View suggestion for interactive segmentation of indoor scenes

Computational Visual Media
DOI 10.1007/s41095-017-0078-4 Vol. 3, No. 2, June 2017, 131–146

Research Article

View suggestion for interactive segmentation of indoor scenes

Sheng Yang[1] (✉), **Jie Xu**[2], **Kang Chen**[1], **and Hongbo Fu**[3]

Abstract Point cloud segmentation is a fundamental problem. Due to the complexity of real-world scenes and the limitations of 3D scanners, interactive segmentation is currently the only way to cope with all kinds of point clouds. However, interactively segmenting complex and large-scale scenes is very time-consuming. In this paper, we present a novel interactive system for segmenting point cloud scenes. Our system automatically suggests a series of camera views, in which users can conveniently specify segmentation guidance. In this way, users may focus on specifying segmentation hints instead of manually searching for desirable views of unsegmented objects, thus significantly reducing user effort. To achieve this, we introduce a novel view preference model, which is based on a set of dedicated view attributes, with weights learned from a user study. We also introduce support relations for both graph-cut-based segmentation and finding similar objects. Our experiments show that our segmentation technique helps users quickly segment various types of scenes, outperforming alternative methods.

Keywords point cloud segmentation; view suggestion; interactive segmentation

1 Introduction

With the prevalence of consumer-grade depth sensors (e.g., Microsoft Kinect), scanning our living environments is becoming easier. However, the resulting 3D point clouds are often noisy, incomplete, and distorted, posing various challenges to traditional point cloud processing algorithms. Thus, in recent years, growing attention has been paid to low-quality point cloud processing problems. Amongst them, semantic segmentation, which aims to provide a decomposition of a 3D point cloud into semantically meaningful objects, is one of the most fundamental problems, and is important for many subsequent tasks such as object detection [1], object recognition [2], scene understanding [3], etc.

Semantic segmentation of 3D point clouds has been extensively studied, resulting in various techniques, based for instance on region growing [4, 5], graph-cut [6–8], learning [9–11], etc. Most of those approaches attempted to achieve semantic segmentation with little or even no user intervention. However, due to the complexity of real-world scenes and the limitations of 3D scanners, manual intervention is often inevitable [12].

Previous interactive segmentation work (e.g., Refs. [13, 14]) typically focuses on improving segmentation results given the same amount of user input (e.g., provided by a commonly used stroke-based interface). We observed that when interactively segmenting scenes at a moderate or large scale, finding appropriate views to provide segmentation hints is very time-consuming. For example, for a scene with multiple rooms containing objects of various types, shapes, and sizes, objects can easily occlude each other, requiring careful selection of viewpoints for interactive segmentation. In addition, due to the discrete nature of point clouds, the distances between viewpoint and objects need to be carefully chosen to ensure the desired point density and that contextual information is in view.

Based on these observations, we present a new interactive system for segmenting cluttered point

1 Tsinghua University, Beijing, China. E-mail: S. Yang, shengyang93fs@gmail.com (✉); K. Chen, chenkangnobel@hotmail.com.
2 Massachusetts Institute of Technology, Cambridge, USA. E-mail: eternal_answer@126.com.
3 City University of Hong Kong, Hong Kong, China. E-mail: fuplus@gmail.com.
Manuscript received: 2016-12-02; accepted: 2017-01-12

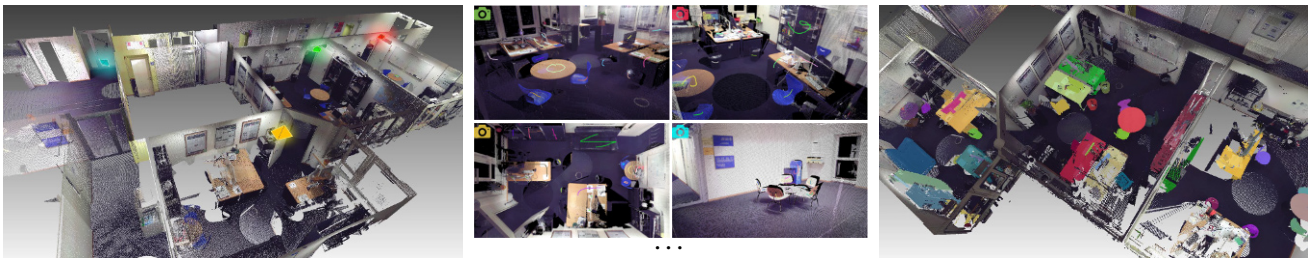


Fig. 1 Given an input scene represented as a 3D point cloud (left), our system automatically suggests a series of reasonable views (middle) for easily inputting segmentation hints for semantic segmentation of the entire scene (right).

