A Robust 3D-2D Interactive Tool for Scene Segmentation and Annotation

Duc Thanh Nguyen, Binh-Son Hua*, Lap-Fai Yu, *Member, IEEE,* and Sai-Kit Yeung, *Member, IEEE*

Abstract—Recent advances of 3D acquisition devices have enabled large-scale acquisition of 3D scene data. Such data, if completely and well annotated, can serve as useful ingredients for a wide spectrum of computer vision and graphics works such as data-driven modeling and scene understanding, object detection and recognition. However, annotating a vast amount of 3D scene data remains challenging due to the lack of an effective tool and/or the complexity of 3D scenes (e.g. clutter, varying illumination conditions). This paper aims to build a robust annotation tool that effectively and conveniently enables the segmentation and annotation of massive 3D data. Our tool works by coupling 2D and 3D information via an interactive framework, through which users can provide high-level semantic annotation for objects. We have experimented our tool and found that a typical indoor scene could be well segmented and annotated in less than 30 minutes by using the tool, as opposed to a few hours if done manually. Along with the tool, we created a dataset of over a hundred 3D scenes associated with complete annotations using our tool. Both the tool and dataset are available at http://scenenn.net.

Index Terms—Annotation tool, semantic annotation, 3D segmentation, 3D reconstruction, 2D-3D interactive framework

1 Introduction

High-quality 3D scene data has become increasingly available thanks to the growing popularity of consumer-grade depth sensors and tremendous progress in 3D scene reconstruction research [1], [2], [3], [4], [5], [6]. Such 3D data, if fully and well annotated, would be useful for powering different computer vision and graphics tasks such as scene understanding [7], [8], object detection and recognition [9], and functionality reasoning in 3D space [10].

Scene segmentation and annotation refer to separating an input scene into meaningful objects. For example, the scene in Fig. 1 can be segmented and annotated into chairs, table, etc. Literature has shown the crucial role of 2D annotation tools (e.g. [11]) and 2D image datasets (e.g. [12], [13], [14]) various tasks like semantic segmentation, object detection and recognition [15], [16]. This inspires us for such tasks on 3D scene data. However, segmentation and annotation of 3D scenes require much more effort due to the large scale of the 3D data (e.g. there are millions of 3D points in a reconstructed scene). Development of a robust tool to facilitate the segmentation and annotation of 3D scenes thus is a demand and also the aim of this work. To this end, we make the following contributions:

- We propose an interactive framework that effectively couples the geometric and appearance information from multi-view RGB data. The framework is able to automatically perform 3D scene segmentation.
- Our tool is facilitated with a 2D segmentation algorithm based on 3D segmentation.
- We develop assistive user-interactive operations that allow users to flexibly manipulate scenes and objects in both 3D and 2D. Users co-operate with the tool by refining the segmentation and providing semantic annotation.
- To further assist users in annotation, we propose an object search algorithm which automatically segments and annotates repetitive objects defined by users.
- We create a dataset including over a hundred scenes. All the scenes are fully segmented and annotated using our tool. This dataset will serve as a benchmark for future works in 3D computer vision and graphics.

Compared with existing works on RGB-D segmentation and annotation (e.g. [17], [18]), our tool holds several advantages. First, segmentation and annotation are centralized in 3D and thus free users from manipulating thousands of images. Second, the tool can adapt with either RGB-D images or scene triangular meshes as the input. This enables the tool to handle meshes reconstructed from either RGB-D images [19] or structure-from-motion [20] in a unified framework.

We note that interactive annotation has also been exploited in a few concurrent works, e.g. SemanticPaint in [21] and Semantic Paintbrush in [22]. Compared with those systems, our annotation tool offers a wider range of interactions. In addition, the tool also provides more assistive functionalities, e.g. 3D object search, 2D segmentation.

2 Related Work

RGB-D Segmentation. A common approach for scene segmentation is to perform the segmentation on RGB/D images. Examples of this approach can be found in [17], [18], [23]. The spatial relationships between objects can also be exploited to infer the scene labels. For example, Jia et al. [24] used

* *Co-first author*

- *Duc Thanh Nguyen is with the School of Information Technology, Deakin University, Australia. E-mails: duc.nguyen@deakin.edu.au*
- *Lap-Fai Yu is with the University of Massachusetts Boston. E-mail: craigyu@cs.umb.edu*
- *Binh-Son Hua and Sai-Kit Yeung are with Information Systems Technology and Design Pillar, Singapore University of Technology and Design. E-mail: {binhson_hua,saikit}@sutd.edu.sg*

Fig. 1. A reconstructed 3D scene segmented and annotated using our tool.

object layout rules for scene labeling. The spatial relationship between objects was modeled by a conditional random field (CRF) in [25], [26] and directed graph in [27].

In general, the above methods segment RGB/D images captured from single viewpoints of a 3D scene individually while the segmentation of an individual image may not be reliable and incomplete. Compared with those methods, our tool can achieve more accurate and complete segmentation results with the 3D models of the scene and its objects.

From 2D to 3D. Labeling a 3D scene can be performed by back-projecting the labels obtained on the 2D images of that scene to 3D space. For example, Wang et al. [28] used the labels provided in ImageNet [12] to infer 3D labels. In [3], 2D labels were obtained by drawing polygons.

Labeling directly on images is time consuming. Typically, a few thousands of images need to be handled. It is possible to perform matching among the images to propagate the annotations from one image to another, e.g. [3], but this process is not reliable.

3D Object Templates. 3D object templates can be used to segment 3D scenes. The templates can be organized in holistic models, e.g., [29], [30], [31], [32], or part-based models, e.g. [33]. The segmentation can be performed on 3D point clouds, e.g. [29], [31], [33], or 3D patches, e.g. [32], [30], [34].

Generally speaking, the above techniques require the template models to be known in advance. They do not fit well our interactive system in which the templates can be provided on the fly by users. In our tool, we propose to use shape matching to help users in the segmentation and annotation task. Shape matching does not require off-line training and is proved to perform efficiently in practice [35].

Online Scene Understanding. Recently, there are methods that combine 3D reconstruction and annotation to achieve online scene understanding. For example, Tateno et al. [36] proposed to segment depth images and fuse the segmentations incrementally into a SLAM framework. SemanticPaint developed in [21] allowed users annotate a scene by touching objects. The SemanticPaint was extended to the Semantic Paintbrush [22] for outdoor scenes annotation by exploiting the farther range of a stereo rig.

In both [21] and [22], objects of interest were identified by touching and the corresponding object classes were modeled by CRFs. These methods used voxel-based TSDF representation rather than triangle mesh (as our approach) to represent objects and implicitly assumed that all objects of the same class have similar appearance (e.g. color). Since the CRFs were built upon the reconstructed data, there also assumed the reconstructed data was good enough. However, reconstructed scenes are often incomplete. To deal with this issue, we describe 3D objects using a shape descriptor which is robust to shape variation and occlusion. Experimental results show that our approach works well under noisy data (e.g. broken mesh) and robustly deal with shape deformation while being efficient for practical use.

3 System Overview

Fig. 2 shows the workflow of our tool. The tool includes four main stages: scene reconstruction, automatic 3D segmentation, interactive refinement and annotation, and 2D segmentation.

In the first stage (section 4), the system takes a sequence of RGB-D frames and reconstructs a triangular mesh, called *3D scene mesh*. We compute and cache the correspondences between the 3D vertices in the reconstructed scene and the 2D pixels on all input frames. This allows seamless switching between segmentation in 3D and 2D in later steps.

In the second stage (section 5), the 3D scene mesh is automatically segmented. We start by clustering the mesh vertices into supervertices (section 5.1). Next, we group the supervertices into regions (section 5.2). We also cache the results of both steps for later use.

The third stage (section 6) is designed for users to interact with the system. We design three segmentation refinement operations: *merge*, *extract*, and *split*. After refinement, users can make semantic annotation for objects in the scene.

To further assist users in segmentation and annotation of repetitive objects, we propose an algorithm to automatically search for repetitive objects specified by a template (section 7). We extend the well-known 2D shape context [35] to 3D space and apply shape matching to implement this functionality.

The fourth stage of the framework (section 8) is designed for segmentation of 2D frames. In this stage, we devise an algorithm based on contour matching that uses the segmentation results in 3D to initialize the 2D segmentation.

