Data-Driven Contextual Modeling for 3D Scene Understanding

Yifei Shi[a], Pinxin Long[c], Kai Xu[a,c,*], Hui Huang[b,c], Yueshan Xiong[a]

[a] *HPCL, National University of Defense Technology*
[b] *Shenzhen University*
[c] *Shenzhen VisuCA Key Lab / SIAT*

Abstract

The recent development of fast depth map fusion technique enables the realtime, detailed scene reconstruction using commodity depth camera, making the indoor scene understanding more possible than ever. To address the specific challenges in object analysis at subscene level, this work proposes a data-driven approach to modeling contextual information covering both intra-object part relations and inter-object object layouts. Our method combines the detection of individual objects and object groups within the same framework, enabling contextual analysis without knowing the objects in the scene *a priori*. The key idea is that while contextual information could benefit the detection of either individual objects or object groups, both can contribute to object extraction when objects are unknown.

Our method starts with a robust segmentation and partitions a subscene into segments, each of which represents either an independent object or a part of some object. A set of classifiers are trained for both individual objects and object groups, using a database of 3D scene models. We employ the multiple kernel learning (MKL) to learn per-category optimized classifiers for objects and object groups. Finally, we perform a graph matching to extract objects using the classifiers, thus grouping the segments into either an object or an object group. The output is an object-level labeled segmentation of the input subscene. Experiments demonstrate that the unified contextual analysis framework achieves robust object detection and recognition over cluttered subscenes.

Keywords: Scene understanding, object recognition, contextual modeling, data-driven approach

1. Introduction

With the rapid development of 3D sensing techniques, the digitalization of large-scale indoor scenes has become unprecedentedly accessible to a wide range of applications. Among the most exciting and promising applications, robot-operated exploration and interaction over unknown indoor environment would benefit significantly from the availability of high-quality and realtime acquired 3D geometry information [1]. Such 3D information can not only improve robot navigation and exploration, but more importantly, facilitate efficient robot-scene interaction with fine-grained understanding of scene objects. The latter may support highly complex robot tasks such as room cleaning.

Motivated by the high demand, extensive research has been devoted to the understanding of scanned indoor scenes. Most existing works on scene understanding focus on large-scale objects, such as furniture, as well as their spatial layout [2, 3, 4, 5, 6], since the analysis

*Corresponding author: kaixu@nudt.edu.cn

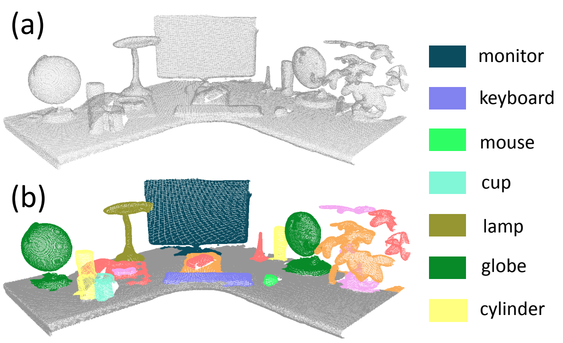

Figure 1: *Scene understanding by our method. (a): The input point cloud of a table-top scene. (b): The labeling result (legends show semantic labels in color).*

is usually limited by the quality and resolution of input scans. Recent advances in volumetric scan fusion technique (such as KinectFusion [7]) has made it possible to reconstruct quality and detailed scenes from scans captured by commodity depth camera (e.g. Microsoft Kinect and Asus Xtion). The dense point clouds processed by KinectFusion can well capture small scale objects such as household objects, which enables detailed understanding at a subscene level, e.g. many objects placed a tabletop; see Figure 1.

Object analysis at subscene level is arguably much more challenging than that at whole scene level. *Firstly*, unlike furnitures which are usually sparsely distributed in an indoor scene, household objects are often highly cluttered due to the limited space of supporting surfaces [8]. For example, a tabletop scene is typically cluttered with many on-table objects. *Secondly*, repetition of objects, which is ubiquitous among furnitures and has been extensively exploited in previous works [3, 5], may not be as commonly seen among household objects. For example, the objects placed on a table are mostly unique within the subscene. *Thirdly*, from the acquisition point of view, smaller objects are often more sensitive to scanning imperfection. These challenges make the existing methods, dealing with large-scale furniture layout, unsuitable for the object analysis of small-scale subscenes.

To address these challenges, it seems a natural option is to fully utilize the inter-object relations, or contextual information. However, a key prerequisite for contextual scene analysis is that all objects are segmented and labeled with semantic tags [9], which is apparently infeasible for an unsegmented scene. Essentially, context is defined with objects. Without knowing objects, how can we utilize contextual information to help the identification of objects? In this work, we try to tackle this problem through integrating the discovery of both individual objects and object groups into a unified framework. While the former involves grouping parts into an object, which detects individual objects, the latter amounts to finding structure groups [10] composed of multiple objects, which can actually enhance or reinforce the detection and recognition of objects within the structure group. The key idea is that contextual information could benefit the detection of either individual objects or object groups, when objects are unknown. However, both can contribute to object extraction.

To enable such unified framework, we take a data-driven approach equipped with several key procedures. First, we propose a robust segmentation method to partition a indoor scene into segments which each represents either an independent object or a part of some object. We then train a set of classifiers for both individual objects and object groups, based on a database of 3D scene models. To improve the classification accuracy, we employ multiple kernel learning (MKL) [11] to learn per-category optimized SVM classifiers for various objects and object groups. Finally, we perform a graph matching to extract objects using the classifiers, thus grouping the segments into either an object or an object group. The input of our algorithm is an indoor scene point cloud, and the output is an object-level labeled segmentation of the input scene. Experiments demonstrate the robust performance for both segment extraction and object recognition on several subscenes.

Our approach possesses two key features compared with previous methods. First, we perform a segmentation process before recognition, which leads to robust handling of cluttered scenes. Second, instead of solving the recognition of individual objects and object groups as two separate problems, we encode features of both individual objects and object layout into a unified classifier via contextual modeling.

2. Related Work

Scene understanding is a long-standing research topic which has received extensive research from both computer vision and computer graphics community. We mainly review those works which take 3D point clouds as input.

Point cloud segmentation. Mesh segmentation is a fundamental shape analysis problem in computer graphics, for which both heuristic methods [12] and data-driven approach [13] have been extensively studied over the years. On the other hand, the segmentation of 3D point clouds remains to be a challenging problem.

There are three kinds of methods for point cloud segmentation [14]. The first type is based on primitive fitting [3, 15, 5]. It is hard for these methods to deal with objects with complex shape. The second kind of techniques is the region growing method. Nan et al. [2] propose a controlled region growing process which searches for meaningful objects in the scene by accumulating surface patches with high classification likelihood. Berner et al. [16] detect symmetric regions using region growing. Another line of methods formulates the point cloud segmentation as a Markov Random Field (MRF) or Conditional Random Field (CRF)

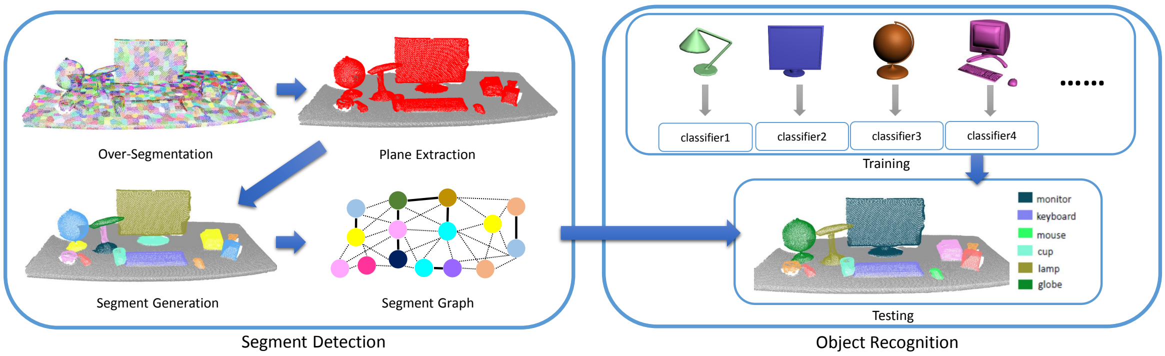

Figure 2: *An overview of our algorithm. We first over-segment the scene and extract the supporting plane on the patch graph, then segment the scene into segments and represent the whole scene using a segment graph (a). To obtain the contextual information, we train a set of classifiers for both single objects and object groups using multiple kernel learning (b). The classifiers are used to group the segments into objects or object groups (c).*

problem [4, 17, 14]. A representative random field segmentation method is the min-cut algorithm [17]. The method extracts foreground from background through building a KNN graph over which min-cut is performed. The shortcoming of min-cut algorithm is that the selection of seed points relies on human interaction. We extend the min-cut algorithm by first generating a set of object hypotheses via multiple binary min-cuts and then selecting the most probable ones based on a voting scheme, thus avoiding the seed selection.