clouds of large real-world scenes. Our system is able to automatically suggest a series of camera views, in which users can conveniently specify segmentation guidance, i.e., 2D strokes in our case. To reduce user effort, we aim to optimize the suggestions, i.e., to provide views that both contain plenty of undetermined objects and can clearly display them. To achieve this, we introduce a novel view preference model, which is based on a set of dedicated view attributes, whose weights are learned from a user study. Given a new scene, our system uses the learned view preference model to find the next best views one by one. In this way, users may focus on specifying segmentation hints, instead of manually searching for desired views for segmentation of unsegmented objects.

To further reduce user effort in interactive segmentation, we incorporate support relations in a graph-cut-based segmentation framework, to find similar objects for segmentation propagation. We have compared the performance of interactive point cloud segmentation with and without view suggestion, and interactive segmentation of RGB-D images. The experiments show that our segmentation technique with view suggestion helps users quickly segment various types of scenes, and outperforms alternative methods.

2 Related work

2.1 Point cloud segmentation

Semantic segmentation of 3D point clouds or RGB-D images has long been an active research topic in the communities of computer graphics, vision, and robotics (see an insightful survey by Nguyen and Le [12]). Below we discuss the most relevant works, and categorize them into supervised, unsupervised, and interactive techniques.

With the growing availability of free 3D datasets (e.g., the *NYU Depth Dataset* [11, 15], the *SUN3D* dataset [16], and ShapeNet [17]), supervised-learning-based segmentation algorithms typically exploit high-level semantic information from labeled datasets, and use the learned knowledge to help detect, recognize, and thus semantically separate object regions from backgrounds. Training data for supervised methods mainly come from two sources: CAD models and RGB-D images. High-quality CAD models are ideal training data as they provide full 3D a priori knowledge about geometric shapes [10, 18] and even contextual relationships [19], but the number and diversity of high-quality digital models are far from enough to cover everything; creating these models is expensive. Since RGB-D images are much easier to acquire, numerous methods (e.g., Refs. [1, 3, 20, 21]) learn various features from labeled RGB-D images. However, despite ease of acquisition, the labeling of RGB-D images is still labor intensive. Our system can be used as a convenient and robust segmentation tool to help produce high-quality training data.

Unsupervised semantic segmentation methods often rely on regular patterns (i.e., symmetry and repetition) observed from input data itself, and work extremely effectively on outdoor building facades [22, 23]. However, interior scenes present much more complex structures. To simplify the problem, previous works (e.g., Refs. [9, 24]) focus on large-scale working environments containing limited types of objects (e.g., office desk, office chair, monitor), each repeated many times. However, real-world scenes often contain unique objects. Thus, manual intervention is often still required to refine segmentation results produced by such automatic methods.

Since automatic segmentation methods are far from perfect, in practice interactive methods are more frequently used for segmenting both indoor

scenes [13, 25] and outdoor scenes [14]. The most commonly used interactive scheme is to let users specify representative foreground and background regions (typically via a stroke-based interface), which are then used to construct a probabilistic inference model (using, e.g., conditional random fields) and optimized using graph-cut [7, 8], or simply used as seeds for region growing [4, 5, 26]. However, such an interactive scheme is designed for segmenting an individual image frame, and thus requires carefully selected views of 3D scene contents for projection to screen space. While the view selection process is time-consuming, especially for large-scale and complex scenes, it has gained little attention. Thus, our work is largely complementary to existing interactive and automatic segmentation techniques.

2.2 Camera control

Finding feasible views to display 3D graphics on a 2D screen is a fundamental problem, which in computer graphics is generally referred to as the camera control problem [27, 28]. According to the camera mode and the scale of the 3D contents, existing solutions to this problem can be divided into two categories: *fly-around* and *walk-through*. We adapt these existing ideas to a new context for interactive point cloud segmentation, and tackle unique challenges such as the handling of cluttered point clouds, and dynamic selection of a series of good views for easy labeling of objects.

A *fly-around* camera allows complete contents rendered to the screen, and is often used when displaying single objects or small-scale scenes. The core problem is best view selection, which aims to automatically select the most representative view of a 3D model. Various low-level view attributes (e.g., projected area, viewpoint entropy, silhouette length, and depth distribution) have been proposed [29, 30] as a basis for solving this problem. The state of the art is probably the work by Secord et al. [29], which learns how to combine low-level view attributes based on an extensive user study of human preferences. Our work differs from the existing best view selection techniques in terms of both inputs (cluttered point clouds versus clean surface or volume models) and outputs (a series of viewpoints looking at different parts of a scene versus a set of independent good views looking at the same target).