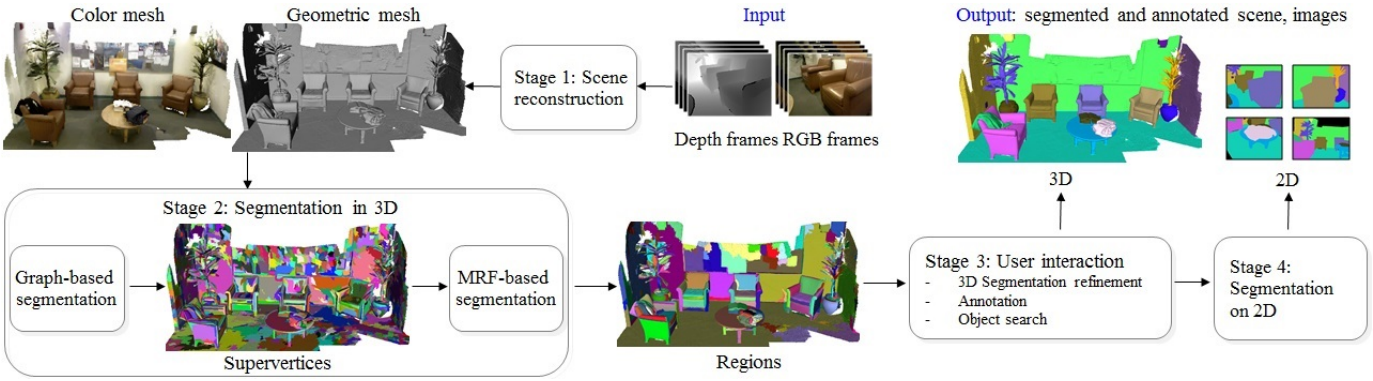


Fig. 2. Overview of our annotation tool.

4 Scene Reconstruction

4.1 Geometry reconstruction

Several techniques have been developed for 3D scene reconstruction. For example, KinectFusion [37] applied frame-to-model alignment to fuse depth information and visualize 3D scenes in real time. However, KinectFusion tends to cause drift where depth maps are not accurately aligned due to accumulation of registration errors over time. Several attempts have been made to avoid drift and led to significant improvements in high-quality 3D reconstruction. For example, Xiao et al. [3] added object constraints to correct misaligned reconstructions. In [38], [4], [39], the input frames were split into small chunks, each of which could be accurately reconstructed. An optimization was then performed to register all the chunks into the same coordinate frame. In many systems, e.g. [40], re-visiting places are used to trigger a loop closure constraint to enforce global consistency of camera poses. In [41], loop closure was enabled by splitting the scene into submaps. The information from every single image frame was accumulated into the submaps and the position and orientation of the submaps were then adjusted accordingly.

In our system, we adopt the method in [4], [19] to calculate camera poses. The triangular mesh then can be extracted using the marching cubes algorithm [42]. The normal of each mesh vertex is given by the area-weighted average over the normals of its neighbor surfaces. We further smooth the resulting normals using a bilateral filter.

4.2 3D-2D Correspondence

Given the reconstructed 3D scene, we align the whole sequence of 2D frames with the 3D scene using the corresponding camera poses obtained from section 4.1. For each 3D vertex, its normal is computed on the 3D mesh and color is estimated as the median of the color of the corresponding pixels on 2D frames.

5 Segmentation in 3D

After the reconstruction, a scene mesh typically consists of millions of vertices. In this stage, those vertices are segmented into much fewer regions. Directly segmenting millions of vertices is computationally expensive and requires a lot of computer resources. To avoid this, we perform a two-level segmentation. At the first level, we divide the reconstructed scene into a number of so-called supervertices by applying a purely geometry-based segmentation method. At the second level, we merge the supervertices into larger regions by considering both surface normals and colors.

5.1 Graph-based Segmentation

We extend the efficient graph-based image segmentation algorithm of Felzenszwalb et al. [43] to 3D space. We have also tried with normalized cuts [44] and found that graph-based segmentation worked more stably and faster. In addition, graph-based segmentation is often selected for scene segmentation, e.g. in [45]. However, we note that other existing superpixel methods [46] could also be considered for this task.

The graph-based segmentation algorithm operates as follows. Given the scene mesh, a graph is defined in which each node in the graph corresponds to a vertex in the mesh. Two nodes in the graph are linked by an edge if their two corresponding vertices in the mesh are the vertices of a triangle. Let $\mathbf{V} = \{\mathbf{v}_i\}$ be the set of vertices in the mesh. The edge connecting two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ is weighted as

$$w(\mathbf{v}_i, \mathbf{v}_j) = 1 - \mathbf{n}_i{}^\top \mathbf{n}_j, \tag{1}$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are the unit normals of $\mathbf{v}_i$ and $\mathbf{v}_j$ respectively.

The graph-based segmenter in [43] employs a number of parameters including a smoothing factor used for noise filtering (normals in our case), a threshold representing the contrast between adjacent regions, and the minimum size of segmented regions. In our implementation, those parameters were set to 0.5, 500, and 20 respectively. However, we also make those parameters available to users for customization.

The graph-based segmentation algorithm results in a set of supervertices $\mathcal{S} = \{s_i\}$. Each supervertex is a group of geometrically homogeneous vertices with similar surface normals. The bottom left image in Fig. 2 shows an example of the supervertices. More examples can be found in Fig. 9.

5.2 MRF-based Segmentation

The graph-based segmentation often produces a large number (e.g. few thousand) supervertices, which could require considerable effort for annotation. To reduce this burden, the

supervertices are clustered into regions via optimizing an MRF model. In particular, for each supervertex $s_i \in \mathcal{S}$, the color and normal of s_i, denoted as $\bar{\mathbf{c}}_i$ and $\bar{\mathbf{n}}_i$, are computed as the means of the color values and normals of all vertices $\mathbf{v} \in s_i$. Each supervertex $s_i \in \mathcal{S}$ is then represented by a node o_i in an MRF. Two nodes o_i and o_j are directly connected if s_i and s_j share some common boundary (i.e. s_i and s_j are adjacent supervertices). Letting l_i be the label of o_i, the unary potentials are defined as

$$\psi_1(o_i, l_i) = -\log \mathcal{G}_i^c(\bar{\mathbf{c}}_i, \boldsymbol{\mu}_{l_i}^c, \boldsymbol{\Sigma}_{l_i}^c) - \log \mathcal{G}_i^n(\bar{\mathbf{n}}_i, \boldsymbol{\mu}_{l_i}^n, \boldsymbol{\Sigma}_{l_i}^n), \quad (2)$$

where $\mathcal{G}_{l_i}^c$ and $\mathcal{G}_{l_i}^n$ are the Gaussians of the color values and normals of the label class of l_i, and $\boldsymbol{\mu}_{l_i}^c / \boldsymbol{\mu}_{l_i}^n$ and $\boldsymbol{\Sigma}_{l_i}^c / \boldsymbol{\Sigma}_{l_i}^n$ are the mean and covariance matrix of $\mathcal{G}_{l_i}^c / \mathcal{G}_{l_i}^n$. The means and covariance matrices are computed based on the current labels and updated accordingly during the labeling process.

For the pairwise potentials, we use the Potts model [47]

$$\psi_2(l_i, l_j) = \begin{cases} -1, & \text{if } l_i = l_j \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

Let $\mathcal{L} = \{l_1, l_2, ..., l_{|\mathcal{S}|}\}$ be a labeling of the supervertices. The optimal labeling $\mathcal{L}^*$ is determined by

$$\mathcal{L}^* = \arg\min_{\mathcal{L}} \left[\sum_i \psi_1(o_i, l_i) + \gamma \sum_{i,j} \psi_2(l_i, l_j) \right] \quad (4)$$

where γ is weight factor that is set to 0.5 in our implementation.

The optimization problem in (4) is solved using the method in [47]. In our implementation, the number of labels was initialized to the number of supervertices; each supervertex was assigned to a different label. Fig. 2 (bottom) shows the result of the MRF-based segmentation. More results of this step are presented in Fig. 9.

6 Segmentation Refinement and Annotation in 3D

The automatic segmentation stage can produce over- and under-segmented regions. To resolve these issues, we design three operations: *merge*, *extract*, and *split*.