Object recognition. Recently, the development of commodity RGB-D cameras has opened many new opportunities for 3D object recognition and scene recognition [18, 19]. With the ever-growing amount of 3D models becoming available, data-driven approach starts to play an important role in 3D object recognition and has gained great success [20].

Nan et al. [2] propose a search-classify approach to scene understanding by interleaving segmentation and classification in an iterative process. Li et al. [6] propose scene reconstruction by retrieving objects from a 3D model database. Song et al. [21] render database models from hundreds of viewpoints and train an exemplar-SVM classifier for each of them to achieve object recognition. Their method overcomes several difficulties in object recognition, such as the variations of texture, illumination, etc. Chen et al. [22] utilize contextual information for indoor scene understanding. Small objects and incomplete scans can be recognized with the help of contextual relationships learned from database objects. Our method lends itself to cluttered indoor scene analysis through integrating segmentation and recognition into a single framework, which leads to a better performance when dealing with close-by objects than the contour-based method of [22].

Another line of analysis method is unsupervised learning based on the presence of repetitions or symmetries in indoor scenes [3, 5, 23]. A limitation of such approaches is that such repetitive patterns are less common in subscenes dominated by household objects, e.g., a tabletop scene.

Plane extraction. Plane extraction from point cloud is another important topic in scene understanding. For example, planes can be used to improve the reconstruction of arbitrary objects containing both planar and non-planar regions [24].

Perhaps the most widely used approach for plane extraction is RANSAC based plane fitting [15]. This method scales well with respect to the size of the input point cloud and the number of planes. Mattausch et al. [5] utilize planar patches as a compact representation of the point cloud of an indoor scene, which facilitates efficient repetition detection in a large-scale scene point cloud. Zhang et al. [24] perform plane extraction to delineate non-planar objects. Plane extraction has also been performed in the analysis of RGB-D data [25, 26]. These works trim the plane boundary and convert the input data into a compact polygonal representation. Recently, Monszpart et al. [27] propose to reconstruct the raw scan of man-made scenes into an arrangement of planes with both local fitting and global regularization.

3. Overview

The input of our algorithm is a 3D point cloud of indoor scene acquired and fused by KinectFusion. Our goal is to detect objects in the scene and recognize their semantic categories automatically. Our method proceeds in two stages. First, we segment the point cloud into segments representing potential objects. Second, to achieve object extraction and recognition, we propose a joint estimation of individual objects and object groups, as well as their semantic categories.

Segment detection. In the first stage, we segment the input scene (Figure 2 (a)). Specifically, we first over-segment the entire scene and build a patch graph. We then extract the supporting plane with a method integrating RANSAC primitive fitting into graph-cut. After plane extraction, the remaining points are grouped into isolated groups. Within each group, we generate segments via a robust segmentation algorithm, which takes both geometry and appearance information into account. Based on the segmentation, we represent the entire scene as a segment graph with two types of edges representing direct spatial adjacency (solid lines in Figure 2) and spatial proximity (dashed lines) between two segments, respectively.

Object extraction and recognition. In the second phase, we extract objects via recognizing both individual objects and object groups within a unified framework, based on the above segment graph representation.

In an off-line stage, we train per-category optimized SVM classifiers with multiple kernel learning for both objects and object groups. The classifiers are trained using 3D database models. Each 3D model is first converted into 3D point cloud using virtual scanning and segmented using the method mentioned above. We then extract features from the corresponding segment graph and train classifiers based on the graph.

In the online stage, we extract objects or object groups from the segment graph of the input scene, through searching for the subgraph matching corresponding to the occurrence of database objects and object groups. Once a matched subgraph is found, we use the corresponding SVM classifier to estimate the probability of the match. Finally, we solve a labeling optimization which minimizes the overall matching cost for all matching probabilities.

4. Segment detection

Our goal is to partition the input scene into segments which each represents either an independent object or a part of an object. In order to segment objects from cluttered scenes, we propose an unsupervised segment detection approach to detect segments in 3D scene.

Specifically, we first over-segment the input point cloud into a set of patches (Sec. 4.1) and detect the supporting plane (Sec. 4.2). We then group the remaining patches to extract potential objects or parts (Sec. 4.3) and represent them as a segment graph (Sec. 4.4). See Algorithm 1 for an overview of our method.

4.1. Patch graph generation

We first over-segment the entire scene S into several patches, using the method in [28]. We build a patch graph based on the patches, denoted with $G_\mathrm{p} = (\mathcal{V}_\mathrm{p}, \mathcal{E}_\mathrm{p})$, where $\mathcal{V}_\mathrm{p}$ and $\mathcal{E}_\mathrm{p}$ represent the patches and the near-by relations within the patches, respectively. Specifically, the near-by relations are determined by comparing the nearest distance between two patches with a threshold.

Essentially, our segment detection algorithm is a graph-cut based approach. The most vital component for graph-cut method is the definition of smooth term. In this section, the smooth terms for all graph-cut optimization are identical, which we first define here:

$$E_s(x_u, x_v) = w_c \cdot E_c + w_p \cdot E_p + w_n \cdot E_n, \qquad (1)$$

where x_u, x_v are two adjacent patches. E_c, E_p, E_n are the differences between two adjacent patches in terms of color, planarity and normal. w_c,w_p,w_n are the weights.

E_c and E_p are computed based on the chi-square distance of the color and planarity histogram between u and v, we normalize them to $(0, 1)$. It is worth mentioning that the planarity histogram are computed as follow: first compute the least-square plane for a patch, then built a histogram for distances of all points in the patch to the plane. The formulation for E_n is different for convex and concave situations. Specifically, the formulation is:

$$E_n(x_u, x_v) = 1 - \eta(1 - \cos\theta_{u,v}), \qquad (2)$$

where $\theta_{u,v}$ is the angle between the average normals of patch P_u and P_v. For η, we take 0.01 (a small value) if the two adjacent patches form a convex dihedral angle

Algorithm 1 :Segment Detection.

Input: scene S
Output: segment graph G_s
$G_p \leftarrow$ OverSegment(S);
$S \leftarrow$ PlaneExtract(S,G_p); //extract plane
$\mathcal{H} \leftarrow$ SegHypGen(S,G_p); //generate seg. hypo.
$T \leftarrow$ SegHypSelect($\mathcal{H}$); //select seg. hypo.
$G_s \leftarrow$ SegGraGen(T); //generate seg. graph
return G_s;

and 1 otherwise, to encourage cuts around a concave region [29].

Our smooth term takes both geometry (planarity and normal) and appearance (color) factors into consideration, thus makes the patches belong to different objects can be detected easily.

4.2. Supporting plane extraction

Supporting plane is usually the largest object in most subscenes of an indoor scene, such as tables, beds, shelves, etc. The extraction of supporting plane is especially useful since it makes the detection of objects on top of the supporting plane easier. Therefore, the first step of our segment generation is supporting plane extraction. For this task, perhaps the most straightforward approach is RANSAC based primitive fitting [15]. Since the objects placed on the supporting plane may be very small or thin, setting a hard threshold for point-to-plane distance may cause a lot of false positives. We therefore improve this method by adding a graph-cut optimization, to robustly segment on-top objects from the supporting plane.

