairwise term in the CRF model separately. They achieved around 60% labeling accuracy for a wide range of object class labels. A recent contribution in [Silberman et al. 2012] proposed to infer the structure classes and supporting relationship for indoor scene simultaneously to improves the understanding of physical interaction between objects. Kopppula et al. [2011] achieved high labeling accuracy, around 80%, through mixed integer optimization. However, the optimization takes a long time and thus is not suitable for interactive applications. Although the learned model can capture the distinctive features of objects and their contextual relationship through training data, its generalization capability is still not good enough to achieve a perfect segmentation and labeling result for a new image. It is also possible to apply some interactive tools to extract one object at a time from a single image [Li et al. 2004]. However, the interaction efforts could be very large due to the large scale of indoor scenes, as the user needs to cutout each object in the scene.

Object detection is an alternative way to extract objects from indoor scene images and object class classifiers need to be trained from the labeled dataset to fulfill the detection task. Janoch et al. [2011] built a category-level 3D object dataset using Microsoft Kinect camera and conducted experiments on sliding-window based 3D object detection in depth data. To reduce the workload in building large size annotated dataset, Lai et al. [2010] proposed to train the classifier using the annotated 3D data available in the web and adapted it to the captured 3D data. However, similar to automatic image segmentation algorithms, their recognition accuracy and generalization power now is not good enough to high-quality indoor scene modeling.

Our image segmentation algorithm combines the best part of automatic and interactive image segmentation approaches. The user interaction is only necessary where the segmentation result from automatic RGBD image segmentation result is not satisfactory. We adopt the automatic image segmentation algorithm in [Silberman and Fergus 2011] to train our CRF model since the generated probabilistic discriminative model for data term can be adapted according to the statistical information from the current indoor scene.

Indoor scene modeling. Indoor scene modeling has attracted a lot of research interest from both the computer graphics and vision communities. Early solutions mount a laser scanner on a mobile robot to scan a sequence of range images and then register the captured data using ICP (iterative closest point) or SLAM (simultaneous localization and mapping) techniques [Fox et al. 1999; Whitaker et al. 1999]. Due to the dependency on expensive hardware, these solutions are not accessible to average users.

Image-based modeling techniques have also been applied to indoor scene modeling. Given a sequence of 2D images of a scene, structure-from-motion, such as bundle adjustment, and multiview geometry techniques can be applied to reconstruct 3D structures [Triggs et al. 2000; Hartley and Zisserman 2004; Snavely et al. 2006]. However, in image-based modeling, the detection of 2D interest points in the input 2D images significantly influences the density of the reconstructed 3D points. Even though the patch-based

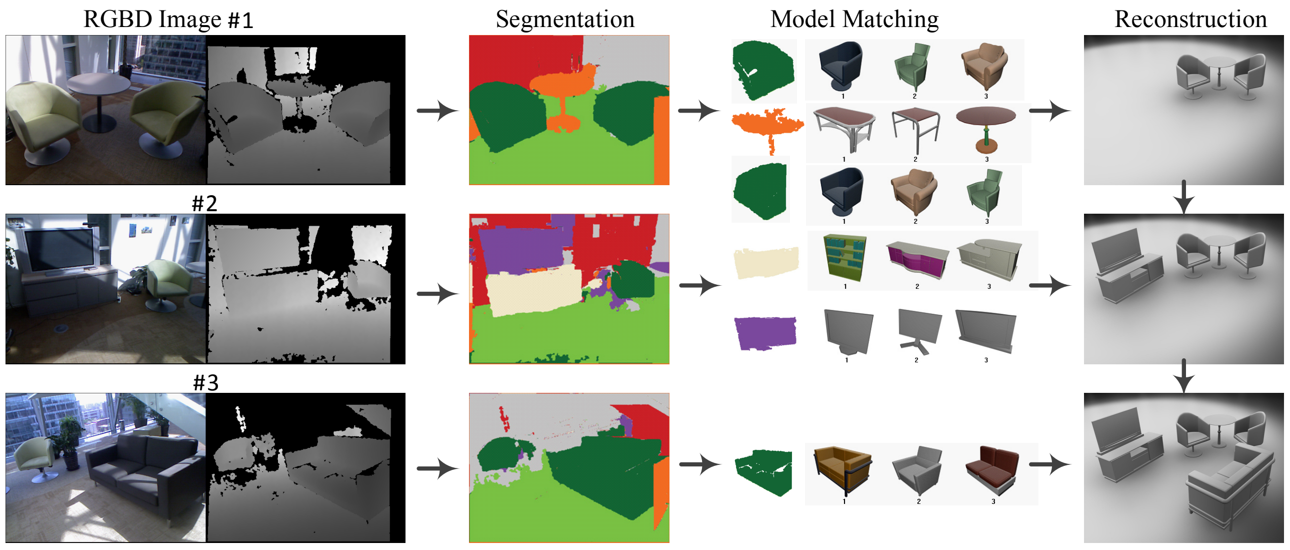


Figure 2: *Semantic modeling system pipeline. The input images are first segmented into semantical regions, and each segmented out region is replaced by its similar 3D models in our database. The system progressively reconstructs and registers the whole scene using consecutively captured RGBD-images.*

framework in [Furukawa and Ponce 2010] is able to reconstruct a dense 3D points, it is still difficult to apply it for indoor scene modeling to generate dense depth maps since there exist a large amount of textureless areas (e.g., walls) in indoor scenes. Based on the Manhattan world assumption, Furukawa et al. [2009a] reconstruct depth maps from 2D images of indoor scenes. Their method is limited to axis-aligned planes. Furukawa et al. [2009b] also propose an image-based rendering method for indoor scenes by building axis-aligned plane geometry proxies.

With the fast development of consumer-level depth cameras, it is possible to capture dense depth maps of an indoor scene. Research works on 3D modeling with consumer-level depth cameras focus on the fusion of depth maps or the combination of color and depth information. A prominent example is the Kinect Fusion system [Izadi et al. 2011]. It uses a volumetric representation to fuse the depth map. Henry et al. [2012] explore the registration of depth maps using both color and depth information. To solve the technical challenges in automatic depth map fusion algorithms, Du et al. [2011] propose to integrate on-line user interaction in indoor scene modeling.

Our work aims to reconstruct not only the geometry information of an indoor scene but also its semantic representation from sparsely captured depth images. The reconstruction result of our algorithm consists of semantic geometry objects, such as chair, sofa, walls and so on. It facilitates the usage of reconstructed indoor scene models in high-level applications, such as furniture layout and social gaming. Two concurrent efforts have also been made to solve this challenging indoor scene modeling problem [Nan et al. 2012; Kim et al. 2012].

3D shape matching. Much research has been conducted on finding models similar to an input 3D model in a database [Funkhouser et al. 2003; Tangelder and Veltkamp 2008; Bronstein et al. 2011]. Unfortunately, these methods cannot be directly applied in our setting, where only a single view of 3D models is present in the captured RGBD image. Therefore, we pose the problem of retrieving similar models of a segmented object in the image as an object instance recognition problem. Spin images [Johnson and Hebert 1999] has been widely used in 3D shape recognition problems [Golovinskiy et al. 2009]. Instead of using hand-designed features, Bo et al. [2011] proposed depth kernel descriptors for object recognition, and Blum et al. [2011] proposed unsupervised feature extraction based on k-means clustering for object recognition on the RGBD object dataset in [Lai et al. 2011].