Walk-through camera mode is thus more relevant to our problem; it is often used to navigate within large-scale virtual scenes [31, 32]. This problem essentially comprises two sub-problems: viewpoint selection [30] and path planning [33, 34]. Unlike the criteria used for fly-around cameras, measuring the quality of viewpoints within a scene is mainly based on *viewpoint entropy* [30]. Path planning is needed to avoid penetrating objects or walls and to present smooth scene roaming. In contrast, our problem demands essentially discrete views for labeling, although smooth transition is weakly considered. Again, existing walk-through methods often take clean scenes as input. Additionally, their extracted walk-through paths can be pre-generated, while in our case the selection of the next best view depends on the segmentation progress and needs to be determined on the fly. We have also found that top views are very useful for our application, but they are seldom used for the walk-through applications.

3 System overview

The main contribution of our work is automatic suggestion of good views for easy labeling of objects in indoor scenes represented as 3D point clouds. We implement view suggestion in an interactive segmentation system, which takes an unsegmented scan of an indoor scene as input. However, our interactive system can be helpful for interactively refining automatically generated segmentation results.

As shown in Fig. 2, our system contains three components: preprocessing, view suggestion, and interactive segmentation. In the preprocessing step (Section 4) the system automatically aligns an input scene with Manhattan directions, extracts storeys (levels), and clusters the points into patches which should be treated as indivisible units in segmentation.

Afterwards our system automatically finds candidate viewpoints and sample views for presenting as suggestions (Section 5.1). To evaluate the quality of a view, we introduce a view preference model (Section 5.2), which involves several attributes such as point density, projected area, and viewpoint entropy. The weights of these attributes are learned by conducting a user study of human preferences for views of scenes. At runtime, each view is evaluated using the learned view preference model, with

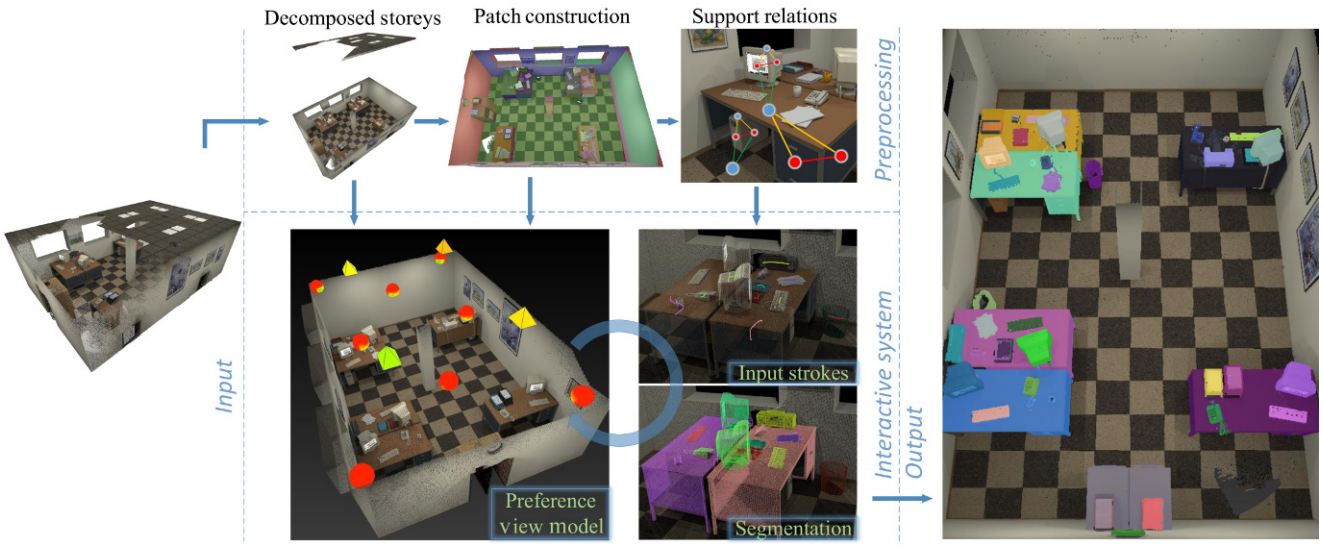


Fig. 2 System overview.

segmentation status and smooth transitions taken into account. The best view is then suggested to the user for interactive segmentation. The user may reject the current suggestion, and our system then updates the strategy for suggesting another view.