We try to assign each patch a binary label, denoted by $X = [x_1, \dots, x_n]$ with $x_i \in \{0, 1\}$. $x_i = 1$ if patch P_i lies in the plane, and $x_i = 0$ otherwise. We formulate the labeling problem as graph cuts over the patch graph:

$$E(X) = \sum_{u \in \mathcal{V}_p} E_d(x_u) + \sum_{(u,v) \in \mathcal{E}_p} E_s(x_u, x_v), \qquad (3)$$

where the data term is defined as:

$$E_d(x_u) = \begin{cases} \delta, & \text{if } x_u = 1 \\ (1 - \frac{p}{p_{max}}) \cdot (1 - \frac{d}{d_{max}}) \cdot \cos\theta_{u,l}, & \text{if } x_u = 0 \end{cases}$$

where δ is a constant value, d the distance between the center of u to the plane, and p the planarity of the patch. d_{max} and p_{max} is the maximum distance and planarity, respectively. We compute p as the average distance of all the points in patch P_u to its corresponding least-square fitting plane. $\theta_{u,l}$ is the angle between the average normal of P_u and the normal of the plane.

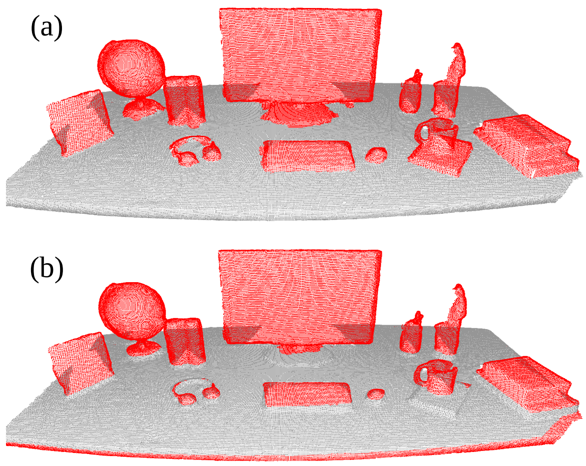

Figure 3: *Plane extraction from the point cloud of a tabletop scene by using our method (a) and RANSAC based primitive fitting (b), respectively. While our method can segment out the supporting plane accurately, RANSAC missed some points due to the thin objects.*

Figure 3 (a) demonstrates the segmentation results of our method. As a comparison, the RANSAC based primitive fitting can also get the majority of points correctly, but it fails when dealing with small and thin objects, as is shown in Figure 3 (b).

4.3. Segment generation

Segment hypothesis generation. After plane removal, object extraction only amounts to segmenting the isolated groups of patches on top of the supporting plane into individual objects. To solve the problem, we propose to first generate a set of segment hypotheses and then select the most prominent ones based on a voting algorithm.

We first update the patch graph G_p by removing the nodes belonging to the extracted plane. Based on the updated patch graph, we generate segment hypotheses by performing several times of binary graph cut, where the foreground corresponds to potential objects or prominent parts.

Different from other graph cut method, we do not select foreground seed heuristically. Instead, we use every patch as seed and perform binary graph cuts for multiple

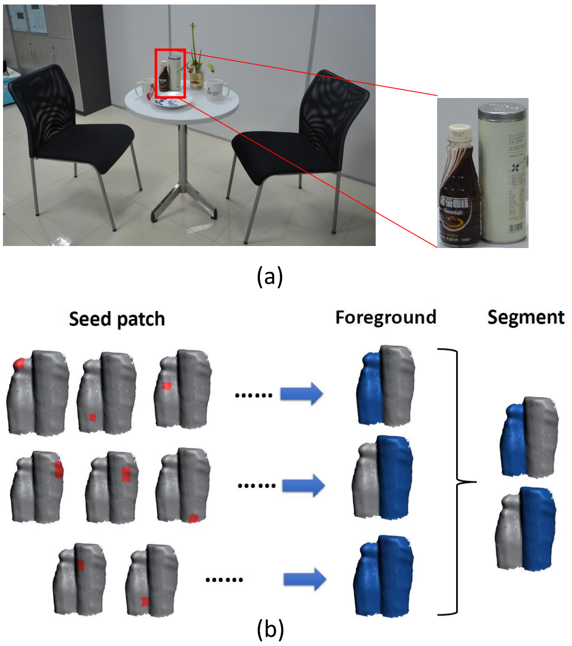

Figure 4: *Illustration of our segment detection method. The scene is composed of two bottles stuck together on a round table (a). We use every patch as seed to generate many foreground hypotheses and then select the most prominent ones (b).*

times, generating many candidate foregrounds. In each binary cut, we select one patch as foreground seed but do not prescribe any seed for background. This is performed by introducing a background penalty for each non-seed patch [30]. Specifically, we select one patch, denoted by P_s, labeling it as foreground $x_s = 1$, and minimize over binary patch labels $X = [x_1, \ldots, x_n]$, $x_i \in \{0, 1\}$ (n is the number of patches) the following parametric energy function:

$$E^{\lambda}(X) = \sum_{u \in \mathcal{V}_{\mathrm{p}}} E_d^{\lambda}(x_u) + \sum_{(u,v) \in \mathcal{E}_{\mathrm{p}}} E_s(x_u, x_v), \tag{4}$$

where the data term is defined as:

$$E_d^{\lambda}(x_u) = \begin{cases} \infty, & \text{if } x_u = 0 \text{ and } u = s \\ 0, & \text{if } x_u = 1 \text{ and } u = s \\ 0, & \text{if } x_u = 0 \text{ and } u \neq s \\ f_u, & \text{if } x_u = 1 \text{ and } u \neq s \end{cases}$$

$$f_u = \begin{cases} k(d(P_s, P_u) - \lambda), & \text{if } d(P_s, P_u) > \lambda \\ 0, & \text{otherwise.} \end{cases}$$

f_u is the background penalty which penalizes a non-seed patch which is distant from the foreground seed.

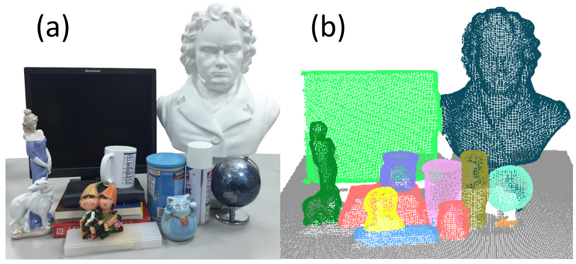

Figure 5: *Segment detection from the point cloud of a highly cluttered scene (a) by using our method (b). The input data has a lot of close-by objects and the back view is not scanned, which makes the segmentation quite challenging. Our method can segment out most objects accurately.*

$d(P_s, P_u)$ is the distance between the centers of patch P_s and P_u. We use $k = 2.0$ for a steep penalty to quickly reject those patches whose distance to P_s is larger than λ to be labeled as foreground. The parameter λ controls the range, centered around the foreground seed, within which one seeks for foreground patches. Instead of using a hard threshold on this range, we slide λ from 0 to ℓ_{d} (the diagonal length of the bounding box of the entire scene) and find the first point where the total cut cost drops significantly (up to 50%) and take the resulting cuts as the segmentation result. The smooth term is the same as the one used for plane extraction in Eq. (1).

Once we select every patch as seed and perform graph cut for each of them, we can obtain a set of foreground segments. To filter out the redundancy, we cluster the foreground segments using non-parametric mean-shift [31]. The similarity between two segments, denoted as S and T, is measured by the Jaccard index, i.e., $s(S, T) = |S \cap T|/|S \cup T|$. For example, as is shown in Figure 4 (b), selecting the seed patches in the same row will led to identical foregrounds, thus these foregrounds will cluster together after the mean-shift processing, as the Jaccard index is high. For each cluster, we choose the cluster center as the segment hypothesis for that cluster. As a result, we obtain a pool of k hypothetic segments, $\mathcal{H} = \{H_i\}_{i=1}^k$.