Our problem is unique: we do not require a one-to-one mapping between the 3D models in the database and the segmented objects. Since the texture information in our model database usually does not match the texture in the captured RGBD images, texture based features are not used in our application. We thus use the random regression forest in [Fanelli et al. 2011] since it can achieve high recognition accuracy only using depth information. We extend the algorithm to multi-instance object recognition for indoor scene modeling.

3 Interactive Context-aware Image Segmentation and Labeling

Given RGBD indoor images captured by the user, our goal is to segment the images into regions with a semantic object label, such as floor, chair and monitor. To this end, we develop an interactive context-aware image segmentation algorithm to fulfill this task. It has two important features: (a) It integrates the benefits of both automatic and interactive approaches to achieve a high-quality segmentation result. User interaction, which is implemented via a stroke-based interface, is minimized since it is necessary only when the automatic segmentation result is not satisfactory. (b) It reflects the context of the current scene according to the segmentation result through dynamically updating the appearance and geometry model learned from the NYU indoor scene image database [Silberman and Fergus 2011]. Precisely, the statistical information from the segmentation result is integrated to make the classifier adapt to the specific scene and thus improve labeling accuracy.

In image segmentation, each pixel is assigned a semantic label. The ten class labels we support for the purpose of indoor scene modeling are sofa, table, monitor, wall, chair, floor, bed, cabinet, ceiling and background. They are all common objects in indoor scenes, except that background represents unknown or meaningless object in the scene. We use the Conditional Random Field (CRF) model to solve the labeling problem. The CRF energy function for a label $\mathbf{c}$ is the following:

$$\mathbf{E}(\mathbf{c}) = \sum_i \mathbf{E}_1(\mathbf{c}_i : \mathrm{x}_i) + \lambda \sum_{i,j} \mathbf{E}_2(\mathbf{c}_i, \mathbf{c}_j), \tag{1}$$

where the data term $E_1(\mathbf{c}_i : \mathrm{x}_i)$ measures the likelihood of label $\mathbf{c}_i$ for pixel i conditioned on its feature x_i, and the compatibility term $E_2(\mathbf{c}_i, \mathbf{c}_j)$ measures the consistency between labels for two neighboring pixels. This objective function can be efficiently minimized using the graph-cuts of Boykov et al. [2001].

3.1 Data Term

The data term $E_1(\mathbf{c}_i : \mathrm{x}_i)$ evaluates the likelihood of an object label $\mathbf{c}_i$ according to the feature x_i at pixel i. It is a sum of two terms on local appearance and geometry model to exploit the color and depth information from the depth camera:

$$\mathbf{E}_1(\mathbf{c}_i : \mathrm{x}_i) = \mathbf{E}_a(\mathbf{c}_i : \mathrm{x}_i^a) + \mathbf{E}_g(\mathbf{c}_i : \mathrm{x}_i^g), \tag{2}$$

where $\mathbf{E}_a(\mathbf{c}_i, \mathrm{x})$ denotes the appearance term, and $\mathbf{E}_g(\mathbf{c}_i, \mathrm{x})$ the geometry term. x_i^a represents the local appearance feature computed using the local color information, and x_i^g the local geometry feature.

Appearance term: The appearance term $\mathbf{E}_a(\mathbf{c}_i : \mathrm{x}_i^a)$ is the following:

$$\mathbf{E}_a(\mathbf{c}_i : \mathrm{x}_i^a) = -\log((1-\alpha_p)P_t(\mathbf{c}_i|\mathrm{x}_i^a) + \alpha_p P_c(\mathbf{c}_i|\mathrm{x}_i^a)). \tag{3}$$

It is a blending of two learned discriminative models: $P_t(\mathbf{c}_i|\mathrm{x}_i^a)$ from the indoor scene image database and $P_c(\mathbf{c}_i|\mathrm{x}_i^a)$ from the segmentation results. The weight α_p is 0.6 in our implementation to favor the appearance model learned from the segmentation results. The addition operation to combine the appearance features is simple, and it proved to be very efficient in our experiments.

We follow [Silberman and Fergus 2011] to learn $P_t(\mathbf{c}_i|\mathrm{x}_i^a)$ using the NYU indoor scene image database captured by a Microsoft Kinect camera. Specifically, we extract the SIFT feature for each pixel from the RGB images and the depth images in the database and concatenate them to form RGBD SIFT features. Supervised learning is then performed using a two layer neural network with cross-entropy loss function to compute $P_t(\mathbf{c}_i|\mathrm{x}_i^a)$. We refer readers to [Silberman and Fergus 2011] for details.

Given the user-edited segmentation results, an appearance model can be learned to reflect the current scene we are modeling. To this end, we learn a discriminative model $P_c(\mathbf{c}_i|\mathrm{x}_i^a)$ from the segmentation results by using k-means which is shown to be very efficient and effective [Li et al. 2004]. Precisely, we aggregate the pixels in the segmentation results for each object class $\mathbf{c}_i$ and then perform K-means on their RGB values. 32 clusters are extracted for each object class in our current implementation. All the cluster centers of each object class represent its appearance information. We thus compute $P_c(\mathbf{c}_i|\mathrm{x}_i^a)$ as follows:

$$P_c(\mathbf{c}_i|\mathrm{x}_i^a) = \frac{1/(-d(\mathrm{x}_i^a, \mathbf{c}_i) + \xi)}{\sum_j 1/(-d(\mathrm{x}_i^a, \mathbf{c}_j) + \xi)}, \tag{4}$$

where $d(\mathrm{x}, \mathbf{c}_i)$ is the distance between the RGB of the current pixel and the closest cluster center for object class $\mathbf{c}_i$, and ξ is a small number, 10^{-6}, to avoid the dividing by zero issue.

Geometry term: The geometry term $\mathbf{E}_g(\mathbf{c}_i : \mathrm{x}_i^g)$ is designed to be similar to the appearance term:

$$\mathbf{E}_g(\mathbf{c}_i : \mathrm{x}_i^g) = -\log((1-\alpha_g)P_t(\mathbf{c}_i|\mathrm{x}_i^g) + \alpha_g P_c(\mathbf{c}_i|\mathrm{x}_i^g)). \tag{5}$$

It also consists of two discriminative geometry models: $P_t(\mathbf{c}_i|\mathrm{x}_i^g)$ is a geometry model learned from depth images in the NYU indoor scene image database [Silberman and Fergus 2011], and $P_t(\mathbf{c}_i|\mathrm{x}_i^g)$ is from the previous segmentation results. α_g is also set to be 0.7.

Rather than using local depth data, the local geometry feature x_i^g is based on the extracted plane primitives, since they are more resilient to the noise in depth data and result in more accurate descriptors. The plane primitives are extracted using the efficient RANSAC algorithm in [Schnabel et al. 2007]. For a pixel i in the image, we first determine which plane primitive covers it, and then compute three types of geometry features as x_i^g:

- Height h_i: The distance of the projection of the pixel i on its plane primitive to the ground.
- Size s_i: The size of the plane primitive that covers the pixel.
- Orientation θ_i: The angle between the normal of the primitive and the ground normal.