Given a view, the user provides segmentation hints on unsegmented, over-segmented, or under-segmented objects that are in view, using a classic stroke-based interface. The patches corresponding to these strokes are used as seeds to trigger a graph-cut-based segmentation optimization. We extract support relations between patches in the preprocessing step, and incorporate them into the segmentation optimization (Section 6.1). To further reduce the amount of user intervention, we find similar objects to already-segmented objects for segmentation propagation (Section 6.2). The above steps of automatic view suggestion and interactive segmentation of scenes in the selected suggested view are repeated until all objects have been labeled. For multi-storey cases, our system provides view suggestions storey by storey.

4 Preprocessing

In this section we briefly introduce our preprocessing step, which generates essential information for later use. Our system takes as input a 3D point cloud of an indoor scene, which can be acquired using different types of scanning devices such as LiDAR or Microsoft Kinect. For example, for RGB-D streams, they can be registered to form a point cloud using KinectFusion or its variants [35, 36]. Like many other point cloud processing pipelines [37], we downsample the data, estimate point normals, and transform the input scene into Manhattan coordinates [38] with the z-axis pointing upwards. Then we extract horizontal planes (by RANSAC) as floors and an optional ceiling. As illustrated in Fig. 3, we decompose a multiple-storey building into individual storeys, with the ceiling of each storey removed to achieve top views of rooms.

4.1 Constructing representative patches

Over-segmentation has often been used in existing works for preprocessing point clouds [7, 14, 24]. We are interested in over-segmenting an input scene into semantic patches, instead of patches with similar sizes but with no semantics [7]. Such patches not only help reduce computational costs but also serve as integral semantic units to analyze the importance of views and support relations.

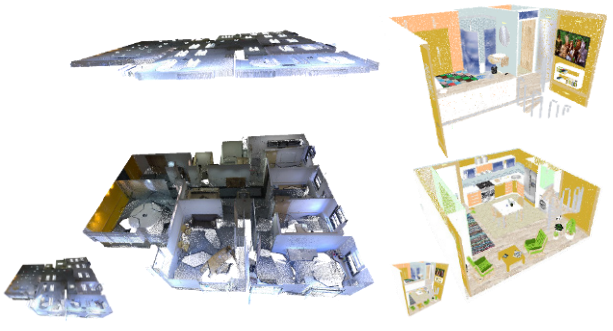


Fig. 3 Decomposition of one-storey (left) and two-storey (right) buildings into individual storeys, with the removal of ceilings (if any).

Our solution is an extension of the region growing approach by Ref. [24]. Since the original approach does not consider any color information, it might not split objects with similar shape but different colors (e.g., causing objects on the table in Fig. 4(c) to disappear). In addition, their approach has been shown to be effective only on good-quality scenes acquired by LiDAR devices, and can lead to numerous tiny patches for point clouds of low quality (e.g., those acquired by Kinect). To address these problems, we improve their approach by appropriately adding a new color condition and performing two rounds of growing, the first on the input point cloud and the second on the patches resulting from the first round.

The first round of region growing is applied to the input points. Specifically, let $G_0 = \langle V_0, E_0 \rangle$ denote a graph, with V_0 representing the input points and E_0 the edges generated by k-nearest neighborhoods ($k = 15$ in our implementation). As points in different colors more likely belong to different objects or parts, as well as the basic normal and position conditions (with the default parameters t_0 and t_1) for patch growing in Ref. [24], we add a color condition as follows:

$$\max\left(||R_i^{\mathrm{HS}} - R_j^{\mathrm{HS}}||, |C_i^{\mathrm{V}} - C_j^{\mathrm{V}}|\right) < t_2 \qquad (1)$$

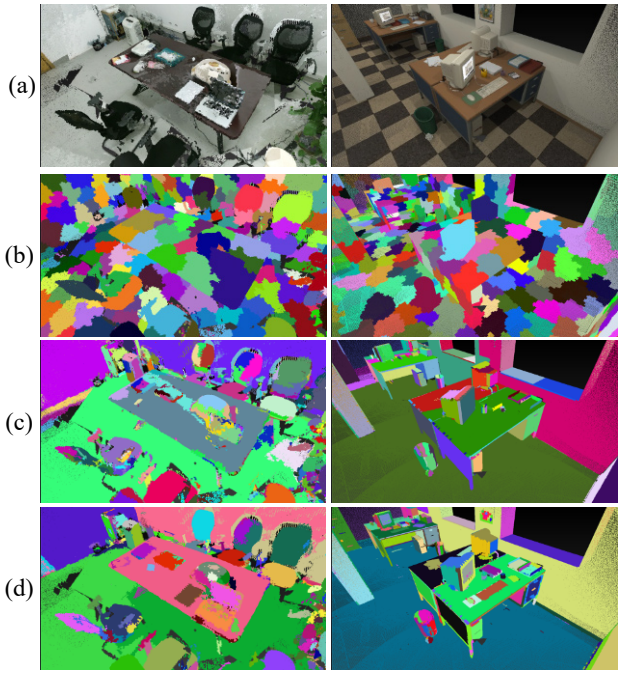


Fig. 4 Patch construction results. (a) Downsampled point clouds from different datasets. (b) By Ref. [7]. (c) By Ref. [24]. (d) Our result.