Merge. This operation is used to resolve over-segmentation. In particular, users identify over-segmented regions that need to be grouped by stroking on them. The merge operation is illustrated in the first row of Fig. 3.

Extract. This operation is designed to handle under-segmentation. In particular, for an under-segmented region, the supervertices composing that region can be retrieved. Users can select a few of those supervertices and use the merge operation to group them to create a new region. The second row of Fig. 3 shows the extract operation.

Split. In a few rare cases, the MRF-based segmentation may perform differently on different regions. This is probably because of the variation of the shape and appearance of objects. For example, a scene may have chairs appearing in a unique color and other chairs each of which composes multiple colors. Therefore, a unique setting of the parameters in the MRF-based segmentation may not adapt to all objects.

To address this issue, we design a split operation enabling user-guided MRF-based segmentation. Specifically, users first select an under-segmented region by stroking on that region. The MRF-based segmentation is then invoked on the selected region with a small value of γ (see (4)) to generate more grained regions. We then enforce a constraint such that the starting and ending point of the stroke belong to two different labels. For example, assume that l_i and l_j are the labels of two supervertices that respectively contain the starting and ending point of the stroke. To bias the objective function in (4), $\psi_2(l_i, l_j)$ in (3) is set to -1 when $l_i \neq l_j$, and to a large value (e.g. 10^9) otherwise. By doing so, the optimization in (4) would favor the case $l_i \neq l_j$. In other words, the supervertices at the starting and ending point are driven to separate regions. Note that the MRF-based segmentation is only re-executed on the selected region. Therefore, the split operation is fast and does not hinder user interaction. In particular, we have empirically found that the split operation takes much less than a second to process a single segment. We also note that the split operation takes into account the regions users select to re-run the MRF based optimization (with some implied prior) and hence would be less cumbersome than manually selecting supervertices as in the extract operation. The third row of Fig. 3 illustrates the split operation.

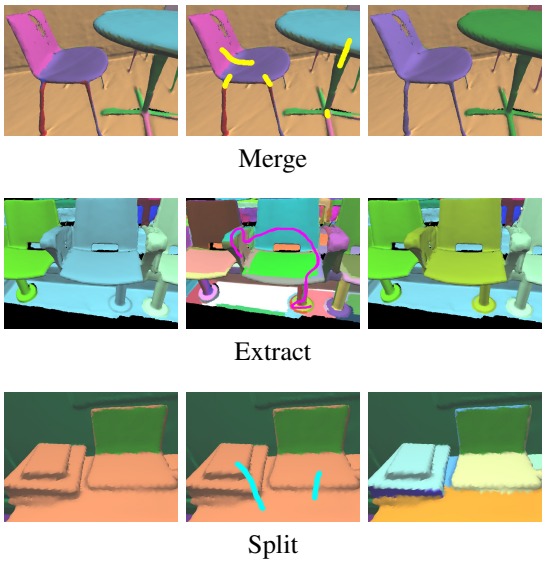


Fig. 3. Segmentation refinement operations. The leftmost and rightmost images illustrate the segmentation before and after applying operations. The center images illustrate the strokes. In the second row, the cyan region (in the left) is under-segmented and refined using Extract operation.

As shown in our user study (see Appendix), users mostly perform merge and extract operations. Split operation is only used when extract operation is not able to handle severe under-segmentations but such cases are not common in practice. When all the 3D segmented regions have been refined, users can annotate the regions by providing the object type. Fig. 4 shows an example of using our tool for annotation.

7 Object Search

There may exist multiple instances of an object class in a scene, e.g. the nine chairs in Fig. 5. To support labeling and

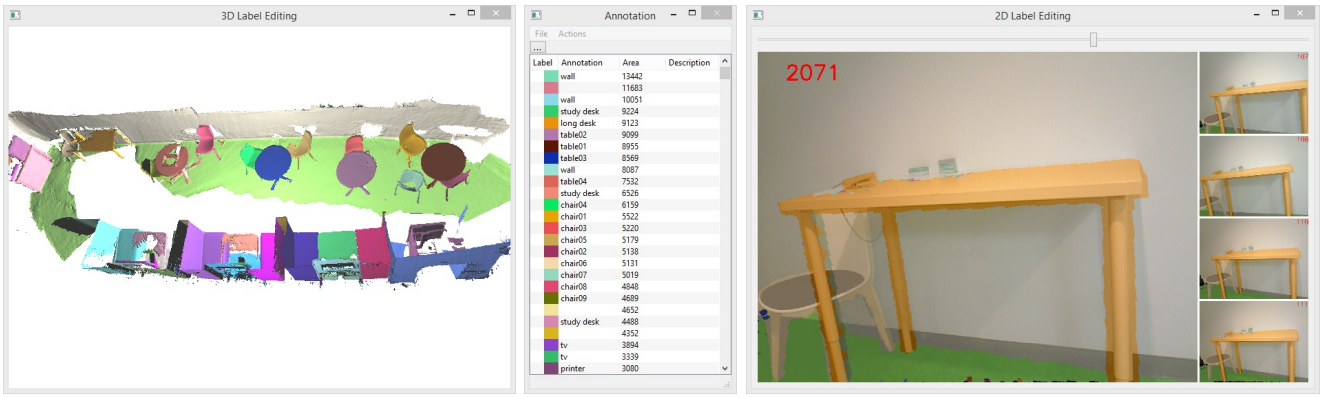


Fig. 4. Scene annotation using our tool. From left to right: 3D view, annotated labels, and 2D view.

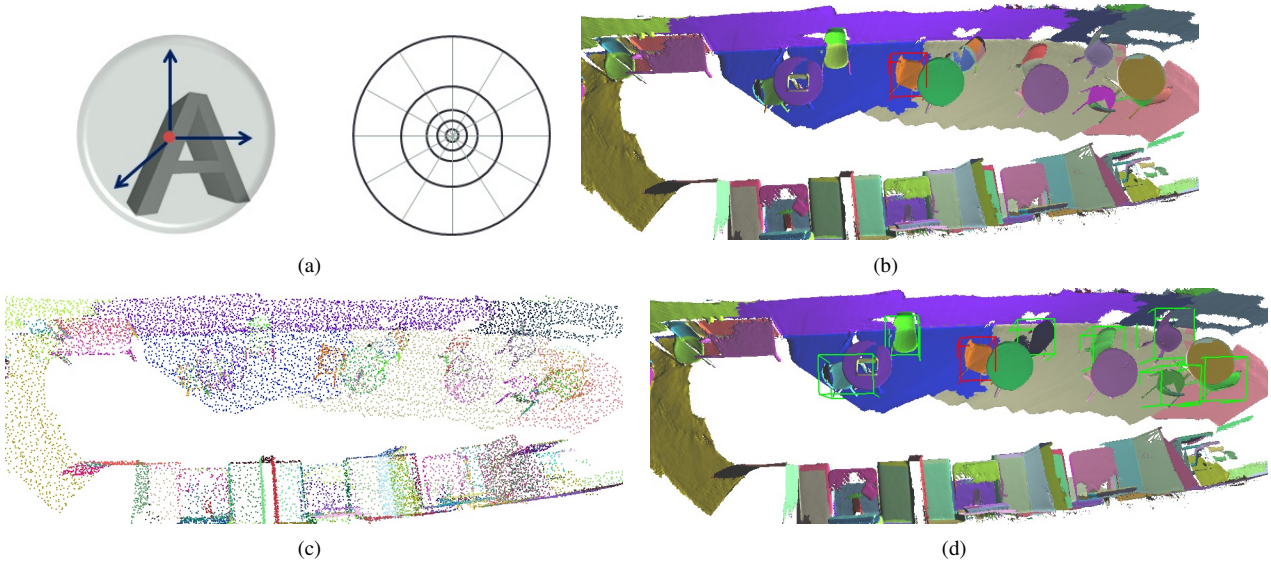


Fig. 5. (a) 3D shape context descriptor. Left: the shape context of a point (in red) can be represented as the spatial distribution in a sphere centered at that point. Right: a 2D view of the sphere. (b) The template (a chair) enclosed by a red box. Each remaining chair consists of multiple regions. (c) The down-sampled point cloud has 20,000 points. (d) Result of the object search: matching objects are found and enclosed by green boxes.

annotating repetitive objects, users can define a template by selecting an existing region or multiple regions composing the template. Those regions are the results of the MRF-based segmentation or user refinement. Given the user-defined template, our system automatically searches for objects that are similar to the template. Note that each repetitive object may be composed of multiple regions. For example, each chair in Fig. 5(a) consists of different regions such as the back, seat, legs. Once a group of regions is found to match well with the template, the regions are merged into a single object and recommended to users for verification. This is because the object search operation may propose inappropriate objects. Note that the object search operation is applied only to unlabeled regions. In our implementation, we extend the 2D shape context proposed in [35] to describe 3D objects (section 7.1). Matching objects with the template is performed via comparing shape context descriptors (section 7.2). The object search is then built upon the sliding-window object detection approach [48] (section 7.3).