Segment hypothesis selection. The set of hypotheses may overlap with each other, making the labeling of patches ambiguous. To select good hypotheses without relying on heuristics or supervision, we propose a multi-class Markov random field (MRF) segmentation with object label selection, which minimizes the following energy function:

$$E(L) = \sum_{u \in \mathcal{V}_\mathrm{p}} E_d(l_u; P_u) + \sum_{(u,v) \in \mathcal{E}_\mathrm{p}} E_s(l_u, l_v), \quad (5)$$

over the labeling for all patches: $L = [l_1, \ldots, l_n], l_i \in \{1, \ldots, k\}$.

The data term $E_d(l_u; P_u)$ is defined as the likelihood that the patch P_u belongs to a particular segment hypothesis. For instance, for patch P_u and hypothesis H_i, we define the data term as the frequency of P_u being covered by the hypotheses in H_i:

$$E_d(H_i; P_u) = -\ln\left(t(P_u, C_i) / \sum_j t(P_u, C_j)\right), \quad (6)$$

where $t(P_u, C_i) = |\{P_u \subset H_j | H_j \in C_i\}|$ is the presence times of patch P_u in cluster C_i. The smooth term is also the same as the one in Eq. (1)).

The data term selects a label for each patch based on a consensus voting by all foreground clusters: The larger a foreground cluster is, the more probable that its corresponding segment hypothesis represents an independent object, since the object is proposed by many binary segmentations. Figure 4 depicts our segment detection algorithm and Figure 5 demonstrates the segmentation results over a highly cluttered scene.

4.4. Segment graph generation

To deal with the recognition for both object and object group, we represent the entire scene as a segment graph $G_\mathrm{s} = (\mathcal{V}_\mathrm{s}, \mathcal{E}_\mathrm{s})$, where $\mathcal{V}_\mathrm{s}$ represents the segments we detected in the input scene and $\mathcal{E}_\mathrm{s}$ encodes the relationship between two segments. We use two kinds of edges to describe relations in G_s . If the shortest distance between two segments is less than a small threshold t_s, we use a *connection edge* to link them, that means the two segments contact with each other and probably belongs to the same object. If the shortest distance between two segments is large than the small threshold but less than a larger threshold t_l, we use a *proximity edge* to connect them, which means they are in the same supporting plane and has the potential to constitute a object group. The two kinds of edges represent the contextual information for intra-object part relations and inter-object object layouts, respectively. t_l is selected as slightly larger than the largest bounding box diagonal length of all object groups in the database. Figure 2 shows an illustration the segment graph of the given input scene.

5. Object Recognition

5.1. Training

When recognizing a scene containing multiple objects, human perception is predominantly affected by three levels of prior knowledge [32]: the shape information of individual parts, the part composition of individual objects, and the contextual relationship among object groups. In our object recognition procedure, we encode all these knowledge in an unified model and recognize objects and object groups simultaneously. Specifically, we train per-category optimized SVM classifiers for all kinds of objects and object groups, and then utilize these classifiers to test the category of the input segments. Here, an object group is refer to a group of objects whose co-occurrence is frequently seen in an indoor scene category [33]. For example, the monitor-keyboard-mouse combo is frequently seen in office.

Data Preparation. To learn the model from the database of 3D scene models, the first step is to convert the database models (training data) into point cloud representation, which is compatible against the input (test data), and extract features from the point clouds.

First, we download a set of 3D CAD models of household objects, denoted by $\{\Gamma_i\}$, from the internet. Each Γ_i contains the models belonging to the same shape category. Second, we collect indoor scene models from the dataset of [9] and [10]. In order to obtain object groups which are not only frequently occurring but also semantically significant, we extract local substructures $\{\Phi_i\}$ from the dataset as the focal points defined in [33]. Each Φ_i contains the substructures belonging to the same semantic group.

We then perform virtual scanning for all models/groups in $\{\Gamma_i\}$ and $\{\Phi_i\}$, similar to [2]. Such virtual scan could mimic the real situation of object clutter or incomplete scan, making the training data more suitable for learning a generalizable recognition model. After the virtual scanning, we compute segment graphs using the method described in Sec. 4 for object groups in $\{\Phi_i\}$. For individual objects in $\{\Gamma_i\}$, we perform the same process except for table extraction. The label of each virtually scanned point is determined by aligning the point cloud with the original 3D CAD models and transferring the labels based on closest point search.

Classifier Learning. We compute two kinds of features for our SVM classifier: node features and edge features.

Algorithm 2 :Training.

Input: object set $\{\Gamma_i\}$ and object group set $\{\Phi_i\}$
Output: classifiers C

for all Γ_i **do**
  for all γ_j in Γ_i **do**
    $\gamma_j \leftarrow \mathrm{VirtualScan}(\gamma_j)$;
  end for
  $g_i \leftarrow \mathrm{ConstructSegGraph}(\Gamma_i)$;
  $c_i^{\gamma} \leftarrow \mathrm{MKL}(g_i)$;
  //train SVM for each single object category
end for
for all Φ_i **do**
  for all ϕ_j in Φ_i **do**
    $\phi_j \leftarrow \mathrm{VirtualScan}(\phi_j)$;
  end for
  $g_i \leftarrow \mathrm{ConstructSegGraph}(\Phi_i)$;
  $c_i^{\phi} \leftarrow \mathrm{MKL}(g_i)$;
  //train SVM for each object group category
end for
return $C = \{c_i^{\gamma}\}_{i=1}^{m} + \{c_i^{\phi}\}_{i=1}^{n}$;

For each node, we voxelize its bounding box and extract features of shape, normal and volume as described in [21]. In addition, we estimate the oriented bounding box (OBB) for each object and measure its anisotropy:

$$c_l = \frac{s_1 - s_2}{(s_1 + s_2 + s_3)}, c_p = \frac{2\,(s_2 - s_3)}{(s_1 + s_2 + s_3)}, c_s = \frac{3 s_3}{(s_1 + s_2 + s_3)} \tag{7}$$

where s_1,s_2,s_3 are the three scales of the OBB with $s_1 > s_2 > s_3 \geq 0$. For each edge, we compute the layout similarity [33] as its feature:

$$\gamma\,(p, q) = \frac{d_H(obb\,(p)\,, obb\,(q))}{dl\,(p) + dl\,(q)}, \tag{8}$$

$$\rho\,(p, q) = angle\left(\mathbf{v}_{dir}\,(p, q)\,, \mathbf{v}_{upright}\right), \tag{9}$$

The two features measure the distance and direction between two objects, respectively.

We compute features and learn pre-category optimized SVM classifiers for each category of individual objects in $\{\Gamma_i\}$ and object groups in $\{\Phi_i\}$. Positive examples are the models from the two datasets, while negative ones are generated by using the method in [21] for individual objects and the method in [22] for object groups. In addition, we associate a triplet (n_n, n_c, n_p) with each classifier, where n_n, n_c and n_p represent the number of segments, edges and proximity edges, respectively. This triplet is used to perform a coarse matching based on the triplet, before testing with the classifier.