where i and j are two colors. $C_i^{\mathrm{H}} \in [0, 2\pi)$ and $C_i^{\mathrm{S}}, C_i^{\mathrm{V}} \in [0, 1]$ denote normalized HSV color values for i. $R_i^{\mathrm{HS}} = (C_i^{\mathrm{S}} \cos C_i^{\mathrm{H}}, C_i^{\mathrm{S}} \sin C_i^{\mathrm{H}})$ denotes the 2D coordinate values of i in the Hue–Saturation color disk. Intuitively, we allow two colors to be merged when their Euclidean distance in the Hue–Saturation color disk and intensity value difference are both within a small threshold t_2 (= 0.05 in our implementation).

This region growing process results in a set of merged patches. However, in our experiments we found that by adding the color condition, individual points might be isolated as individual patches, especially when the quality of the input point cloud is poor. To address this problem, we perform a second round of region growing on the merged patches from the first pass, but with relatively looser rules. Like G_0 is constructed over points, $G_1 = \langle V_1, E_1 \rangle$ is constructed over patches. For merging, we sort the patches in descending order of the number of contained points. For each patch, we find the normal of the best-fit plane and its centroid to check the respective normal and position conditions (the corresponding parameters are denoted t_3, t_4). The color condition is based on the color distribution and is more specifically defined using the χ-squared distance [39] between the HSV color histograms of two patches P_i and P_j in V_1:

$$\chi^2(I_i, I_j) < t_5 \qquad (2)$$

where I_i denotes the normalized HSV color histogram of points in P_i. Specifically, H, S, and V channels are discretized into 16, 16, and 8 bins respectively. We discretize V values into fewer bins to reduce the influence of different lighting conditions. The threshold t_5 is set to 1.6 in our system. The above patch-based region growing procedure is repeated multiple times (5 times in our implementation). After each iteration, the thresholds are relaxed by 20% to merge small isolated patches into larger patches. In our system, we set $t_0 = 0.8$, $t_1 = 0.05$, $t_3 = 0.75$, and $t_4 = 0.2$ for all input scenes.

5 View suggestion

In this section we first discuss view sampling, then perceptual assessment of views, and finally how to automatically suggest views for interactive segmentation.

5.1 View sampling

View sampling simplifies our problem, since it results in a discrete set of views for assessment. A view is basically determined by three vectors: the camera position, view direction, and up direction.

Inspired by previous works on urban city reconstruction using both airborne and street-borne data [40], we consider two types of views: *interior perspectives* and *top views*. Top views provide a good summary of a scene or its parts, since objects often lie on flat surfaces (e.g., tables, floors) and are thus easily separable from a top view. In contrast, interior perspectives enable a more detailed examination of a scene from a closer distance, and are useful for dealing with objects of small size or objects that are blocked in top views (e.g., objects in bookshelves). Thus these two types of views are largely complementary to each other. Next we explain how we sample these two types of views.

For interior perspectives, we fix the orientation of a view so that its up vector in image space is aligned with the upright orientation of a scene [29]. An interior perspective can then be characterized by (x, y, z, θ, ϕ), where x, y, and z define the camera position, and θ and ϕ control the view direction. A possible viewpoint should meet the following requirements: it must stay inside a room (above the floor, away from the walls, below the ceiling if any) and avoid hitting objects. Objects are approximately detected as columns using the column representation proposed by Fisher et al. [41]. We then perform uniform sampling in the space of a storey excluding the space occupied by the columns to get possible camera positions (x, y, z), each of which is associated with a view sphere parameterized by (θ, ϕ) (see Fig. 2). Both θ and ϕ are sampled every 15°, resulting in 266 views at each viewpoint.