7.1 Shape Context

Shape context was proposed by Belongie et al. [35] as a 2D shape descriptor and is well-known for many desirable properties such as being discriminative and robust to noise, shape deformation and transformation, and partial occlusions. Those properties fit our need well for several reasons. First, reconstructed scene meshes could be incomplete and contain noisy surfaces. Second, occlusions may also appear due to the lack of sufficient images completely covering objects. Third, the tool is expected to adapt to the variation of object shapes, e.g. chairs with and without arms.

In our work, a 3D object is represented by a set $\mathcal{V}$ of vertices obtained from the 3D reconstruction step. For each vertex $\mathbf{v}_i \in \mathcal{V}$, the shape context of $\mathbf{v}_i$ is denoted as $\mathbf{s}(\mathbf{v}_i)$ and represented by the histogram of the relative locations of other vertices $\mathbf{v}_j$, $j \neq i$, to $\mathbf{v}_i$. Let $\mathbf{u}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. The relative location of a vertex $\mathbf{v}_j \in \mathcal{V}$ to $\mathbf{v}_i$ is encoded by the length $\|\mathbf{u}_{ij}\|$ and the spherical coordinate $(\theta, \phi)_{ij}$ of $\mathbf{u}_{ij}$. In our implementation, the lengths $\|\mathbf{u}_{ij}\|$ were quantized into

5 levels. To make the shape context $\mathbf{s}(\mathbf{v}_i)$ more sensitive to local deformations, $\|\mathbf{u}_{ij}\|$ were quantized in a log-scale space. The spherical angles $(\theta, \phi)_{ij}$ were quantized uniformly into 6 discrete values. Fig. 5(a) illustrates the 3D shape context.

The shape context descriptor is endowed with scale invariance by normalizing $\|\mathbf{u}_{ij}\|$ by the mean of the lengths of all vectors. To make the shape context rotation invariant, Kortgen et al. [49] computed the spherical coordinates $(\theta, \phi)_{ij}$ relative to the eigenvectors of the covariance matrix of all vertices. However, the eigenvectors may not be computed reliably for shapes having no dominant orientations, e.g. rounded objects. In addition, the eigenvectors are only informative when the shape is complete while our scene meshes may be incomplete. To overcome this issue, we establish a local coordinate frame at each vertex on a shape using its normal and tangent vector. The tangent vector of a vertex $\mathbf{v}_i$ is the one connecting $\mathbf{v}_i$ to the centroid of the shape. We found that this approach worked more reliably.

Since a reconstructed scene often contains millions of vertices, prior to applying the object search, we uniformly sample $20,000$ points from the scene, which results in objects of $100-200$ vertices.

7.2 Shape Matching

Comparing (matching) two given shapes $\mathcal{V}$ and $\mathcal{Y}$ is to maximize the correspondences between pairs of vertices on these two shapes, i.e. minimizing the deformation of the two shapes in a point-wise fashion. We define the deformation cost between two vertices $\mathbf{v}_i \in \mathcal{V}$ and $\mathbf{y}_j \in \mathcal{Y}$ to be the $\chi^2(\mathbf{s}(\mathbf{v}_i), \mathbf{s}(\mathbf{y}_j))$ distance between the two corresponding shape context descriptors extracted at $\mathbf{v}_i$ and $\mathbf{y}_j$ as follow,

$$\chi^2(\mathbf{s}(\mathbf{v}_i), \mathbf{s}(\mathbf{y}_j)) = \frac{1}{2} \sum_{b=1}^{\dim(\mathbf{s}(\mathbf{v}_i))} \frac{(\mathbf{s}(\mathbf{v}_i)[b] - \mathbf{s}(\mathbf{y}_j)[b])^2}{\mathbf{s}(\mathbf{v}_i)[b] + \mathbf{s}(\mathbf{y}_j)[b]} \quad (5)$$

where $\dim(\mathbf{s}(\mathbf{v}_i))$ is the dimension (i.e. the number of bins) of $\mathbf{s}(\mathbf{v}_i)$, and $\mathbf{s}(\mathbf{v}_i)[b]$ is the value of $\mathbf{s}(\mathbf{v}_i)$ at the b-th bin.

Given the deformation cost of every pair of vertices on two shapes $\mathcal{V}$ and $\mathcal{Y}$, shape matching can be solved using the shortest augmenting path algorithm [50]. To make the matching algorithm adaptive to shapes with different number of vertices, "dummy" vertices are added. This enables the matching method to be robust to noisy data and partial occlusions. Formally, the deformation cost $C(\mathcal{V}, \mathcal{Y})$ between two shapes $\mathcal{V}$ and $\mathcal{Y}$ is computed as,

$$C(\mathcal{V}, \mathcal{Y}) = \sum_{\mathbf{v}_i \in \hat{\mathcal{V}}} \chi^2(\mathbf{s}(\mathbf{v}_i), \mathbf{s}(\pi(\mathbf{v}_i))) \quad (6)$$

where $\hat{\mathcal{V}}$ is identical to $\mathcal{V}$ or augmented from $\mathcal{V}$ by adding dummy vertices and $\pi(\mathbf{v}_i) \in \hat{\mathcal{Y}}$ is the matching vertex of $\mathbf{v}_i$ determined by using [50].

To further improve the matching, we also consider how well the two shapes correspond. In particular, we first align $\mathcal{V}$ to $\mathcal{Y}$ using a rigid transformation. This rigid transformation is represented by a 4×4 matrix and estimated using the RANSAC algorithm that randomly picks three pairs of correspondences and determine the rotation and translation [51]. We then compute an alignment error,

$$E(\mathcal{V}, \mathcal{Y}) = \min\left\{\sqrt{\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\epsilon_i^{(\mathcal{V})}}, \sqrt{\frac{1}{|\mathcal{Y}|}\sum_{i=1}^{|\mathcal{Y}|}\epsilon_i^{(\mathcal{Y})}}\right\} \quad (7)$$

where

$$\epsilon_i^{(\mathcal{V})} = \begin{cases} \|\pi(\mathbf{v}_i) - T * \mathbf{v}_i\|^2 & \text{if } \pi(\mathbf{v}_i) \text{ exists for } \mathbf{v}_i \in \mathcal{V} \\ \Delta^2 & \text{otherwise} \end{cases} \quad (8)$$

and, similarly for $\epsilon_i^{(\mathcal{Y})}$, where T is the rigid transformation matrix and Δ is a large value used to penalize misalignments.

A match is confirmed if: (i) $C(\mathcal{V}, \mathcal{Y}) < \tau_s$ and (ii) $E(\mathcal{V}, \mathcal{Y}) < \tau_a$ where τ_s and τ_a are thresholds. In our experiments, we set $\Delta = 2$ (meters), $\tau_s = 0.7$, $\tau_a = 0.4$. We found that the object search method was not too sensitive to parameter settings, and that these settings achieved the best performance.

7.3 Searching

Object search can be performed based on the sliding-window approach [48]. Specifically, we take the 3D bounding box of the template and use it as the window to scan a 3D scene. At each location in the scene, all regions that intersect the window are considered for their possibility to be part of a matching object. However, it would be intractable to consider every possible combination of all regions. To deal with this issue, we propose a greedy algorithm that operates iteratively by adding and removing regions.

The general idea is as follows. Let $\mathcal{R}$ be the set of regions that intersect the window $\mathcal{W}$, i.e. the 3D bounding box of the template. For a region $r \in \mathcal{W}\backslash\mathcal{R}$, we verify whether the object made by $\mathcal{R} \cup \{r\}$ could be more similar to the user-defined template $\mathcal{O}$ in comparison with $\mathcal{R}$. Similarly, for every region $r \in \mathcal{R}$ we also verify the object made by $\mathcal{R}\backslash\{r\}$. These adding and removing steps are performed interchangeably in a small number of iterations until the best matching result (i.e. a group of regions) is found. This procedure is called *grow-shrink* and described in Algorithm 1.