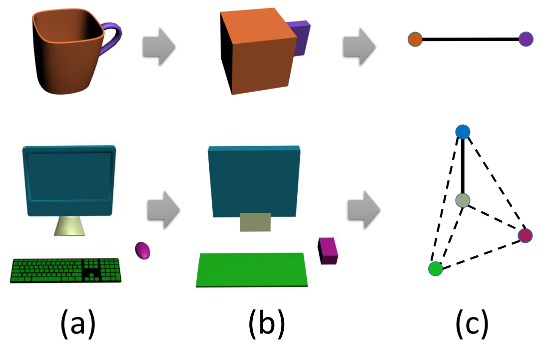

Figure 6: *The generation of object and object group. The input is a segmented object or object group (a). We compute the OBB for each part (b) and connect them into a graph (c). The solid and the dashed lines in (c) are connection and proximity edge, respectively.*

Multiple Kernel Learning. Kernel method has been successfully applied into many learning areas, while the results of these methods are heavily dependent on the selection of kernels. Instead of choosing a single kernel, it is better to have a set of kernels and use the combination of them [11]. Since our features are computed for both individual objects and their relations, it is especially desirable to combine several kernels and to allow the classifiers to choose their optimized kernels, in order to reduce their bias [34]. The idea is to use a combination of basic kernels $k(\mathbf{x}, \mathbf{y}) = \Sigma w_i \cdot k_i(\mathbf{x}, \mathbf{y})$ rather than a specific kernel in SVM. The basic kernels could be linear kernel, Gaussian kernel, polynomial kernel, etc.

Figure 7 illustrates the architecture of our MKL-based classification. Given the segment graph of an individual object or an object group, we first represent it in the feature space spanned with six kinds of features. We then transform the data from feature space to kernel space using several predefined kernels. By computing the optimized weights for each kernel space, we obtain the final MKL classifier. The procedure for training the classifiers is detailed in Algorithm 2.

5.2. Testing

Data Preprocessing. The segments in scenes acquired by Kinect or any other commodity depth camera are usually noisy and low-quality, making the recognition quite difficult. Therefore, we first surface reconstruction [35] to form a watertight surface for each segment, and then compute features as described in Sec. 5.1.

Algorithm 3 :Testing.

Input: classifiers C and segments T
Output: segments label X

for all c_i in C **do**
  if Matching(c_i,T) **then**
    $cost_i \leftarrow$ ComputeProbability(c_i,T);
  end if
end for;
$X \leftarrow$ ComputeLabel($\{cost_i\}_{i=1}^{k}$);
//compute label for all segments
return X;

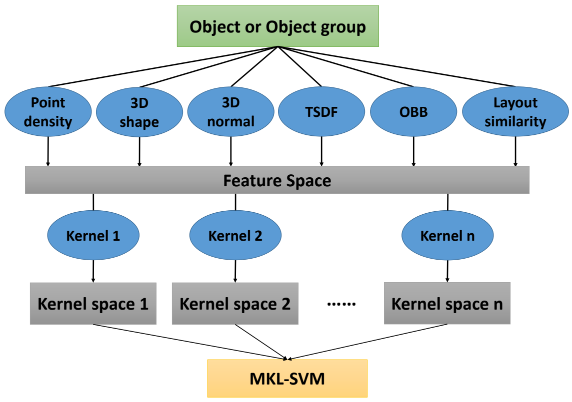

Figure 7: *The architecture of our MKL-based classifier. Given an object or a object group, we compute its features and map it into several kernel spaces with several basic kernels. The MKL-SVM classifier is learned by computing the optimized weight for each kernel.*

Labeling Optimization. To extract objects and object groups from the segment graph, we search from the segment graph of the input scene for the subgraphs corresponding to the occurrences of database objects and object groups. Graph matching can be formulated as quadratic assignment problem, which is known to be NP-hard, so an exhaustive search over the whole graph leads to high computational cost.

In our method, the graph matching is performed as follows. For each MKL classifier, we first use the associated triplet (n_n, n_c, n_p) to filter subgraph matchings. A subgraph is filtered if any one of the three terms is different from that of the classifier. For the remaining subgraphs, we use the learned MKL classifiers to test if it belongs to the corresponding category and record the probability if yes. The probability will be used as the labeling cost which penalizes the mislabeling in the following optimization.

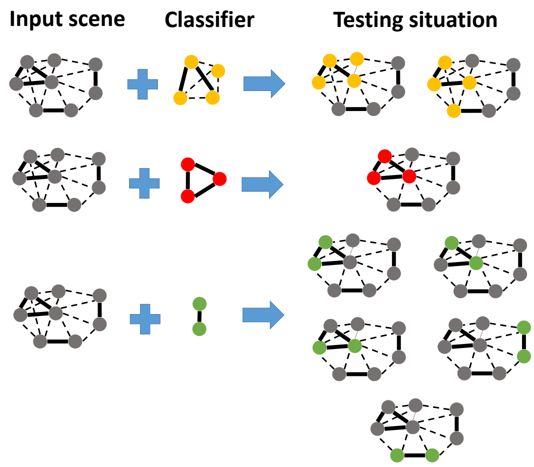

Figure 8: *The matching strategy of our algorithm. Given a segment graph of the input scene on the left, we use all the three classifiers to test the occurrence of the corresponding subgraph. The testing samples are shown on the right. Note that some connection edge in the first row can be turned into a proximity one to allow more matches.*

After applying all classifiers, we detect all the potential objects or object groups in the input scene. The graph matching strategy is illustrated in Figure 8. Note that we allow a connection edge to be converted into a proximity one to produce more matchings. The rationale of this is that some segments not belonging to the same object could be linked by connection edges mistakenly due to small mutual distance.

Next, we solve a labeling optimization which minimizes the overall matching cost computed from all the matching probability. The final labeling, X, for all segments of the input scene is computed by:

$$X = \text{argmin}_X \sum_{c_i \in C} D(X, c_i) \qquad (10)$$

where:

$$D(X, c_i) = \begin{cases} 0, & \text{if recognized subgraph by } c_i \text{ is labeled correctly in } X \\ \text{cost}(X, c_i), & \text{otherwise.} \end{cases}$$

where $\text{cost}(X, c_i)$ is the labeling cost penalizing the wrong labeling of the subgraph detected by the classifier c_i. We found it suffices to solve this labeling optimization using a combinatorial search over all labeling possibilities since the possible labeling for each segment is limited after the classifier filtering and testing. The

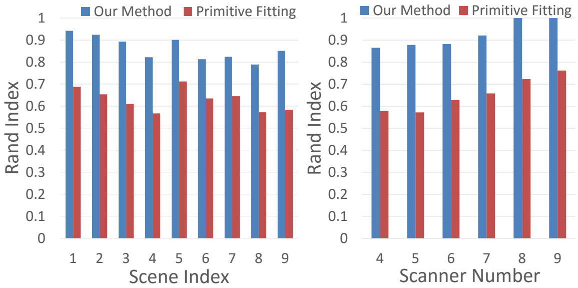

Figure 9: *Segmentation comparison against the RANSAC based primitive fitting method [15]. Left: Comparison over nine test scenes. Right: Results of our method and the RANSAC-based one over scene #2 with increasing number of scans.*

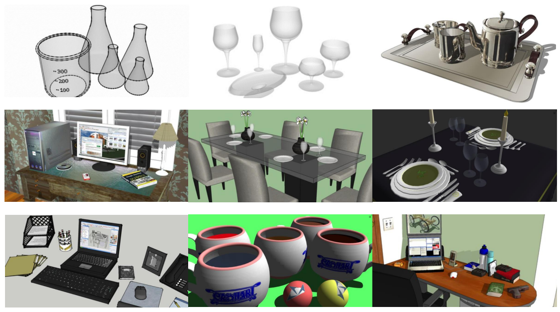

Figure 10: *The test scenes used in segmentation evaluation.*

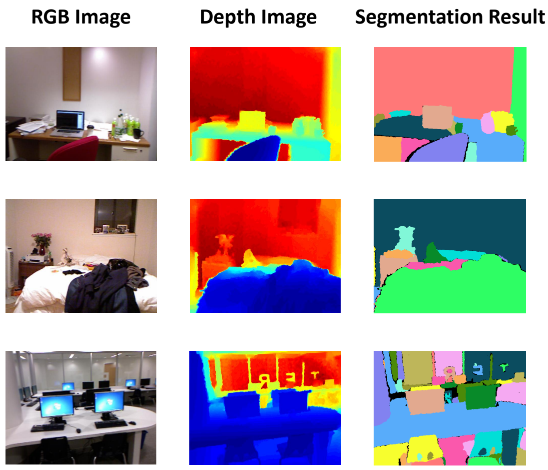

Figure 11: *The segmentation results our algorithm over the scenes from the NYU-Depth V2 dataset. Our method can segment most objects correctly in the highly cluttered scenes.*

whole testing process for object and object group detection is described in Algorithm 3.