For every top view, its view direction is fixed and always points downward along the negative z-axis. Lean view directions are not used since objects in such resulting views would often be severely blocked by vertical walls. Hence, θ is fixed at π for all top views in our implementation. We thus use (x, y, z, ϕ) for mapping the camera position of a view and its up direction. We uncover the ceiling of a room or storey and uniformly sample the viewpoints and poses inside its bounding box. It can be easily seen that higher viewpoints produce wider-range views but fewer details. In the case of a multiple-storey building, we decompose it into multiple storeys (Section 4), and sample interior perspectives and top views for individual storeys.

5.2 View preference model

For simplicity here we assume no object has been segmented in a view; we discuss how to incorporate the segmentation status in Section 5.3. The simplified problem is similar to the problem of best view selection for a single object; the latter is often represented as a polygonal mesh. In contrast we need to deal with scenes of objects, represented as cluttered point clouds. We observe that good views have at least the following properties. First, objects in such views should be easy to recognize. Second, such views should contain as many *unsegmented* objects as possible so that only a small set of views are needed to cover every object in a scene. These two properties are somewhat conflicting. For example, a bird's eye view of a scene might include many objects in the view, but individual objects might not be easily recognizable. Thus there is no single definition to describe whether a given view is good or not.

We explored various attributes to describe the quality of a view from different perspectives; some are based on previous work while others were designed by us. A key challenge is to evaluate the impact of each attribute when combining them together into a unified view preference model. This is achieved by learning from view preferences of human viewers in a user study. Below we first describe the important attributes indicated by the user study and then give the details of the study itself.

In a preprocessing step, we pre-compute a *supplementary mesh* from the point cloud using a greedy projection triangulation algorithm [42]. This supplementary mesh greatly facilitates visibility checking. In addition, observing that people tend to focus more on objects located at the center of a view, we add weights to pixels in the *supplementary mesh* view: each pixel p_i corresponds to a projected point; its weight is defined as $w_i = 1 - \lambda_w d_i^2$, where d_i is the normalized Euclidean distance between p_i and the center of the view, while λ_w is a small factor to help emphasize the central areas ($\lambda_w = 0.2$ in our system).

Our attributes are:

A_{pa}: *projected area.* This is a rather basic attribute and has been proved effective in previous works like Ref. [29]. Let $\mathcal{P} = \{P_n\}$ denote the set of *visible* patches in a view. Visibility is checked using the pre-computed supplementary mesh. A_{pa}, the sum of the projected areas of visible patches in view, is then defined as

$$A_{\mathrm{pa}} = S_{\mathcal{P}} = \sum_{P_n \in \mathcal{P}} S_n, \quad \text{with } S_n = \sum_{p_i \in P_n} w_i \tag{3}$$

where S_n is the projected area of a visible patch, where each pixel is weighted by the focus-attention weight w_i.

A_{ve}: *viewpoint entropy.* To estimate the richness of a view, we follow Vázquez et al. [30], and calculate view entropy on the supplementary mesh.

A_{pd}: *point density.* Due to the discrete nature of our point clouds, the distance between viewpoint and objects must be carefully set to provide the desired point density and contextual information in the view. We perform such a view-related density measurement by comparing the difference of views between the point cloud and the supplementary mesh. Let p'_i be a pixel of the point cloud view, and S'_i be the weighted areas that the pixels in the point cloud view and the mesh view belong to, for the same patch i. A_{pd} representing the average density of patches in view is then defined as

$$A_{\mathrm{pd}} = \frac{\sum_{P_n \in \mathcal{P}} \Omega(S'_n/S_n) S_n}{S_{\mathcal{P}}}, \quad \text{with } S'_n = \sum_{p'_i = p_i,\ p'_i \in P_n} w_i \tag{4}$$

where $\Omega(x) = 1/(1 + \mathrm{e}^{-\lambda_\Omega \cdot (x - t_6)})$ is a sigmoid function for evaluating the density of one patch. Intuitively, if S'_n/S_n exceeds a threshold t_6, the density is acceptable and allows users to recognize objects. In our system, we set $\lambda_\Omega = 16$ and $t_6 = 0.3$.

Besides the attributes listed above, we also explored additional attributes considering aesthetic perception, including: *depth information* ($A_{\mathrm{dd}}, A_{\mathrm{dc}}$), *object layout* ($A_{\mathrm{op}}, A_{\mathrm{od}}, A_{\mathrm{ov}}$), and *radial patches* (A_{rp}). Please refer to the Electronic Supplementary Material (ESM) for detailed formulations of these attributes. However, according to the user study, they are relatively dispensable.