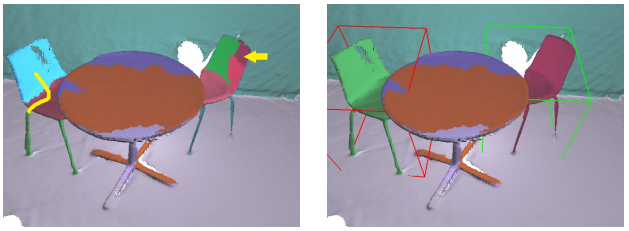
In our implementation, the spatial strides on the $x-$, $y-$, and $z-$ direction of the window $\mathcal{W}$ were set to the size of $\mathcal{W}$. The number of iterations in Algorithm 1 was set to 10, which resulted in satisfactory accuracy and efficiency (see Section 9.3).

Since a region may be contained in more than one window, it may be verified multiple times in multiple groups of regions. To avoid this, if an object candidate is found in a window, its regions will not be considered in any other objects and any other windows. Fig. 5 illustrates the robustness of the object search in localizing repetitive objects under severe conditions (e.g. objects with incomplete shape).

The search procedure may miss some objects. To handle such cases, we design an operation called *guided merge*. In particular, after defining the template, users simply select one of the regions of a target object that is missed by the object search. The grow-shrink procedure is then applied on the selected region to seek a better match with the template. Fig. 6 shows an example of the guided merge operation.

function GrowShrink $(\mathcal{R}, \mathcal{W}, \mathcal{O})$
Input : $\mathcal{R}$: set of regions to examine,
$\mathcal{W}$: window,
$\mathcal{O}$: user-defined template
Output: $\mathcal{A}$: best matching object
begin
  $\mathcal{A} \leftarrow \mathcal{R}$
  for $i \leftarrow 1$ **to** *iterations* **do**
    // grow
    $\mathcal{M} \leftarrow \mathcal{A}$
    for $r \in \mathcal{W} \setminus \mathcal{M}$ **do**
      if $C(\mathcal{M} \cup \{r\}, \mathcal{O}) < C(\mathcal{A}, \mathcal{O})$ **and** $E(\mathcal{M} \cup \{r\}, \mathcal{O}) < \tau_a$ **then**
        $\mathcal{A} \leftarrow \mathcal{M} \cup \{r\}$
      end
    end
    // shrink
    $\mathcal{M} \leftarrow \mathcal{A}$
    for $r \in \mathcal{M}$ **do**
      if $C(\mathcal{M} \setminus \{r\}, \mathcal{O}) < C(\mathcal{A}, \mathcal{O})$ **and** $E(\mathcal{M} \setminus \{r\}, \mathcal{O}) < \tau_a$ **then**
        $\mathcal{A} \leftarrow \mathcal{M} \setminus \{r\}$
      end
    end
  end
  return $\mathcal{A}$
end

Algorithm 1: Grow-shrink procedure. C and E are the matching cost and alignment error defined in (6) and (7).



(a) (b)

Fig. 6. Suppose that the right chair cannot be detected by the object search. (a) Input: a template specified by a stroke on its regions and an initial region (marked by the arrow) of the target object. (b) Output: labels of the target (the right chair) are merged automatically by applying the grow-shrink procedure.

8 Segmentation in 2D

Segmentation in 2D can be done by projecting regions in 3D space into 2D frames. However, the projected regions may not align well with the true objects in 2D frames (see Fig. 7). There are several reasons for this issue. For example, the depth and color images used to reconstruct a scene might not be exactly aligned at object boundaries, or the camera intrinsics might be from factory settings and not well calibrated. We note that manual calibration is not easy for novice users to perform. Moreover, auto calibration might not work reliably especially when scene features are lacking and camera registration during reconstruction exhibits drift.

To overcome this issue, we propose an alignment algorithm which aims to fit the boundaries of projected regions to true boundaries in 2D frames. The true boundaries in a 2D frame can be extracted using some edge detector (e.g. the Canny edge detector [52]). Let $\mathcal{E} = \{e_j\}$ denote the set of edge points on the edge map of a 2D frame. Let $U = \{u_i\}$ be the set of contour points of a projected object in that frame. U is then ordered using the Moore neighbor tracing algorithm [53]. The ordering step is used to express the contour alignment problem in a form to which dynamic programming can be applied for efficient implementation.

At each contour point u_i, we consider a 21×21-pixel window centered at u_i (in relative to a 640×480-pixel image). We then extract the histogram h_{u_i} of the orientations of vectors (u_i, u_k), $u_k \in U$ in the window. The orientations are uniformly quantized into 16 bins. We also perform this operation for edge points $e_j \in \mathcal{E}$. The dissimilarity between the two local shapes at a contour point u_i and edge point e_j is computed as $\chi^2(h_{u_i}, h_{e_j})$ (similarly to (5)).

We also consider the continuity and smoothness of contours. In particular, the continuity between two adjacent points u_i and u_{i-1} is defined as $\|u_i - u_{i-1}\|$. The smoothness of a fragment including three consecutive points u_i, u_{i-1}, u_{i-2} is computed as $\cos(u_i - u_{i-1}, u_{i-2} - u_{i-1})$ where $u_i - u_{i-1}$ and $u_{i-2} - u_{i-1}$ denote the vectors connecting u_{i-1} to u_i and connecting u_{i-1} to u_{i-2} respectively, and $\cos(\cdot, \cdot)$ is the cosine of the angle formed by these two vectors.

Alignment of U to $\mathcal{E}$ is achieved by identifying a mapping function $f : U \rightarrow \mathcal{E}$ that maps a contour point $u_i \in U$ to an edge point $e_j \in \mathcal{E}$ so as to,

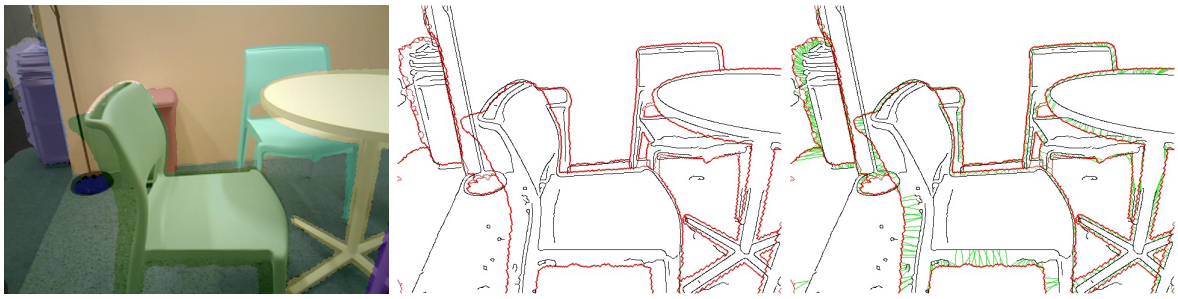
$$\begin{aligned} \text{minimize} \Bigg[& \sum_{i=1}^{|U|} \chi^2(h_{u_i}, h_{f(u_i)}) \\ & + \kappa_1 \sum_{i=2}^{|U|} \|f(u_i) - f(u_{i-1})\| \\ & + \kappa_2 \sum_{i=3}^{|U|} \cos(f(u_i) - f(u_{i-1}), f(u_{i-2}) - f(u_{i-1})) \Bigg] \end{aligned} \quad (9)$$

The optimization problem in (9) can be considered as the bipartite graph matching problem [50]. However, since U is ordered, this optimization can be solved efficiently using dynamic programming [54]. In particular, denoting $m_{i,j} = \chi^2(h_{u_i}, h_{e_j})$, $f_i = f(u_i)$, $f_{i,j} = f(u_i) - f(u_j)$, the objective function in (9) can be rewritten as,

$$\mathcal{F}_i = \begin{cases} \min_{j \in \mathcal{E}} \{\mathcal{F}_{i-1} + m_{i,j} + \kappa_1 \|f_{i,i-1}\| & \\ \quad + \kappa_2 \cos(f_{i,i-1}, f_{i-2,i-1})\}, & \text{if } i > 2 \\ \min_{j \in \mathcal{E}} \{\mathcal{F}_{i-1} + m_{i,j} + \kappa_1 \|f_{i,i-1}\|\}, & \text{if } i = 2 \\ \min_{j \in \mathcal{E}} \{m_{i,j}\}, & \text{if } i = 1 \end{cases} \quad (10)$$

where κ_1 and κ_2 are user parameters. Empirically, we set $\kappa_1 = 0.1$ and $\kappa_2 = 3.0$ in all of our experiments.