6. Results and Evaluation

We test our method on both real-world and virtually scanned scenes. A gallery of results is shown in Figure 18. We first describe the experimental setting of our method and then evaluate our method in two aspects, i.e., the segment detection and the object recognition.

Experimental Setting. Our method is implemented using C++ and run on a desktop PC with an Intel I5-3750 CPU (quad core, 3.4GHz) and Nvidia GeForce GTX 460 graphics card. We scan a few indoor scenes using a Microsoft Kinect. We also use the Washington scene dataset [36] acquired by an ASUS Xtion PRO LIVE RGB-D sensor. The parameter settings are provided below. Patch size (diameter): 8cm for NYU-Depth V2 dataset and 4cm for others; w_c, w_p, and w_n in 1: 0.2, 0.3, and 0.5, respectively; δ for table extraction: 0.95 for all datasets; t_s and t_l for segment graph construction: 3cm and 50cm, respectively; Poisson iso-point sampling density: 2cm; basic kernels for MKL (we use SimpleMKL [37]): five Gaussian kernels and two polynomial kernels.

Segment Detection. We test our segment detection algorithm on nine tabletop scenes downloaded from the Internet (Figure 10) and virtually scanned. We compare our method with the RANSAC-based primitive fitting method in [15]. The Rand Index [38] is used as the evaluation criterion. We perform six tests on each scene with different number of scan and quality and take the average Rand Index. In the virtual scanning, the virtual scanners are positioned around the scene being scanned and oriented to the center of the scene. The plot in Figure 9 (left) show that the Rand Index of our method is higher than that of the RANSAC-based method over the nine test scenes. We also evaluate how scan quality would affect the segmentation results with the varying number of scans for scene #2; see Figure 9 (right).

We also test our segmentation approach on NYU-Depth V2 dataset. A significant feature of the depth images is that the point cloud is of low resolution, making our segmentation infeasible. In order to tackle this kind of input, we made some changes over our algorithm. Given an RGB-D image and its camera parameters, we first

	Primitive fitting[16]	Support relation[9]	Our method
Rand Index	61.8%	78.7%	76.4%

Figure 12: *A comparison of the segmentation accuracy (Rand Index) of the methods in [15] and [8] and ours on the NYU-Depth V2 dataset.*

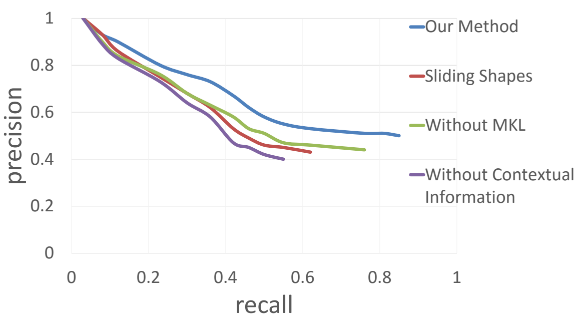

Figure 13: *Precision-recall curves for object recognition. Comparison is made between our method and the other three methods by testing on the database in [36].*

project the 2D points into 3D space to reconstruct a 3D scene. We skip the table extraction process and detect the segments for the near-camera points (distance less than 2m) using our method, and cluster the rest distant points using Euclidean cluster extraction [39].

We test our method on a selected subset of the NYU-Depth V2 dataset as in [22], which contains 45 living rooms and offices. Some results of our algorithm are shown in Figure 11. We compare our method with the support relation based method in [8] and the RANSAC-based one in [15]. The segmentation Rand Index measures for the three methods are shown in Figure 12. The support relation based method slightly outperforms our method, due to the incorporation of the high-level prior.

Object Recognition. Our recognition database contains 900 objects in 18 categories and 10 kinds of object groups. We test our object recognition method on two scanned scene datasets. The first one is several real-world scenes such as office, meeting room, and laboratory, scanned by ourselves and the second dataset from [36]. The scenes contain a variety of object categories with noisy and low quality scans.

Figure 18 demonstrates the results on six indoor scenes. The semantic labels are shown using distinct colors, while the contextual information is illustrated with red dots and dashed lines. The majority of objects can be recognized correctly, benefiting from the contextual information. The geometric ambiguity between different categories of objects, such a keyboard and a book, are resolved with the help of contextual information. Some segments are correctly segmented but not successfully recognized due to capability of our recognition model learned from the limited model database. This can be improved by collecting more data and training a more powerful model.

We evaluate our method on the database of [36] containing 58 indoor scenes collected using KinectFusion. We compare to three alternative methods: the sliding shapes [21], a reduced version of our method by using linear SVM classifiers, and a reduced method without using contextual information. The precision-recall curves for recognition are plotted in Figure 13. It is obvious that our method outperforms sliding shapes, thanks to the object-group-level analysis and the MKL classifiers in our method. The reduced method without contextual information is slightly inferior to sliding shapes. This is because sliding shapes use a plethora of classifiers, which is three orders of magnitude more than what our method uses.

As demonstrated in Figure 14, our method benefits from the contextual information in two ways. First, context helps to eliminate recognition ambiguity. For example, the object in Figure 14 (a) can either be a book or a keyboard, which is correctly recognized with the help of the monitor-keyboard-mouse combo. Second, context can enhance the recognition ability under low data quality. For example, the cup in Figure 14 (b) is hard to be recognized due to the low data quality, where the cup-cup group helps recognize it.

We make two observations from the results. (1) The precision is consistently high with the increasing of the recall. (2) The recall converges to a high value but never reaches 1 with the precision decreasing. These observations can be explained by the inter-restriction of the multiple MKL classifiers. Our method finds a labeling that tries to satisfy all the MKL detectors as much as possible, leading to more reliable labeling result.

To evaluate the performance our method on cluttered scenes, we scan six desktop scenes with an increasing degree of object clutter. The objects we recognized are highlighted with boxes in Figure 15. It is clear that our method achieves robust recognition on these cluttered scenes, especially the one in Figure 15 (c). As a comparison, the method in [22] cannot recognize the mouse in (c), because the contour-based approach fails when

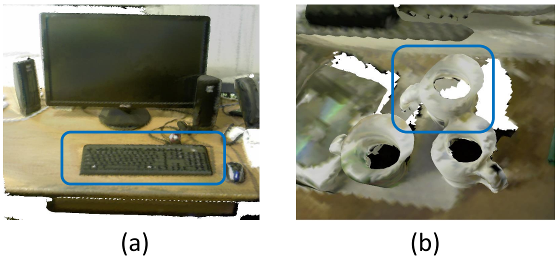

Figure 14: *The contextual knowledge could benefit object recognition in two ways. (a): Resolving recognition ambiguity: The keyboard in blue box is recognized correctly due to the contextual information of the monitor-keyboard-mouse combo. (b): Enhancing recognition ability: The cup in blue box is in low scan quality but can be recognized based on the cup-cup combo.*

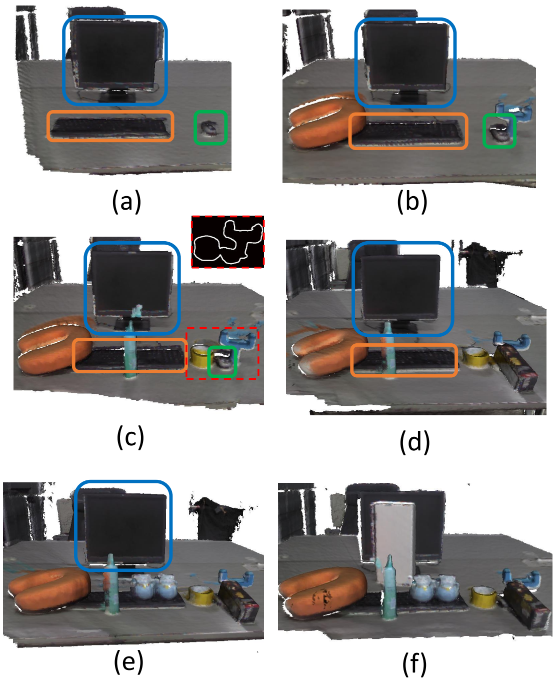

Figure 15: *Our recognition results on several scenes with increasing degree of object from (a) to (f). The monitors, keyboards and mouses are correctly recognized by our method and labeled with blue, orange and green boxes.*

dealing with cluttered scenes due to the incorrect contour extraction. The contour of the red box area in (c) is shown on the top-right corner.