The ground plane in RGBD images can be easily determined, since the Microsoft Kinect camera has an accelerometer installed. Precisely, we can easily compute the roll, pitch and yaw angle of the camera using the data from the accelerometer, and then rectify the camera pose to be parallel to the ground. The ground is just the plane with lowest height in the rectified depth image.

$P_t(\mathbf{c}_i|x_i^g)$ is actually represented by histograms on each geometry feature. To this end, we first discretize each feature into a set of discrete values, e.g., $\{h_i\}_{i=1}^{n_h}$ for the height. Then, we build a histogram for each discrete value, $P_t(\mathbf{c}_i|h_i)$, to describe the posterior probability that a pixel with height h_i belongs to object $\mathbf{c}_i$. Similarly, we build the histograms for the other two types of features. Next, we compute the overall posterior probability as $P_t(\mathbf{c}_i|\mathrm{x}_i^g) = P_t(\mathbf{c}_i|h_i)P_t(\mathbf{c}_i|s_i)P_t(\mathbf{c}_i|\theta_i)$.

$P_c(\mathbf{c}_i|\mathrm{x}_i^g)$ is built with the same procedure. However, they are built on the segmented images to reflect the status of the current scene. In reality, we found that the learnt posterior probability (histogram) from the current scene is very sparse because the variance of the geometric features for an object instance is small.

Model updating: During the segmentation of captured images, the appearance and geometry model learned from the current scene, $P_c(\mathbf{c}_i|\mathrm{x}_i^a)$ and $P_c(\mathbf{c}_i|\mathrm{x}_i^g)$, are added into the data term to adapt the initial models learned from the training database to the current scene. These two models are also continuously calculated as more segmentation results are computed. In the case when there are only a subset of object classes present in the current segmentation result, we only update their conditional probability and keep the rest the same. For the first captured image, only the discriminative models learned from indoor scene database are applied to achieve the initial segmentation result. Usually, it needs the most interaction in our experiments since there is no information about the current scene at that stage. The amount of editing on other images is usually small. It depends on the scale of the indoor scene and is no more than 30 strokes for the user to draw strokes to correct segmentation result for medium scale indoor scene. The user can edit the segmentation result anytime he/she thinks the automatic segmentation result is not satisfactory.

3.2 Compatibility Term

The compatibility term $\mathbf{E}_2(\mathbf{c}_i, \mathbf{c}_j)$ aims to impose a smoothness of the labels for neighboring pixels. We compute this term as

$$\mathbf{E}_2(\mathbf{c}_i, \mathbf{c}_j) = \delta[\mathbf{c}_i \neq \mathbf{c}_j]\,\mathrm{sim}(\mathbf{f}_i, \mathbf{f}_j). \tag{6}$$

Here, $\mathbf{f}_i = [r, g, b, d]^T$ is the concatenation of the RGB values and depth value at pixel i. The similarity between two pixels is computed by $\mathrm{sim}(\mathbf{f}_i, \mathbf{f}_j) = \exp(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2\sigma^2})$, where σ is the average distance between the features.

4 Data-Driven Construction of Indoor scenes

One option for semantic modeling is to reconstruct 3D models for each segmented object. However, it requires high quality depth images that cover the 3D objects, which is very difficult to acquire due to the severe occlusion in indoor scenes. We thus propose to populate the image with 3D models in the database at appropriate places. This data-driven procedure can automatically build a semantic 3D indoor scene since the objects in the scene are constructed separately and associated with the high level information from segmentation. The data-driven scene construction consists of four sub-steps: object extraction from segmentation, matching with random regression forest, 3D model placement and scene registration.

4.1 Matching with Random Regression Forest

The most important step in the flowchart is to match the segmented object to 3D models in the database to locate a similar 3D model for scene construction. A brute force approach is to perform pixel-level comparison between the depth data of the object and the rendered depth images of 3D models in the database. However, this approach is of huge computational cost and susceptible to noise in the captured depth image. Moreover, it cannot well handle the partial depth data resulting from occlusions in the indoor scene.

We pose the model matching problem as a model instance recognition problem. A random forest is adopted to solve this problem since it is fast in both training and testing even for large scale data. It avoids the over-fitting problem of a single decision tree and has high generalization power. Precisely, for each segmented object, we use the discriminative random regression forest in [Fanelli et al. 2011] to learn a mapping from the sampled patches to the conditional probability $p(\mathbf{m}|\hat{P})$, where $\mathbf{m}$ is the model instance label, i.e. its index in the database, and $\hat{P}$ denotes the patch drawn from the depth data of the object. The models with highest probability are deemed to be similar to the segmented object. We also learn the mapping from the depth patches to the orientation and translation of the segmented objects to facilitate 3D model placement. Since the mapping is built on the sampled patches, the trained random regression forest can well handle partial data that frequently occurs in the captured depth images.

Two modifications of the algorithm in [Fanelli et al. 2011] are made so that it can be applied in our system: a) The two-class classification in [Fanelli et al. 2011] is extended to multi-class classification. The model instance label is treated as a discrete random variable, and the calculation of its entropy for tree construction and of its probability distribution at the testing stage are modified. We use a vector to represent its probability distribution in implementation. b) Various geometry features are adopted, such as geometry moments at sub-patch level, spin images, to improve the recognition accuracy. The random regression forest algorithm itself is in essence similar.

Note that we only need to separately train random regression forests for each class supported in image segmentation. The model matching algorithm can automatically determine which forest to use according to the labeling information from segmentation.

Training: The random regression forest is trained from the rendered depth images of 3D models in the database annotated with model class labels, orientation angles and its distance to the virtual camera. Each tree T in the forest $\hat{T} = \{T_t\}$ is constructed from a set of patches $\left\{\hat{P}_i = (\mathbf{I}_i, \boldsymbol{\theta}_i)\right\}$ randomly sampled from the training depth images. $\mathbf{I}_i$ is the extracted feature computed from the pixels in a patch, and $\boldsymbol{\theta}_i = \{\theta_{yaw}, \theta_{pitch}, \theta_{roll}, t, \mathbf{m}_i\}$ is the parameters associated to patch $\hat{P}_i$, where $\{\theta_{yaw}, \theta_{pitch}, \theta_{roll}\}$ are the orientation angles and t is the distance between the model center and the camera. $\mathbf{m}_i$ is the model index associated to the patch.

The feature $\mathbf{I}_i$ is critical to the success of model matching. However, we do not know which feature will be useful for recognition in advance. As a result, we compute several informative geometry features to form a 496 dimensional feature vectors, and let the random regression forest algorithm perform feature selection at each node via maximizing information gain. The feature $\mathbf{I}_i$ consists of the following entries: depth difference, normal structure tensor, geometry moment and spin image. The details of feature computation are in the appendix.