To save the computational cost of (10), for each contour point u_i, we consider only its k nearest edge points whose distance to u_i is less than a distance δ set to 10% of the maximum of the image dimension, e.g., $\delta = 48$ for a 640×480-pixel



(a) (b) (c)

Fig. 7. Segmentation of a 2D frame. (a) Segmentation by projecting objects from 3D into 2D. Projected regions are overlaid on the RGB frame. (b) Contours of projected regions (in red) and Canny's edges (in black). (c) The correspondences between contour points and edge points (green lines) obtained from the alignment algorithm. Note that only a few sampled points on the contours are selected for illustration.

image. The number of nearest edge points (i.e. k) was set to 30. Fig. 7(c) shows an example of contour alignment by optimizing (10) using dynamic programming.

We have verified the contribution of the continuity and smoothness in Fig. 8. The results show that, when all the cues are taken into account, the contours are mostly well aligned with the true object boundaries. It is noticeable that the seat of the green chair is not correctly recovered. We have found that this is because the Canny detector missed important edges on the boundaries of the chair. Recently, new edge detectors (e.g. [55]) have been shown to work robustly under severe illumination conditions and in the face of complex texture changes. These edge detectors could be used to enhance the alignment algorithm.

9 Experiments

9.1 Dataset

We created a dataset consisting of over 100 scenes. The dataset includes six scenes from publicly available datasets: the *copyroom* and *lounge* from the Stanford dataset [38], the *hotel* and *dorm* from the SUN3D [3], and the *kitchen* and *office* sequences from the Microsoft dataset [56]. The Stanford and SUN3D dataset also provide registered RGB and depth image pairs. These datasets also include the camera pose data.

In addition to existing scenes, we collected 100 scenes using Asus Xtion and Microsoft Kinect v2. Our scenes were captured from the campus of the University of Massachusetts Boston and the Singapore University of Technology and Design. These scenes were captured from various locations such as lecturer rooms, theaters, university hall, library, computer labs, dormitory, etc. There were about 1,600 object instances of 20 object categories. The objects vary in shape and color. Their sizes also vary significantly; some objects are quite small, e.g. glasses ($<$ 10cm), while others are very large, e.g. beds (about 2m), walls (about 5m). All the scenes and objects were fully segmented and annotated using our tool. The camera pose information is also included. The dataset is available at http://scenenn.net. Fig. 9 shows several of our collected scenes.

9.2 Evaluation of 3D Segmentation

We evaluated the impact of the graph-based and MRF-based segmentation on our dataset. We considered the annotated results obtained using our tool as the ground-truth. To measure the segmentation performance, we extended the object-level consistency error (OCE), the image segmentation evaluation metric proposed in [57] to 3D vertices. Essentially, the OCE reflects the coincidence of pixels/vertices of segmented regions and ground-truth regions. As indicated in [57], compared with other segmentation evaluation metrics (e.g. the global and local consistency error in [58]), the OCE considers both over- and under-segmentation errors in a single measure. In addition, OCE can quantify the accuracy of multi-object segmentation and thus fits our evaluation purpose well.

Table 1 summarizes the OCE of the graph-based and MRF-based segmentation. As shown in the table, compared with the graph-based segmentation, the segmentation accuracy is significantly improved by the MRF-based segmentation. It is also noticeable that the number of supervertices and regions yielded by the segmentation process is much smaller than the number of 3D vertices that were passed in, making them significantly more convenient for the users to work with. The OCE values presented in Table 1 show that automatic segmentation is still not approaching the quality achieved by human beings. Thus, user interactions are necessary. This is for two reasons: first, both the graph-based and MRF-based segmentation aim to segment a 3D scene into homogenous regions rather than semantically-meaningful objects; second, semantic segmentation is user-dependent and subjective [58].

After user interaction, the number of final labels is typically less than a hundred. The number of objects is around 10 to 20 in most of the cases. Note that the numbers of final labels and annotated objects are not identical. This is because there could have been labels whose semantics is not well defined, e.g. miscellaneous items on a table or some small labels appearing as noise in the 3D reconstruction.

We also report the time required for the segmentation and user interactions with the tool in Table 1. As shown in the table, with the assistance of the tool, a complex 3D scene (with millions of vertices) could be completely segmented and annotated in less than 30 minutes, as opposed to approximately

Segmentation by projecting 3D objects into a 2D frame

Using the local shape only (i.e. $\kappa_1 = \kappa_2 = 0$)

Using the local shape and continuity

Using the local shape and smoothness

Using the local shape, continuity, and smoothness

Fig. 8. Illustration of 2D segmentation. First column: segmentation result obtained by projection of 3D regions and overlaid on RGB images. Remaining columns: close-ups of four regions marked in the segmentation result in the first column.

Fig. 9. Results of our captured scenes. From left to right: the result of graph-based segmentation, MRF-based segmentation, and final segmentation and annotation made by user interaction. Please refer to our supplemental document for the results of other scenes.

TABLE 1
Comparison of the graph-based segmentation and MRF-based segmentation. For our captured scenes, the statistical data is the average numbers calculated over all the scenes. Note that for user refined results, the numbers of objects annotated are fewer than the numbers of labels (i.e. segments). This is because the annotation was done only for objects that are common in practice.

Scene	#Vertices	Graph-based			MRF-based			User refined		Interactive time
		#Supervertices	OCE	Time (seconds)	#Regions	OCE	Time (seconds)	#Labels	#Objects	(minutes)
copyroom	1,309,421	1,996	0.92	1.0	347	0.73	10.9	157	15	19
lounge	1,597,553	2,554	0.97	1.1	506	0.93	7.3	53	12	16
hotel	3,572,776	13,839	0.98	2.7	1433	0.88	17.8	96	21	27
dorm	1,823,483	3,276	0.97	1.2	363	0.78	7.8	75	10	15
kitchen	2,557,593	4,640	0.97	1.8	470	0.85	12.2	75	24	23
office	2,349,679	4,026	0.97	1.7	422	0.84	10.9	69	19	24
Our scenes	1,450,748	2,498	0.93	1.4	481	0.77	12.1	179	19	30

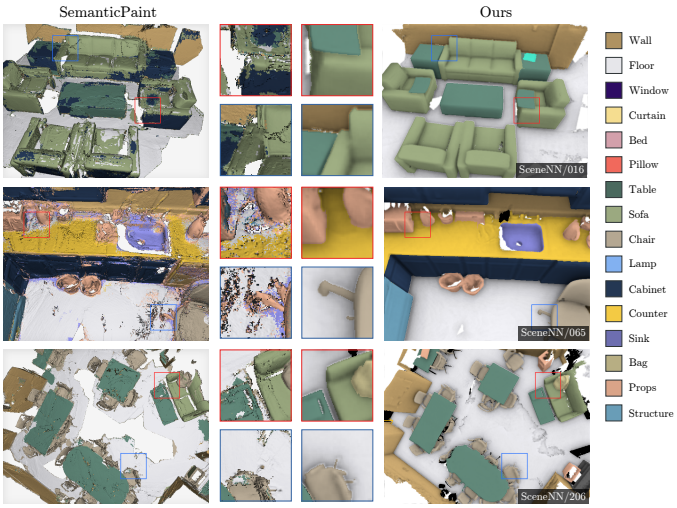


Fig. 10. Comparison of SemanticPaint [21] and our system.

a few hours to be done manually. Note that the interactive time is subject to a user's experience. Several results of the tool are shown in Fig. 9. We also compare our overall system with the public version of SemanticPaint [21] in Fig. 10. Note that this is only a qualitative comparison since different reconstruction methods were used in two systems. Although SemanticPaint offers real-time performance, it suffers from high frequency noise due to the segment propagation mechanism. By contrast, our system shows smoother segmentation results because it is based on supervertices.