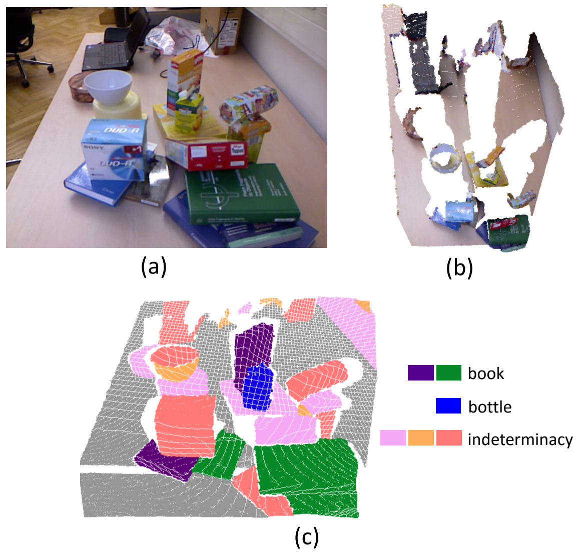

Figure 16: *A failure case of our method. Our method cannot recognize most of the objects in a cluttered scene (c). This is due to the fact that the scene point cloud is only a single-view scan (b).*

Time Performance. For a scene with 100K points, the segment detection takes 20 seconds. The training procedure of our object recognition is determined by the number of individual object and object group categories. In our case, it takes about 1 hour to train a classifier using SimpleMKL averagely. The training process takes about 32 hours in total for the 18 objects and the 10 object groups. The testing time is determined by the number of segments and the degree of object clutter. The testing time for the scenes in Figure 18 (a) to (f) are 7.8, 19.1, 39.5, 20.3, 1.7 and 12.9 minutes, respectively.

Limitations. Our method has the following limitations. First, our method does not provide a mechanism to deal with input data with severe missing parts. For example, if the input contains only a single-view scan, our method would not be able to produce meaningful segments for further analysis. A failure case of this is shown in Figure 16. Second, our method can tolerate only moderate shape variation. It might fail when recognizing objects with too special structure of segment graph, such as the case shown in Figure 17. Last, our method works the best for a scene containing a planar support. Although quite commonly seen in everyday indoor environments, the assumption does not generalize well for outdoor scenes.

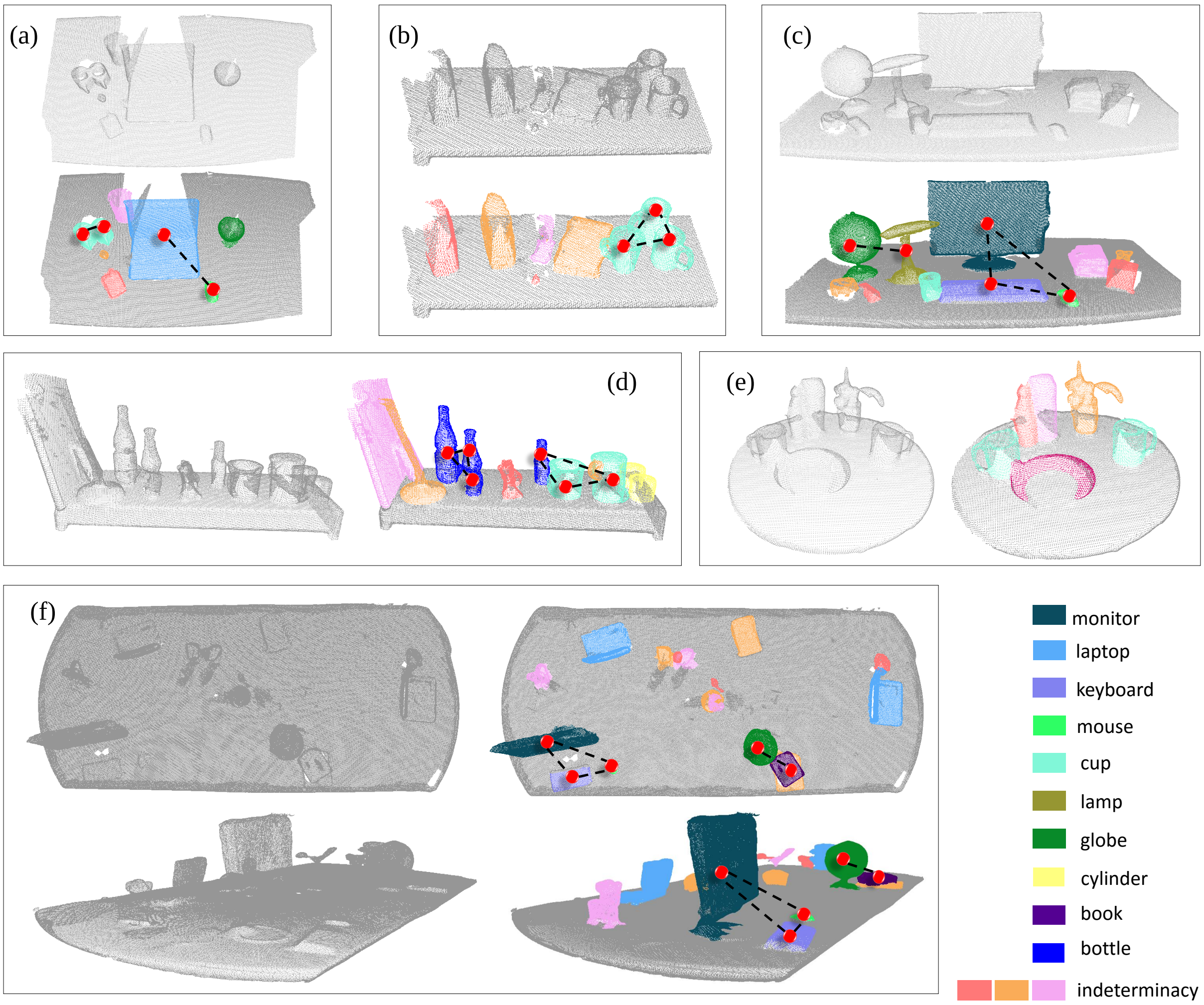

Figure 18: *A gallery of scene understanding results by our method.*

7. Discussion and future work

To achieve object analysis from clustered subscenes, we have developed a unified framework for the discovery of both individual objects and object groups, both of which are based on the contextual information learned from a database of 3D scene models. Our method makes the contextual information applicable even without knowing the object segmentation of the input scene. The latter has so far been predominantly assumed by existing methods, e.g., [22].

We see three venues for future work. First, our current work focuses on subscene analysis. It would be interesting to extend our method to deal with whole scene, leading to multi-scale scene analysis in a unified framework. Currently, the contextual information is based on spatial proximity. As another future work, we would like to expand our contextual features with multi-modal object interaction, such as dynamic motion, to address more complex mutual relations among objects. Finally, it is natural to utilize our framework in robot-operated autonomous scene scanning and understanding.