User study. In order to analyze the importance of these attributes to our view preference model, we conducted a user study. We chose three scenes captured by different types of devices (Kinect V2 and LiDAR) to alleviate the influence of different input qualities. For each scene, we randomly sampled 60 views (using the sampling approach described in Section 5.1), including both top views (5%) and interior perspectives (95%). 16 participants with basic knowledge of computer graphics were recruited to manually evaluate the quality of each view. To ensure a proper comparative evaluation, each time, each participant was given a group of views from the same scene (6 views per group, randomly selected from 60 views) and was asked to rate each view on a scale from 1 (worst) to 5 (best). They were also told that views containing easily recognizable and multiple objects without severe occlusion should be given high scores. Representative views are shown in Fig. 5. Please refer to the ESM for other scenes and results.

Learning weights of attributes. After collecting the scores for each view, we studied the importance of attributes. We normalized the observed attributes and adopted lasso regression [43] to determine the weights β_i of the attributes: it is

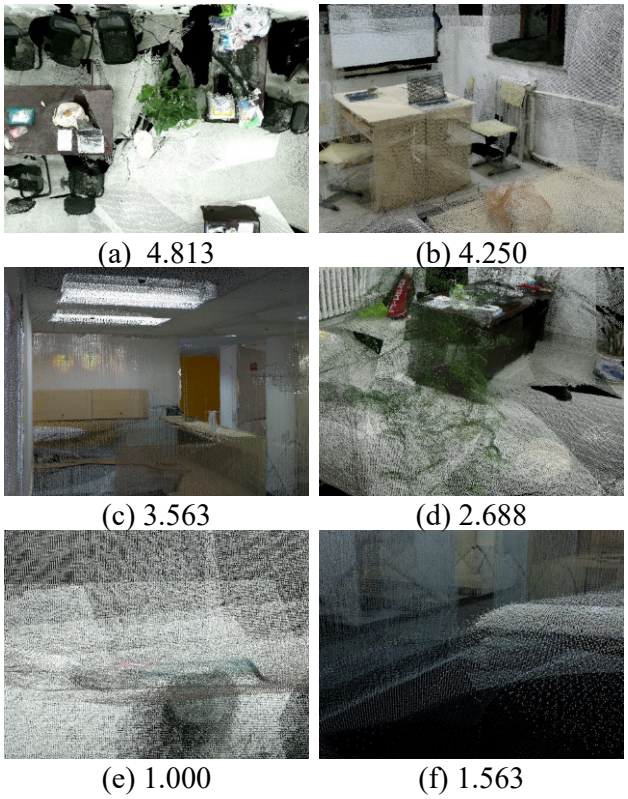


Fig. 5 Representative views from the user study. (a) High-score top view. (b, c) High-score interior views. (d) Low-score interior view (the tree seriously blocks the objects behind). (e, f) Low-score interior views.

able to perform variable selection in order to enhance the accuracy of the predictions by the statistical model it produces. The learned model by lasso regression is simply formulated as

$$\widehat{Y} = \sum_i \beta_i \widehat{A}_i + \beta_0 \tag{5}$$

where $\widehat{A}_i$ is the normalized attribute of a view, β_i is the learned weight for each attribute, and $\widehat{Y}$ is the average score for the view.

Lasso regression shows that among all the attributes included in the regression testing, *view entropy* and *point density* played the most significant roles, while other attributes were dispensable. Thus, we discarded these other attributes and performed lasso regression again. The final weights provided for *view entropy* and *point density* were 2.11 and 3.05, respectively. Detailed information about the regression process can be found in the ESM.

5.3 View suggestion

Given an input scene, we first sample views (see Section 5.1) in the preprocessing step. At runtime, we use the learned view preference model (Eq. (5)) to automatically suggest views. Specifically, we generate a set of candidate views, denoted by $\mathcal{V}$, by picking views with $A_{\text{pa}} > 0.6$. The view in $\mathcal{V}$ with the highest score ($V_0 = \underset{V_i \in \mathcal{V}}{\text{argmax}}\, Y_i$) is suggested as the first view.

Given a suggested view, the user may perform interactive segmentation (see Section 6) and then request another view. To avoid repetitive suggestions, each time a view is suggested, we lower its score by a scale factor 0.7. We also allow users to reject a current view. In this case our system will suggest new views. We suggest new views according to the following principles:

- View suggestions should respect the current segmentation status. Views with many already-segmented objects should have lower priority.
- A smooth transition between the current and the new views is preferred.
- View suggestion should take into account user rejection of suggested views.

To satisfy the first guideline, we update the score of each candidate view according to the current segmentation status. More specifically, we replace A_{ve} with the residual entropy A'_{ve}, which is calculated by removing the terms corresponding to already-labeled patches in A_{ve}, since such labeled patches provide little information. In contrast, A_{pd} remains unchanged because intuitively this term is used to clearly display point clouds to the screen, and thus is somewhat independent of labeling.