9.3 Evaluation of Object Search

To evaluate the object search, we collected a set of 45 objects from our dataset. Those objects were selected so that they are semantically-meaningful and their shapes are discriminative. For example, drawers of cabinets were not selected since they were present in flat surfaces which could be easily found in many structures, e.g. walls, pictures, etc. For each scene and each object class (e.g. chair), each object instance in the class was used as the template while the object search was applied to find the remaining objects of the same class.

We used the intersection over union (IoU) metric [13] to determine true detections and false alarms. However, instead of computing the IoU on object bounding boxes as in [13], we computed the IoU at point-level. This is because our aim is not only to localize repetitive objects but also to segment them. In particular, an object $\mathcal{O}$ (a set of vertices) formed by the object search procedure is considered as true detection if there exists an annotated object $\mathcal{R}$ in the ground-truth such that $\frac{|\mathcal{O}\cap\mathcal{R}|}{|\mathcal{O}\cup\mathcal{R}|} > 0.5$ where $|\cdot|$ denotes the area.

The evaluation was performed on every template. The precision, recall, and F-measure ($= 2\times\frac{Precision\times Recall}{Precision+Recall}$) were then averaged over all evaluations. Table 2 shows the averaged precision, recall, and F-measure of the object search. As shown, the object search can localize 70% of repetitive objects with 69% precision and 65% F-measure. We have also tested the object search without considering the alignment error, i.e. E in (7). We have found that, compared with the solely use of shape context dissimilarity score (i.e. C in (6)), while the augmentation of alignment error could slightly incur a loss of the detection rate (about 2%), it largely improved the precision (from 22% to 69%). This led to a significant increase of the F-measure (from 30% to 65%).

Experimental results show that, the object search worked efficiently with templates represented by about 200 points. For example, the scene presented in Fig. 5 was completed within 15 seconds with a 150-point template and on a machine equipped by an Intel(R) Core(TM) i7 2.10 GHz CPU and 32 GB of memory. Note that threads can be used to run the object search in the background while users are performing interactions.

TABLE 2
Performance of the proposed object search.

	Precision	Recall	F-measure
Without alignment error	0.22	**0.72**	0.30
With alignment error	**0.69**	0.70	**0.65**

9.4 Evaluation of 2D Segmentation

We also evaluated the performance of the 2D segmentation using the OCE metric. This experiment was conducted on

the *dorm* sequence from the SUN3D dataset [3]. The *dorm* sequence contained 58 images in which the ground-truth labels were manually crafted and publicly available.

We report the segmentation performance obtained by projecting 3D regions into 2D images and by applying our alignment algorithm in Table 3. The impact of the local shape, continuity, and smoothness is also quantified. As shown in Table 3, the combination of the local shape, continuity, and smoothness achieves the best performance. We have visually found the alignment algorithm could make the projected contours smoother and closer to the true edges.

Experimental results show our alignment algorithm worked efficiently. On average, the alignment could be done in about 1 second for a 640×480-pixel frame.

TABLE 3
Comparison of different segmentation methods.

Segmentation method	OCE
Projection	0.57
Local shape	0.60
Local shape + Continuity	0.55
Local shape + Smoothness	0.55
Local shape + Continuity + Smoothness	**0.54**

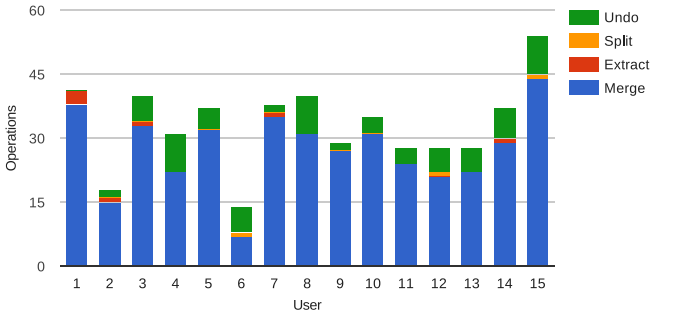
10 Conclusion

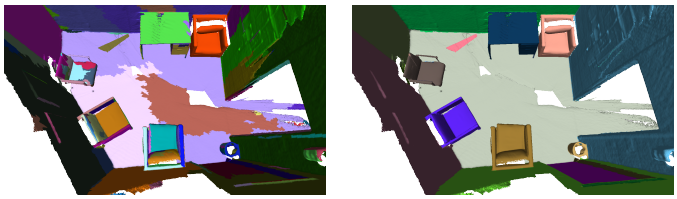
This paper proposed a robust tool for segmentation and annotation of 3D scenes. The tool couples 3D geometric information and multi-view 2D RGB information in an interactive framework. To enhance the usability of the tool, we developed assistive, interactive operations that allow users to flexibly manipulate scenes and objects in both 3D and 2D space. The tool is also equipped with automated functionalities such as scene and image segmentation, and object search.

Along with the tool, we created a dataset of more than 100 scenes. All the scenes were annotated using our tool. The overall performance of the tool depends on the quality of 3D reconstruction. We are currently improving the quality of 3D meshes by recovering broken surfaces and missing object parts. User interactions could provide valuable information for many tasks, e.g. segmentation, annotation, and object search. Enhancing the performance of these tasks based on learning user interaction patterns (e.g. automatically learning templates and parameters) will also be our future work.

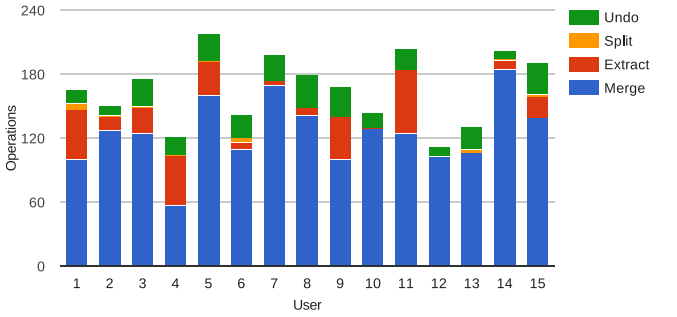
Appendix

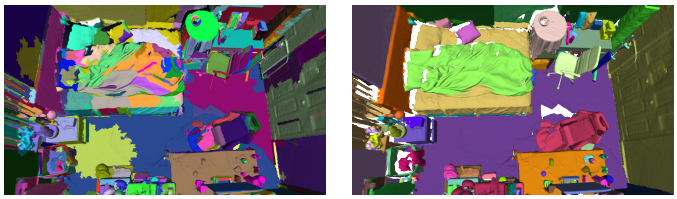
We have conducted a user study on the effectiveness of our interactive operations (merge, split, extract, and undo). There were 15 users involved in this study. The user study included two tasks (A and B) designed to handle simple and complex scenes. In task A, users were asked to segment a scene with only a few chairs and a table in two minutes. In task B, users were required to segment a complex bedroom scene containing a significant amount of furniture and many small objects in ten minutes. All operations users performed were logged. The statistics are shown in Fig. 11.





(a) Task A (two minutes)





(b) Task B (ten minutes)

Fig. 11. User study statistics. In each task, users were given an initial segmentation (left), which was then refined using interactive operations (right).

As can be seen, in both segmentation tasks, merge operation dominates, followed by extract and split operation. While a few users prefer split over extract (e.g. user 4, 6, 12, and 15 in task A; user 1 and 6 in task B), in such cases the number of split operation is always very few compared to merge. Note that our tool offers both flexibility in segmenting objects with various sizes as well as efficiency in computation since small segments for extract can be pre-computed.

In addition, task A also shows that merge is the most straightforward operation for new users. Extract and split are more useful when a scene gets more complex, with some coarse segments spread over small objects.

ACKNOWLEDGMENT

Binh-Son Hua is supported by the SUTD Digital Manufacturing and Design (DManD) Centre which is supported by the National Research Foundation (NRF) of Singapore.

Lap-Fai Yu is supported by the Joseph P. Healey Research Grant Program provided by the Office of the Vice Provost for Research and Strategic Initiatives & Dean of Graduate Studies of UMass Boston. This research is also supported by the National Science Foundation under award number 1565978. We acknowledge NVIDIA Corporation for graphics card donation.