We build the trees using the random split selection strategy [Dietterich 2000; Breiman 2001]. At each non-leaf node, a large number of binary tests are performed on a randomly selected feature channel of $\mathbf{I}_i$. It is defined as the following:

$$\mathbf{I}_i^f > \tau \tag{7}$$

where $\mathbf{I}_i^f$ indicates the value of a selected feature channel f, and τ is a randomly generated threshold. After the test, the training data are split into two sets: the patches satisfying the test are passed to the right child, and the remaining patches that failed the test are passed to the left child.

The binary test that can maximize the information gain IG is selected as the final test. To compute the information gain, we model the vectors $\boldsymbol{\theta}$ of the patches at each node as instances drawn from a probability distribution $p(\boldsymbol{\theta}) = p(\mathbf{a}) * p(\mathbf{m})$, where $\mathbf{a} = \{\theta_{yaw}, \theta_{pitch}, \theta_{roll}, t\}$. $p(\mathbf{a}) = \mathcal{N}(\mathbf{a}; \bar{\mathbf{a}}, \sum_a)$ is a multivariate Gaussian distribution, and its mean and variance are estimated from the vectors $\boldsymbol{\theta}$ stored at the current node by assuming that the variables in $\mathbf{a}$ are independent. $p(\mathbf{m})$ is computed as the percentage of each model class label in the node. The information gain is just the difference between the entropy of the patches at the parent node and the sum of entropies at its two children nodes L and R:

$$IG = H(\mathbf{a}) - \sum_{i \in \{L,R\}} w_i H(\mathbf{a}_i) + \alpha(H(\mathbf{m}) - \sum_{i \in \{L,R\}} w_i H(\mathbf{m}_i)), \tag{8}$$

where $H(\mathbf{a})$ is the entropy of the orientation angles and scale, and $H(\mathbf{m})$ the entropy of model class labels. As in [Fanelli et al. 2011], the weight α is set to be $1.0 - \exp^{-\frac{d}{\lambda}}$, where d is the depth of the tree and λ is set to be 10. In this way, we favor classification in the construction of tree at top levels, and the regression to other parameters gains more weight while we descend toward the leaves. A leaf l is constructed when the tree reaches the maximum depth or a minimum number of patches, 20 in our implementation, are left. The probability distribution of $p(\boldsymbol{\theta})$ and $p(\mathbf{m})$ for the patches arriving at the leaf node are calculated and stored for testing by random forest.

Testing: In the testing, we pass the densely sampled patches from the depth data of a segmented object to each tree in the forest. The patches flow down the tree according to the binary test stored at each node until they reach a leaf node. The probability distribution at leaf node l is then used to estimate parameters of the patches.

During the testing, all sampled patches cast a large number of votes in the parameter space. We thus need to aggregate them to get the

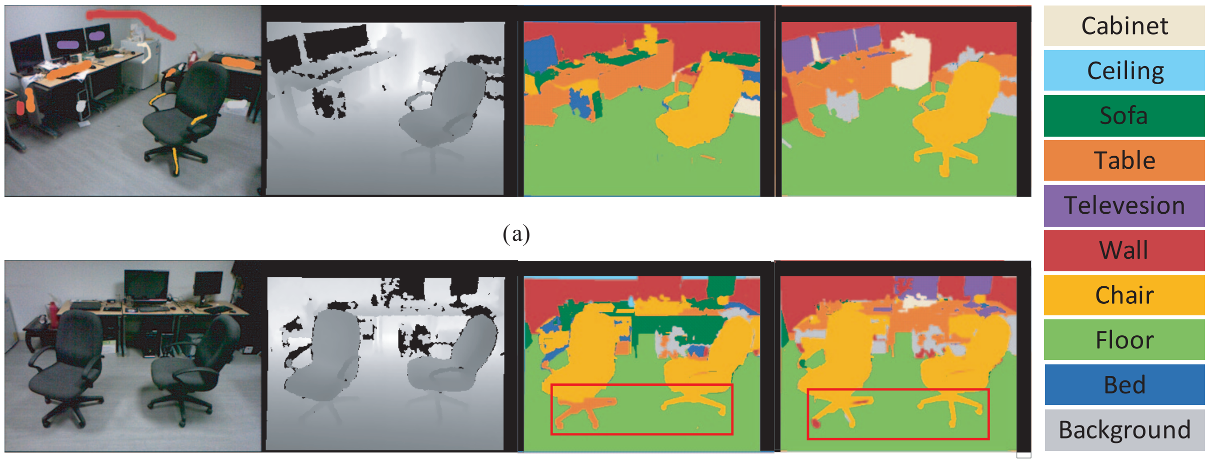


Figure 3: *Model updating in segmentation. (a) Left: RGB and depth images with user strokes. Middle: Segmentation result of the automatic image segmentation algorithm in [Silberman and Fergus 2011]. Right: Segmentation result updated according to the strokes. (b) Left: Second captured RGB and depth images from the same scene. Middle: Segmentation result of the same automatic image segmentation algorithm. Right: Segmentation result using updated appearance and geometry model according to the segmentation result in (a). Note the improvements on the segmentation of chair, monitor and table. Pixels without depth information are in black in the depth images.*

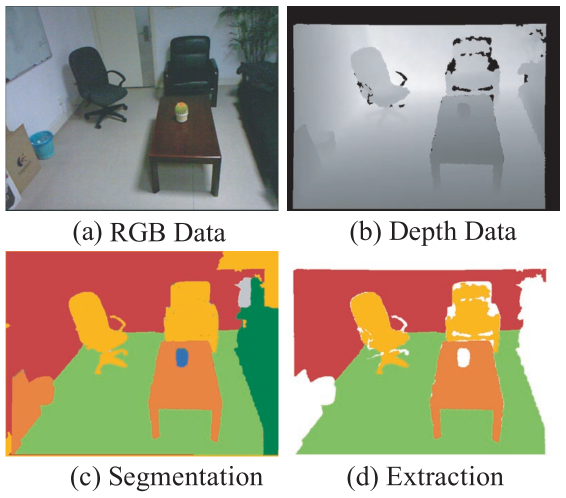


Figure 4: *Object extraction. Note the black pixels on the depth image are also discarded in object extraction due to lack of depth information there.*

final answer. For estimation of the model class label with a given segmented object $\mathbf{O}$, we simply average all the conditional class distributions from the candidate votes as the final distribution:

$$p(\mathbf{m}|\mathbf{O}) = \frac{1}{K \times N} \sum_{i=1}^{N} \sum_{l=1}^{K} p(\mathbf{m}_l|\hat{P}_i). \qquad (9)$$

where N is the number of sampled patches, and K is the number of trees in the forest.

To estimate the orientation angles, we first use the mean shift algorithm to cluster the mean of the distribution of orientation angles from all the votes. The kernel radius is set to be 15 degrees. For those votes falling in the cluster with largest number of points, we average the means of their orientation angles and distance distributions to give the final estimation. In this step, we discard leaves whose variance sum of random variables is larger than 0.3, since leaves with high variances are not informative and might introduce outliers in the clustering.

The models of three highest probabilities are the candidates of similar shape to the segmented object. Our system first performs model placement procedure (see the detailed description below) and chooses the one with the highest matching score as the final result. As an interactive system, the user is allowed to override the result by selecting one from the suggested models.