Acknowledgements

We thank all the reviewers for their comments and feedback. We would also like to acknowledge our research grants: NSFC (61572507, 61202333, 61379103),

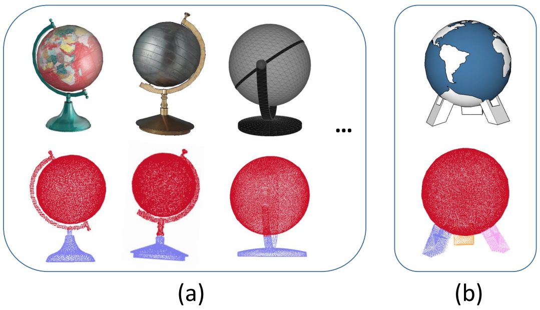

Figure 17: *The object classifier for a globe is trained using the examples containing two components (a). The recognition may fail when testing an exceptional instance of globe with three legs (b).*

973 Program (2014CB360503), Guangdong Science and Technology Program (2015A030312015, 2014B050502009, 2014TX01X033), Shenzhen VisuCA Key Lab (CXB201104220029A).

References

[1] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, B. Chen, Autoscanning for coupled scene reconstruction and proactive object analysis, ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 34 (6) (2015) 177:1–177:14.

[2] L. Nan, K. Xie, A. Sharf, A search-classify approach for cluttered indoor scene understanding, ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 31 (6) (2012) 137:1–137:10.

[3] Y. M. Kim, N. J. Mitra, D.-M. Yan, L. Guibas, Acquiring 3d indoor environments with variability and repetition, ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 31 (6) (2012) 138:1–138:11.

[4] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, B. Guo, An interactive approach to semantic modeling of indoor scenes with an RGBD camera, ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 31 (6) (2012) 136:1–136:11.

[5] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, R. Pajarola, Object detection and classification from large-scale cluttered indoor scans, Computer Graphics Forum (Proc. of Eurographics) 33 (2) 11–21.

[6] Y. Li, A. Dai, L. Guibas, M. Nießner, Database-assisted object retrieval for real-time 3d reconstruction, Computer Graphics Forum (Proc. of Eurographics) 34 (2) (2015) 435–446.

[7] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, A. Fitzgibbon, KinectFusion: Real-time dense surface mapping and tracking, in: Proc. IEEE Int. Symp. on Mixed and Augmented Reality, 2011, pp. 127–136.

[8] N. Silberman, P. Kohli, D. Hoiem, R. Fergus, Indoor segmentation and support inference from rgbd images, in: Proc. Euro. Conf. on Computer Vision, 2012, pp. 746–760.

[9] M. Fisher, M. Savva, P. Hanrahan, Characterizing structural relationships in scenes using graph kernels, ACM Trans. on Graphics (Proc. of SIGGRAPH) 30 (4) (2011) 34:1–34:11.

[10] K. Xu, K. Chen, H. Fu, W.-L. Sun, S.-M. Hu, Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models, ACM Trans. on Graphics (Proc. of SIGGRAPH) 32 (4) (2013) 123:1–123:10.

[11] M. Gönen, E. Alpaydın, Multiple kernel learning algorithms, Journal of Machine Learning Research 12 (2011) 2211–2268.

[12] A. Shamir, A survey on mesh segmentation techniques, Computer Graphics Forum 27 (6) (2008) 1539–1556.

[13] E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3d mesh segmentation and labeling, ACM Trans. Graph. 29 (2010) 102:1–102:12.

[14] M. Johnson-Roberson, J. Bohg, M. Björkman, D. Kragic, Attention-based active 3d point cloud segmentation., in: Proc. IEEE Int. Conf. on Intelligent Robots & Systems, 2010, pp. 1165–1170.

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

[16] A. Berner, M. Bokeloh, M. Wand, A. Schilling, H.-P. Seidel, A graph-based approach to symmetry detection., in: Volume Graphics, Vol. 40, 2008, pp. 1–8.

[17] A. Golovinskiy, T. Funkhouser, Min-cut based segmentation of point clouds, in: Proc. Int. Conf. on Computer Vision, IEEE, 2009, pp. 39–46.

[18] L. A. Alexandre, 3d descriptors for object and category recognition: a comparative evaluation, in: Proc. IEEE Int. Conf. on Intelligent Robots & Systems, Vol. 1, pp. 1–6.

[19] K. Lai, L. Bo, X. Ren, D. Fox, Rgb-d object recognition: Features, algorithms, and a large scale benchmark, in: Consumer Depth Cameras for Computer Vision, Springer, 2013, pp. 167–192.

[20] K. Xu, V. G. Kim, Q. Huang, E. Kalogerakis, Data-driven shape analysis and processing, Computer Graphics Forum (2015) to appear.

[21] S. Song, J. Xiao, Sliding shapes for 3d object detection in depth images, in: Proc. Euro. Conf. on Computer Vision, Springer, 2014, pp. 634–651.

[22] K. Chen, Y.-K. Lai, Y.-X. Wu, R. Martin, S.-M. Hu, Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information, ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 33 (6) (2014) 208:1–208:15.

[23] J. S. Rudolph Triebel, Roland Siegwart, Unsupervised discovery of repetitive objects, in: Proc. IEEE Int. Conf. on Robotics & Automation, 2010, pp. 5041 – 5046.

[24] Y. Zhang, W. Xu, Y. Tong, K. Zhou, Online structure analysis for real-time indoor scene reconstruction, ACM Trans. on Graphics 159:1–159:12.

[25] J. Biswas, M. Veloso, Planar polygon extraction and merging from depth images, in: Proc. IEEE Int. Conf. on Intelligent Robots & Systems, IEEE, 2012, pp. 3859–3864.

[26] M. Dou, L. Guan, J.-M. Frahm, H. Fuchs, Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held rgb-d camera, in: Computer Vision-ACCV 2012 Workshops, Springer, 2013, pp. 94–108.

[27] A. Monszpart, N. Mellado, G. Brostow, N. Mitra, RAPter: Rebuilding man-made scenes with regular arrangements of planes 34 (2015) 103:1–103:12.

[28] J. Papon, A. Abramov, M. Schoeler, F. Worgotter, Voxel cloud connectivity segmentation-supervoxels for point clouds, in: Proc. IEEE Conf. on Computer Vision & Pattern Recognition, IEEE, 2013, pp. 2027–2034.

[29] S. Katz, A. Tal, Hierarchical mesh decomposition using fuzzy clustering and cuts, ACM Trans. on Graphics (Proc. of SIGGRAPH) 22 (3) (2003) 954–961.

[30] A. Golovinskiy, V. G. Kim, T. A. Funkhouser, Shape-based

recognition of 3d point clouds in urban environments, in: Proc. Int. Conf. on Computer Vision, 2009, pp. 2154–2161.

[31] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Analysis & Machine Intelligence 17 (8) (1995) 790–799.

[32] N. J. Mitra, M. Wand, H. Zhang, D. Cohen-Or, V. Kim, Q.-X. Huang, Structure-aware shape processing, in: ACM SIGGRAPH 2014 Courses, 2014.

[33] K. Xu, R. Ma, H. Zhang, C. Zhu, A. Shamir, D. Cohen-Or, H. Huang, Organizing heterogeneous scene collection through contextual focal points, ACM Trans. on Graphics (Proc. of SIGGRAPH) 33 (4) (2014) 35:1–35:12.

[34] C. Zhu, X. Liu, Q. Liu, Y. Ming, J. Yin, Distance based multiple kernel elm: A fast multiple kernel learning approach, Mathematical Problems in Engineering 2015.

[35] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Trans. on Graphics 32 (3) (2013) 29:1–29:13.

[36] A. Karpathy, S. Miller, L. Fei-Fei, Object discovery in 3d scenes via shape analysis, in: Proc. IEEE Int. Conf. on Robotics & Automation, IEEE, 2013, pp. 2088–2095.

[37] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, SimpleMKL, Journal of Machine Learning Research 9 (2008) 2491–2521.

[38] J. Chen, D. Bautembach, S. Izadi, Scalable real-time volumetric surface reconstruction, ACM Trans. on Graphics (Proc. of SIGGRAPH) 32 (4) (2013) 113:1–113:16.

[39] D. Sparks, Euclidean cluster analysis, Applied Statistics (1973) 126–130.