Sai-Kit Yeung is supported by Singapore MOE Academic Research Fund MOE2016-T2-2-154 and Supported by Heritage Research Grant of the National Heritage Board, Singapore. We acknowledge the support of the SUTD Digital Manufacturing and Design (DManD) Centre which is supported by the National Research Foundation (NRF) of Singapore. This research is also supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and its Environmental & Water Technologies Strategic Research Programme and administered by the PUB, Singapores National Water Agency. This material is based on research/work supported in part by the National Research Foundation under Virtual Singapore Award No. NRF2015VSG-AA3DCM001-014.

We thank Fangyu Lin and Quang-Hieu Pham for assisting with data capture and Guoxuan Zhang for the early version of the tool. The Lounge scene data in Figure 2 is courtesy of Zhou and Koltun [38].

REFERENCES

[1] H. Roth and M. Vona, "Moving volume kinectfusion," in *British Machine Vision Conference*, 2012.

[2] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, 2013.

[3] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in *IEEE International Conference on Computer Vision*, 2013.

[4] Q. Y. Zhou and V. Koltun, "Simultaneous localization and calibration: Self-calibration of consumer depth cameras," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2014.

[5] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. Mcdonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[6] O. Kähler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 11, pp. 1241–1250, 2015.

[7] J. P. C. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. S. Torr, "Mesh based semantic modelling for indoor and outdoor scenes," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[8] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys, "Joint 3D scene reconstruction and class segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[9] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shape modeling," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[10] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert, "From 3D scene geometry to human workspace," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[11] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[12] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, "Imagenet: A large-scale hierarchical image database," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[13] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[14] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[15] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[16] J. Deng, A. C. Berg, , K. Li, and F. F. Li, "What does classifying more than 10, 000 image categories tell us?" in *European Conference on Computer Vision*, 2010.

[17] X. Ren, L. Bo, and D. Fox, "RGB-(D) scene labeling: features and algorithms," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2012.

[18] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[19] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[20] M. Jancosek and T. Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[21] J. Valentin, V. Vineet, M. M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Niessner, A. Criminisi, S. Izadi, and P. Torr, "Semanticpaint: Interative 3D labeling and learning at your fingertips," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 5, 2015.

[22] O. Miksik, V. Vineet, M. Lidegaard, R. Prasaath, M. Nießner, S. Golodetz, S. L. Hicks, P. Pérez, S. Izadi, and P. H. S. Torr, "The semantic paintbrush: Interactive 3D mapping and recognition in large outdoor spaces," in *ACM Conference on Human Factors in Computing Systems*, 2015.

[23] R. Guo and D. Hoiem, "Support surface prediction in indoor scenes," in *IEEE International Conference on Computer Vision*, 2013.

[24] Z. Jia, A. C. Gallagher, A. Saxena, and T. Chen, "3D-based reasoning with blocks, support, and stability," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[25] D. Lin, S. Fidler, and R. Urtasun, "Holistic scene understanding for 3D object detection with RGBD cameras," in *IEEE International Conference on Computer Vision*, 2013.

[26] B. Kim, P. Kohli, and S. Savarese, "3D scene understanding by voxel-CRF," in *IEEE International Conference on Computer Vision*, 2013.

[27] Y. S. Wong, H. K. Chu, and N. J. Mitra, "Smartannotator: An interactive tool for annotating RGBD indoor images," *Computer Graphics Forum*, vol. 34, no. 2, 2015.

[28] Y. Wang, R. Ji, and S. F. Chang, "Label propagation from imagenet to 3D point clouds," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[29] Y. M. Kim, N. J. Mitra, D. M. Yan, and L. Guibas, "Acquiring 3D indoor environments with variability and repetition," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, 2012.

[30] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[31] L. Nan, K. Xie, and A. Sharf, "A search-classify approach for cluttered indoor scene understanding," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, 2012.

[32] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an RGBD camera," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, 2012.

[33] K. Chen, Y. K. Lai, Y. X. Wu, R. Martin, and S. M. Hu, "Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, 2014.

[34] Y. Zhang, W. Xu, Y. Tong, and K. Zhou, "Online structure analysis for real-time indoor scene reconstruction," *ACM Transactions on Graphics*, 2015.

[35] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[36] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[37] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[38] Q. Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, 2013.

[39] N. Fioraio, J. Taylor, A. W. Fitzgibbon, L. D. Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[40] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[41] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3D reconstruction with loop closure," in *European Conference on Computer Vision*, 2016.

[42] W. E. Lorensen and H. E. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[43] P. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[44] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[45] A. Karpathy, S. D. Miller, and F. F. Li, "Object discovery in 3D scenes via shape analysis," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2088–2095.

[46] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[47] S. A. Barker and P. J. W. Rayner, "Unsupervised image segmentation using markov random field models," *Pattern Recognition*, vol. 33, no. 4, pp. 587–602, 2000.

[48] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[49] M. Körtgen, G. J. Park, M. Novotni, and R. Klein, "3D shape matching with 3D shape contexts," in *Central European Seminar on Computer Graphics*, 2003.

[50] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, pp. 325–340, 1987.

[51] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[52] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[53] N. Narappanawar, B. M. Rao, and M. Joshi, "Graph theory based segmentation of traced boundary into open and close sub-sections," *Computer Vision and Image Understanding*, vol. 115, no. 11, pp. 1552–1558, 2011.

[54] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2003, pp. 127–133.

[55] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[56] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. W. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

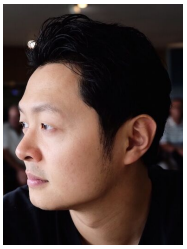
[57] M. Polak, H. Zhang, and M. Pi, "An evaluation metric for image segmentation of multiple objects," *Image and Vision Computing*, vol. 27, pp. 1123–1127, 2009.

[58] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating algorithms and measuring ecological statistics," in *IEEE International Conference on Computer Vision*, 2001, pp. 416–423.

Duc Thanh Nguyen received his Ph.D. degree in Computer Science from the University of Wollongong, Australia, in 2012. Currently, he is a lecturer at the School of Information Technology, Deakin University, Australia. His research interests include Computer Vision and Pattern Recognition. Dr. Nguyen has published his work in highly ranked publication venues in Computer Vision and Pattern Recognition such as the Journal of Pattern Recognition, CVPR, ICCV, ECCV. He also has served a technical program committee member of the IEEE Int. Conf. Image Process. (from 2012) and reviewer of the IEEE Trans. Intell. Transp. Syst., IEEE Trans. Image Process., IEEE Signal Processing Letters, Image and Vision Computing.

Binh-Son Hua is currently a postdoctoral researcher in Singapore University of Technology and Design. He received his PhD degree in Computer Science from National University of Singapore in 2015. He also received his B.E. (Hons) from Ho Chi Minh City University of Technology, Vietnam, in 2008. His research interests are 3D reconstruction, 3D scene understanding, and physically based rendering.

Lap-Fai (Craig) Yu is an assistant professor at the University of Massachusetts at Boston. He obtained his PhD degree in computer science from UCLA in 2013. His research interests are in computer graphics and vision, especially in the topics of synthesizing and analysing 3D models from the perspectives of functionality, physics, intentionality and causality. He is the recipient of the Cisco Outstanding Graduate Research Award, the UCLA Dissertation Year Fellowship, the Sir Edward Youde Memorial Fellowship and the Award of Excellence from Microsoft Research. His research has been featured in New Scientist, the UCLA Headlines and newspapers internationally. He regularly serves on the program committee of EUROGRAPHICS, Pacific Graphics and IEEE Virtual Reality.



Sai-Kit Yeung is currently an Assistant Professor at the Singapore University of Technology and Design (SUTD), where he leads the Vision, Graphics and Computational Design (VGD) Group. He was also a Visiting Assistant Professor at Stanford University and MIT. Before joining SUTD, he had been a Postdoctoral Scholar in the Department of Mathematics, University of California, Los Angeles (UCLA). He was also a visiting student at the Image Processing Research Group at UCLA in 2008 and at the Image Sciences Institute, University Medical Center Utrecht, the Netherlands in 2007. He received his PhD in Electronic and Computer Engineering from the Hong Kong University of Science and Technology (HKUST) in 2009. He also received a BEng degree (First Class Honors) in Computer Engineering in 2003 and a MPhil degree in Bioengineering in 2005 from HKUST. His research interests include computer vision, computer graphics and computational fabrication.