Training data generation: For each model in the database, we render its depth images by randomly sampling the orientation angles and distance parameters. Specifically, the roll angle is sampled in the range from -15° to 15°, pitch from -30° to 30°, and the range of yaw is 360°. The ranges are set so that they can simulate common camera poses when people take pictures of an indoor scene. The range of translation t are from 1.3 to 2.3 to cover the possible distance variation from the camera to the objects in the image. Totally 540 depth images of dimension 640 $\times$ 480 are generated for each model in the database.

4.2 Construction Procedure

In this section, we describe the details of the other three steps in the construction of a 3D indoor scene.

Object extraction from segmentation result: The output of the segmentation algorithm is an object label for each pixel in the image. The object extraction step is to convert this pixel-level representation into an object-level representation. This is done by checking connected regions in both the image domain and geometry domain.

We first extract a connected region with the same object label from the segmented images. However, two objects separated in 3D are possibly connected to each other in their projections to a 2D image. Since the objects in an indoor scene are usually put on the ground, we project the extracted connected regions in the 2D image onto the ground plane according to the depth information, and detect independent regions in this dimension. With this operation, most objects in the scene can be correctly detected. We deem those regions, whose average maximum class probability is less than 0.5 or their areas are less than 3,000 pixels, to be unreliable objects that are not suitable for reconstruction, and discard them in the subsequent steps. Figure 4 shows the extracted objects after this preprocessing. The salient objects in the scene are correctly preserved, and the noisy regions are eliminated.

3D model placement: After the model matching with random regression forest, we have determined which model in the database is most similar to the segmented object and its estimated orientation and scale. However, the estimated transformation usually contains error due to noise in the depth data. 3D model placement aims to optimize the estimated orientation and scale of the matched model to achieve the largest overlap between the model and the segmented object. Specifically, we maximize the following objective function in model placement:

$$\max_{\mathbf{T}} \sum_i \exp(-d(\mathbf{p}_i, \mathbf{T}(\mathbf{M}))), \tag{10}$$

where $\mathbf{T}$ contains six parameters: rotation along the normal of ground plane, 2D translation on the ground plane and three canonical scale parameters of the model, since we assume most objects in the indoor scene are put on the ground or the plane parallel to the ground. $\mathbf{p}_i$ is a pixel on the segmented object. $d(\mathbf{p}_i, \mathbf{T}(\mathbf{M}))$ is the distance between $\mathbf{p}_i$ and the model $\mathbf{M}$ under the transformation $\mathbf{T}$, and it is computed via projecting $\mathbf{p}_i$ to the transformed model along its normal. If the angle between the normal at $\mathbf{p}_i$ and the normal at its projected point on the model exceeds a threshold or it cannot be projected to model $\mathbf{M}$ along its normal, the value of $d(\mathbf{p}_i, \mathbf{T}(\mathbf{M}))$ is set to be infinite to cancel its contribution to the objective function. We perform gradient decent optimization to maximize the objective function, and the derivatives are computed numerically. An 3D model placement result is shown in Figure 5. In case the automatic model placement result is not satisfactory, our system allows the user to interactively adjust it.

Scene registration: To reconstruct an indoor scene, we need to register all the reconstructed objects from the captured RGBD images into one coordinate system. The most computation-intensive step in registration is how to figure out the correct correspondences between two images. Similar to the registration algorithm in [Henry et al. 2012; Du et al. 2011], we also perform the RANSAC algorithm to compute the correct correspondences between frames using the SIFT feature [Lowe 2004], and the semantic labels are used to cull the wrong correspondences. Interactive loop closure is also adopted to remove the global inconsistency in frame-by-frame registration [Du et al. 2011].

Wall and floor construction: Walls and floors are important to the layout of the indoor scene. After object displacement, we fit planes to those regions labeled wall and floor and compute the bounding box of those 3D points as their final geometry representation.

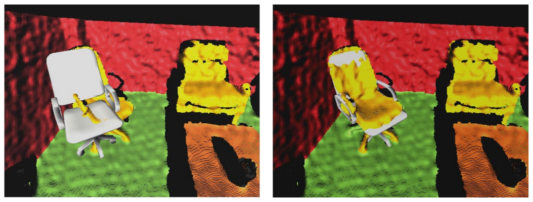


Figure 5: *Model placement. Left:The initial orientation estimated from random regression tree. Right:Optimization result. The depth data in Figure 4 are rendered in 3D to show the model placement result.*

Scene	#Photo	#Object	#Stroke	Labeling/placement
Bedroom	2	5	15	9 sec./10 sec.
Office	3	7	8	12 sec./12 sec.
Lab	6	20	28	23 sec./32 sec.
Lounge	11	19	30	43 sec./40 sec.

Table 1: *Statistics & timing. #Stroke indicates the number of strokes drawn by the user to refine the segmentation. Labeling indicates the total segmentation and labeling time for the captured RGBD images in seconds. Placement indicates the time in seconds to replace the segmented out objects with 3D models in our database. The major computation cost in placement is in the optimization of orientation and scale parameters in 3D model placement.*

Planes that cover image regions with less than 8,000 pixels are discarded to eliminate the influence of outliers. Constraints, such as the perpendicularity between the wall and floor, are imposed following the optimization procedure in [Li et al. 2011].

5 Experimental Results

We have implemented the system on a 2.83G Hz quad core Intel PC with 8G memory and tested the indoor scene prototyping system on a variety of indoor scenes, such as bedroom, office and lab scenes [1]. Table 1 lists the statistics and timing of the modeling of each scene. The statistics of our 3D model database is in the supplementary material.

Context-aware image segmentation: Figure 3 illustrates the improved labeling accuracy through the context-aware image segmentation. As shown in Figure 3a, the segmentation result from the automatic indoor scene image segmentation algorithm in [Silberman and Fergus 2011] is not satisfactory. Erroneous labels occur: the legs of the office chair are labeled to be floor and the monitors in the scene are labeled to be bed and wall. With the user sketched strokes to correct the label on the first image, the second image can be segmented well with an updated appearance and geometry model, since the updated model reflects the correlations on the appearance and shape of the objects in the same indoor scene. The rightmost two images in Figure 3b show the comparison. The erroneous labels in automatic segmentation, such as the office chair legs, monitors and table, are greatly improved by using the updated discriminative model in the data terms for graph cut. Overall, in the semantic of indoor scene modeling, the number of user strokes required in context-aware image segmentation is around 40% less than the strokes required to correct the segmentation result from fully automatic indoor scene image segmentation algorithm in [Silberman and Fergus 2011].

Model matching: There are a few parameters to control the setup and testing of the random regression forest. We train totally 15 trees for each forest, and the maximum tree depth is 18. The patch size is 120×80 pixels, and 75 patches are sampled in each training image. The memory footprint of the forest is dependent on the number of trees, their maximum depth and the probability distribution stored at the leaf nodes. The average is around 150 mega-bytes for 40 models in our current system.

We use 25 segmented objects from the captured depth images to test the recognition accuracy of the random regression forest. They are selected to be of different scale and orientation. We manually determine the three best matches for them in the database, and deem the recognition to be correct if the best matches appear in the top three highest probabilities. The recognition accuracy is the percentage of success matches for them. The plot in Figure 8a shows

[1]Data available at http://www.weiweixu.net/indoor/indoordata.zip

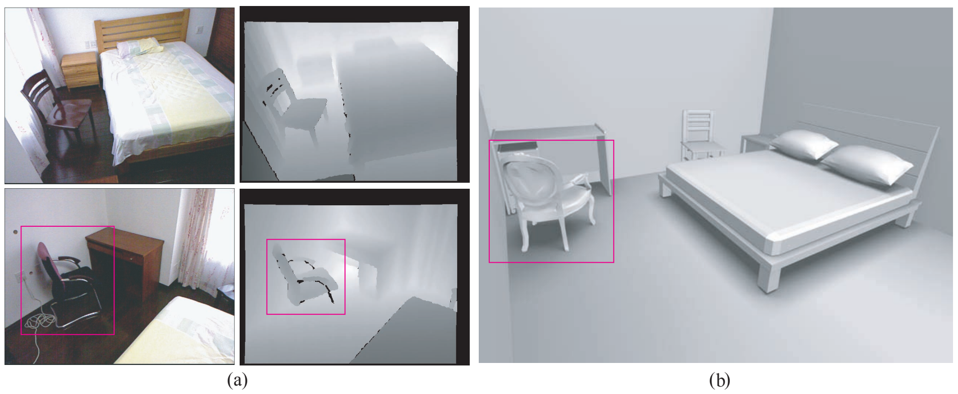


Figure 6: *Bedroom scene modeling. (a) Captured RGBD images. (b) The reconstruction result. As the depth data of the chair leg in second image are missing (indicated by the red rectangle), our system erroneously recognizes a chair of different style as its similar model.*

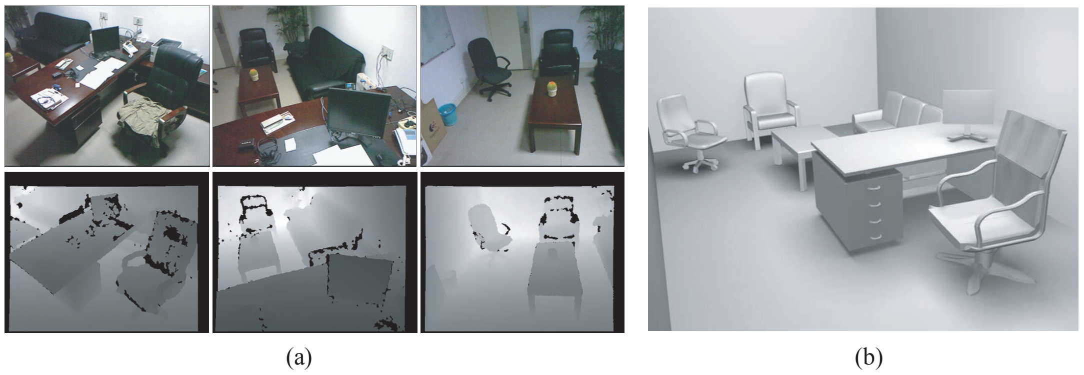


Figure 7: *Office scene modeling. (a) Captured depth images. (b) Reconstruction result.*

the recognition accuracy as a function of the number of sampled patches at each training image. The accuracy increases with an increasing number of sampled patches. We thus choose 75 in current implementation to balance recognition accuracy and computational cost in the training stage. The plot in Figure 8b shows the recognition accuracy is dependent on the sampling stride for testing. The smaller the stride value, the more patches we are sampling in the captured depth data for the testing. This plot shows that recognition accuracy benefits from densely sampled test data, since it does not lose information in the captured data. We thus chose the stride to be 15 for testing and the averaged testing time for a segmented out object is 0.2 second. We also test the influence of maximum tree depth on recognition accuracy (Figure 8c). In the current implementation, we use maximum depth of 18 since it achieves better orientation and scale estimation while keeping the recognition accuracy high. Please see the supplementary material for the RGBD images of the selected objects.

Quick-prototyping: Our system support modeling the indoor scenes quickly from sparsely captured images. The reconstruction procedure is progressive: Once the user captures a new image, the objects in the image are segmented, replaced by the 3D models in database and merged into the scene reconstructed from previous images.

Our system quickly reconstructs small scale indoor scenes. Figure 6 shows the modeling result for a bedroom. For this small scale indoor scene, only two RGB-D images are necessary to cover the main featured objects in the scene for quick prototyping. Its overall modeling session takes less than 2 minutes. Figure 7 illustrates the reconstruction of an office room with three RGB-D images, whose modeling session is also less than 2 minutes. Figure 11 illustrates another reconstruction result of a bed room using NYU dataset [Silberman et al. 2012], which shows that our reconstruction algorithm do not limit to carefully captured RGBD images.

Our system is also effective in the reconstruction of medium scale indoor scenes. Figure 1 illustrates the reconstruction result for a lab room. Its area is around 30 square meters, and we capture 6 images along a planned path to cover the whole scene. Due to the polarization effect in liquid crystal displays, it is possible that Microsoft Kinect camera cannot capture its depth since the active infrared light emitted by the camera might not be reflected (Figure 10). In this case, we let the system to reuse matched 3D monitor models in the scene for those monitors without depth. Even though there are severe occlusions for the table object in the scene, our model matching algorithm can still identify the similar 3D model but failed to detect their orientation and scales. The orientation and scale of table models are interactively adjusted to match the depth data better. The overall modeling session takes 18 minutes. Another reconstruction result for a lounge scene is illustrated in Figure 9. Even though it consists of objects of different styles and sizes, our system still successfully figures out their similar models in database to reconstruct the scene. Please see the supplementary material for the complete set of captured RGBD images for these two medium scale scenes.

Failure case: Due to the varying materials and lighting conditions in a real scene, the Microsoft Kinect camera might miss the depth data of important features of objects in the scene. As illustrated in

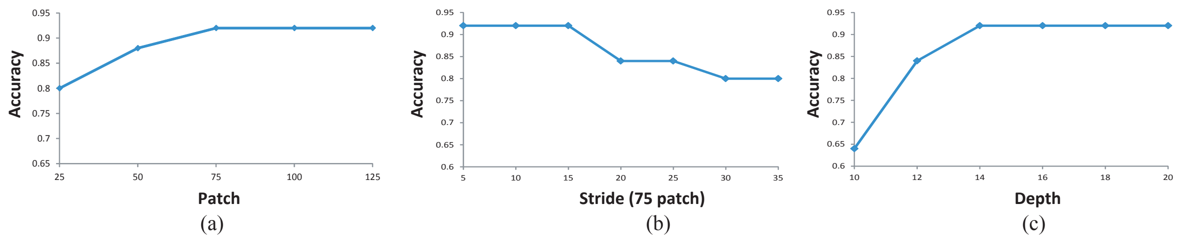


Figure 8: *Effect of matching parameters on recognition accuracy. (a) Accuracy as function of sample patch number on training images. (b) Accuracy as function of testing stride. (c) Accuracy as function of tree depth*

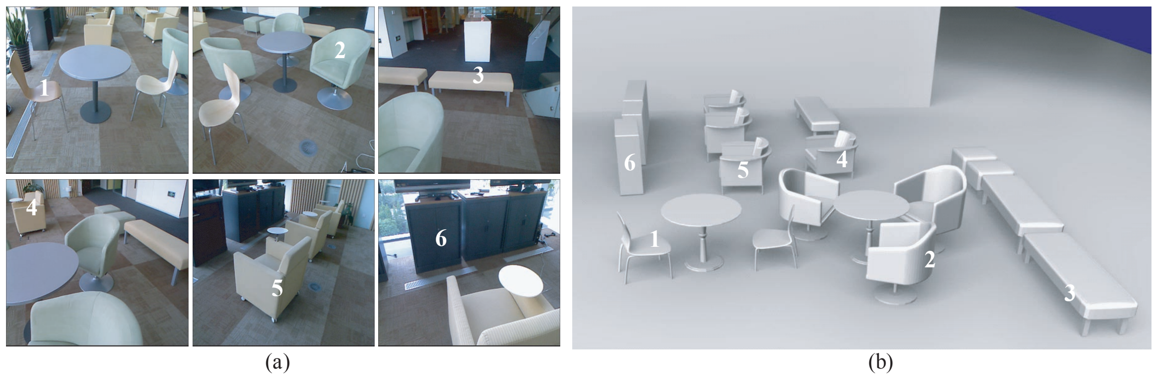


Figure 9: *Lounge scene with 19 objects. (a) Captured images (6 out of 11 captured images are shown here). (b) Reconstruction result. The numbers in white indicate the correspondence between objects in the image and their reconstruction result.*

Figure 6, the model matching algorithm returns a different type of 3D chair model in the database for the chair in the bottom image since the depth data of its leg are missing.

6 Discussion

The goal of our system is to enable the average user (not a 3D expert) to model his living or working environments. To this end, we try to automate the modeling process as much as possible while leveraging a reasonable amount of user interaction to improve robustness. In particular, our system tries to reduce 3D interaction as much as possible since the user is not expected to be a 3D expert.

An alternative to our approach is to develop a simple interface to allow users to manually pick 3D models and put them together according to the captured RGBD images. Although this is certain, the interaction required would be tedious and much effort would have to be devoted to manually browsing the database to find similar models and carefully resizing them and placing them at appropriate positions and orientations. With today's 3D interface, this is especially challenging for a user without 3D expertise. With our approach, most of this tedious manual work is automated. The most required user interaction is to draw 2D strokes to assist the segmentation algorithm. 3D models can be placed into the depth images automatically in most cases. Thus, the interaction effort is significantly reduced even for the moderately complex scenes shown in our paper. For all examples shown in the paper, only Figure 1 and Figure 9 need some 3D interaction. In Figure 1 we manually placed the monitors (due to the missing depth information in the RGBD images) and adjusted the orientation of a chair and the position of a desk. In Figure 9 we only rotated two chairs to correct their orientations. Moreover, the random-forest-regression based 3D model retrieval can suggest the similar models matched with the depth data automatically. This capability is important since it would be time consuming for the user to manually browse the database for similar 3D models for every object in the scene.

Our experience with medium-scale real-world scenes indicates that this is a powerful way to model 3D scenes. We believe our results will inspire further research on 3D scene modeling from a sparse set of RGBD images with the help of a 3D model database and reasonable amount of user interaction.

7 Conclusion and Future work

We presented an interactive semantic modeling approach for indoor scenes. The captured indoor scene images are first segmented into regions with object label, and then the segmented objects are replaced by their matched 3D models in the database. As the user continues to capture images of an indoor scene, our system is capable of progressively reconstructing a prototype of the indoor scene. The reconstructed semantic scene can directly applied in computer graphics applications, not only in rendering and gaming which only requires geometry information but also in applications requiring semantic scene information, such as furniture layout.

The limitation of our approach is that the geometry details of the objects are missing in the reconstructed scene. We believe that the similarity between the reconstructed scene and real scene can be significantly improved if the scale of the 3D model database is increased. Recognition accuracy is dependent on the quality of the captured depth data. Although the random regression forest based model matching algorithm can handle noise and partial data well, it still fails to figure out the best matches when the depth data of important features of the objects are missing. Currently, the size of the 3D model database is relatively small, totally 180 models in the database. The scalability of random regression tree based model recognition algorithm to large scale database needs to be further tested in terms of the recognition accuracy and memory footprint.

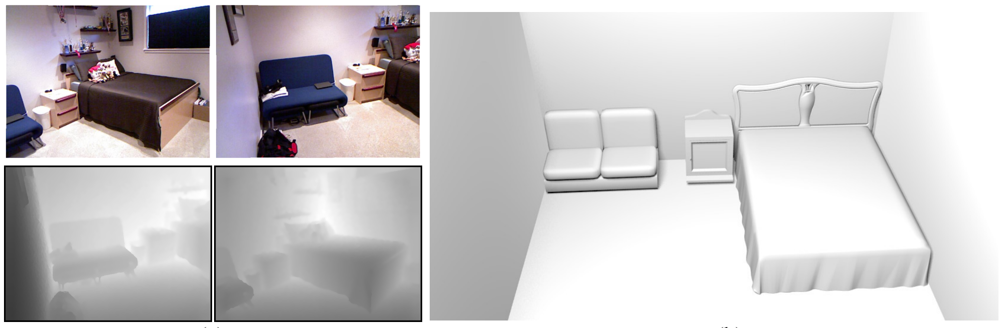


Figure 11: *A reconstruction result using NYU Data set (http://cs.nyu.edu/ silberman/datasets). (a) Two selected indoor RGBD images. (b) Reconstruction result.*

In the future, we plan to investigate part recognition of the objects in the scene to facilitate deformation of the model in the database to better fit the acquired depth data. We are also interested in various applications of semantic indoor scene modeling, such as context aware augmented reality, virtual indoor decoration and so on.

Acknowledgements

Kun Zhou is supported by NSFC (No. 61272305) and the 973 program of China (No. 2009CB320801). Weiwei Xu is partially supported by NSFC (No. 61272392). We would like to thank Steve Lin for proof reading, and the SIGGRAPH reviewers for their constructive comments.

References

ANAND, A., KOPPULA, H. S., JOACHIMS, T., AND SAXENA, A. 2011. Contextually guided semantic labeling and search for 3d point clouds. *CoRR abs/1111.5358*.

BLUM, M., SPRINGENBERG, J. T., WULFING, J., AND RIEDMILLER, M. 2011. On the applicability of unsupervised feature learning for object recognition in rgb-d data. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

BO, L., REN, X., , AND FOX, D. 2011. Depth kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (Nov.), 1222–1239.

BREIMAN, L. 2001. Random forests. *Mach. Learn. 45*, 1 (Oct.), 5–32.

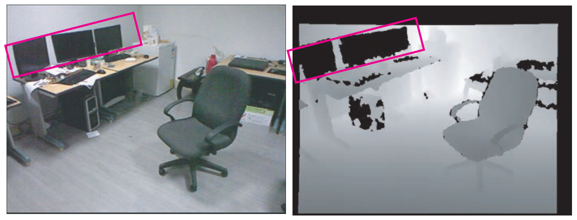


Figure 10: *A depth data missing case. Due to the polarization effect in liquid crystal display, it is possible for Microsoft Kinect camera to miss the depth data of the three monitors in the RGBD image.*

BRONSTEIN, A. M., BRONSTEIN, M. M., GUIBAS, L. J., AND OVSJANIKOV, M. 2011. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph. 30*, 1 (Feb.), 1:1–1:20.

CLARK, J. 2011. Object digitization for everyone. *IEEE Computer 44*, 10 (Oct.), 81–83.

DIETTERICH, T. G. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn. 40*, 2 (Aug.), 139–157.

DU, H., HENRY, P., REN, X., CHENG, M., GOLDMAN, D. B., SEITZ, S. M., AND FOX, D. 2011. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of Ubicomp*, 75–84.

FANELLI, G., WEISE, T., GALL, J., AND GOOL, L. V. 2011. Real time head pose estimation from consumer depth cameras. In *Proceedings of the 33rd international conference on Pattern recognition*, Springer-Verlag, Berlin, Heidelberg, DAGM'11, 101–110.

FOX, D., BURGARD, W., DELLAERT, F., AND THRUN, S. 1999. Monte carlo localization: efficient position estimation for mobile robots. In *Proceedings of AAAI'99/IAAI'99*, American Association for Artificial Intelligence, Menlo Park, CA, USA, AAAI '99/IAAI '99, 343–349.

FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., AND JACOBS, D. 2003. A search engine for 3d models. *ACM Trans. Graph. 22*, 1 (Jan.), 83–105.

FURUKAWA, Y., AND PONCE, J. 2010. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 8 (Aug.), 1362–1376.

FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2009. Manhattan-world stereo. *CVPR*, 1422–1429.

FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2009. Reconstructing Building Interiors from Images. In *ICCV*.

GOLOVINSKIY, A., KIM, V., AND FUNKHOUSER, T. 2009. Shape-based recognition of 3d point clouds in urban environments. In *ICCV*.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press.

HENRY, P., KRAININ, M., HERBST, E., REN, X., AND FOX, D. 2012. Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments. *Internatioal Journal of Robotics Research 31*, 5, 647–663.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., AND FITZGIBBON, A. 2011. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '11, 559–568.

JANOCH, A., KARAYEV, S., JIA, Y., BARRON, J. T., FRITZ, M., SAENKO, K., AND DARRELL, T. 2011. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV Workshop on Consumer Depth Cameras for Computer Vision*, 1168–1174.

JOHNSON, A. E., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 5 (May), 433–449.

KIM, Y. M., MITRA, N. J., YAN, D. M., AND GUIBAS, L. 2012. Acquiring 3d indoor environments with variability and repetition. *To appear in SIGGRAPH Asia 2012*.

KOPPULA, H., ANAND, A., JOACHIMS, T., AND SAXENA, A. 2011. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*.

LAI, K., AND FOX, D. 2010. Object recognition in 3d point clouds using web data and domain adaptation. *International Journal of Robotics Research 29*, 8 (July), 1019–1037.

LAI, K., BO, L., REN, X., AND FOX, D. 2011. A large-scale hierarchical multi-view rgb-d object dataset. In *Proc. of IEEE International Conference on Robotics and Automation*.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Trans. Graph. 23*, 3 (Aug.), 303–308.

LI, Y., WU, X., CHRYSATHOU, Y., SHARF, A., COHEN-OR, D., AND MITRA, N. J. 2011. Globfit: consistently fitting primitives by discovering global relations. *ACM Trans. Graph. 30*, 4 (Aug.), 52:1–52:12.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov.), 91–110.

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M., AND KOLTUN, V. 2011. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30* (July), 87:1–87:12.

NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *To appear in SIGGRAPH Asia 2012*.

SCHNABEL, R., WAHL, R., AND KLEIN, R. 2007. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum 26*, 2 (June), 214–226.

SILBERMAN, N., AND FERGUS, R. 2011. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*.

SILBERMAN, N., HOIEM, D., KOHLI, P., AND FERGUS, R. 2012. Indoor segmentation and support inference from rgbd images. In *ECCV*.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph. 25*, 3 (July), 835–846.

TANGELDER, J. W., AND VELTKAMP, R. C. 2008. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl. 39*, 3 (Sept.), 441–471.

TRIGGS, B., MCLAUCHLAN, P., HARTLEY, R., AND FITZGIBBON, A. 2000. Bundle adjustment a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, Springer Verlag, 298–375.

WHITAKER, R. T., GREGOR, J., AND CHEN, P. F. 1999. Indoor scene reconstruction from sets of noisy range images. In *Proceedings of the 2nd international conference on 3-D digital imaging and modeling*, 3DIM'99, 348–357.

XIONG, X., AND HUBER, D. 2010. Using context to create semantic 3d models of indoor environments. In *BMVC*, 1–11.

YU, L.-F., YEUNG, S.-K., TANG, C.-K., TERZOPOULOS, D., CHAN, T. F., AND OSHER, S. J. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph. 30* (July), 86:1–86:12.

Appendix

We compute a 496 dimensional feature vector $\mathbf{I}$ For each patch sampled from depth images. The features are as follows:

a) Depth difference: Depth difference has been used in [Fanelli et al. 2011] for head pose estimation. We compute the depth difference between the average values of two rectangular areas in the patch to be less sensitive to the noise. The maximum size of the rectangular area is 40×40 pixels in our implementation. We randomly sample the starting position, weight and height of each rectangle, and the sampling information are kept same for each patch. A 144 dimensional feature vector is formed by sampling 144 pairs of rectangles for the computation of depth difference.

b) Normal structure tensor: It is used to measure the principle directions in the normal distribution of pixels in the patch. We subdivide the patch into 2×3 sub-patches so that the local normal distribution can be efficiently measured. For each sub-patch, a tensor is computed by the following formula:

$$\mathbf{G} = \frac{1}{N} \sum_{i}^{N} \mathbf{n}_i \mathbf{n}_i^T,$$

where $\mathbf{n}_i$ is the normal at pixel i. $\mathbf{G}$ is normalized by its Frobenius norm, i.e. $\hat{\mathbf{G}} = \frac{\mathbf{G}}{\|\mathbf{G}\|_F}$, as the final normal structure tensor feature at the sub-patches.

c) Geometry moments. Geometry moments are also computed at 6 sub-patches similar to normal structure tensor. For each sub-patch, 10 moments for (x, y, z) coordinates are computed with following equation:

$$M_{pqr} = \frac{1}{N} \sum_{i}^{N} x^p y^q z^r, p + q + r < 3.$$

In the computation of moments, the (x, y, z) coordinates are normalized according to local axis-aligned bounding box of 3D points in the sub-patch.

d) Spin image: Spin image [Johnson and Hebert 1999] is computed with 16x16 bin resolution for each patch, yielding 256 features. We found that computing principal directions at patch level for spin image feature is less sensitive to the noise in the captured depth image.