To encourage a smooth transition from V_n to V_{n+1}, we pick V_{n+1} according to the following rule:

$$V_{n+1} = \underset{V_i \in \mathcal{V}}{\text{argmax}}\, Y'_i \mathrm{e}^{-D(K_i, K_n)} \tag{6}$$

where Y'_x is the online score of view V_x, and $D(K_i, K_n)$ is the horizontal Euclidean distance between the current camera position K_n and the candidate position K_i. We use an exponential decay of preference to favor nearby rather than distant view candidates.

If a user rejects a suggested view, it generally means that unlabeled patches in this view are less important, and thus, their priority should be lowered when assessing subsequent suggestions. To do so, we lower the residual entropy A_{ve} of unlabeled patches in rejected views by a scale factor 0.8.

For multi-storey cases, our system begins with the highest storey and suggests views related to the current storey until the labeling progress of the current storey surpasses 90%. Once surpassed, our system will suggest the best view for the next lower storey and continue.

6 Interactive segmentation

We adopt a stroke-based interface similar to that in Ref. [14] for interactive segmentation. Given a view, users draw different strokes on unsegmented regions to indicate different objects (see Fig. 6(left)). Graph-cut-based optimization [44] is then used to achieve the desired segmentation results. Our work contributes to this paradigm by introducing support relations into graph-cut, and finding similar objects to further reduce the amount of user intervention.

6.1 Support relations for graph-cut

Previous graph-cut-based point cloud segmentation techniques (e.g., Refs. [7, 14]) mainly focus on the

Fig. 6 Graph-cut without (middle) and with (right) support relations. Left: input strokes on unsegmented objects in a given view.

use of low-level geometric features (e.g., position, normal, color) to propagate the segmentation information from the user-specified seeds to the rest of the scene. Since man-made objects often exhibit box-like shapes, due to the sharp change in normal between different faces of such objects, multiple strokes (one for each face) may be needed to segment a single object (e.g., a desk or monitor in Fig. 6(middle)).

To reduce the amount of user intervention, we introduce *support relations* as an additional measurement of distance between patches in the graph-cut formulation. This helps the reduction of both the number of user-specified strokes and the number of needed views. For example, with our support relations considered (see Fig. 6(right)), the user only needs to input strokes on the top faces of a desk and monitor in a top view to segment the objects. Without, the user would need to switch to different views to input further strokes.

Support relations are common in indoor scenes because of the influence of gravity. There are two levels of support relations: support between objects and support between primitive shapes which constitute an object. The former requires the availability of object-level segmentation (the output of our system) [11]. Our focus is on the second type of support relations to guide object-level segmentation.

We take the patches constructed in Section 4.1 as input. First we classify all patches into two types: quasi-vertical Q_{v} and quasi-horizontal Q_{h} according to the angles between the planes fitted to them and the ground plane (relative to 45°). Then we concentrate on two common relationships: a quasi-vertical patch supporting an quasi-horizontal patch Q_{vh} above it, and a quasi-vertical patch being supported by a quasi-horizontal patch Q_{hv} below it, as shown in Fig. 7.

For each pair of an adjacent quasi-vertical patch P_i and a quasi-horizontal patch P_j (above P_i), the likelihood that patches P_i and P_j have a Q_{vh} relationship is defined as follows:

$$Q_{\mathrm{vh}}\langle P_i, P_j\rangle = 1 - \lambda_Q^{U(P_i,P_j)} \tag{7}$$

where $U(P_i, P_j)$ is the number of quasi-vertical patches within the neighborhood of P_i (including P_i) which have P_j as an upper patch and $\lambda_Q = 0.2$ is a fixed parameter. Intuitively, more quasi-vertical patches supporting the same quasi-horizontal patch lead to higher likelihood of having the Q_{vh} relationship.

Similarly, for each pair of an adjacent quasi-vertical patch P_i and a quasi-horizontal patch P_j (below P_i), the likelihood that patches P_i and P_j have a Q_{hv} relationship is defined as follows:

$$Q_{\mathrm{hv}}\langle P_i, P_j\rangle = \begin{cases} 0, & U(P_i, P_k) = 0, P_k \in Q_{\mathrm{h}} \\ \frac{\min(W_i, W_j)}{\max(W_i, W_j)}, & \text{otherwise} \end{cases} \tag{8}$$

where W_i is the area of the convex hull of points in patch P_i. We consider a Q_{hv} relationship for a quasi-vertical patch P_i only when there is no Q_{vh} relationship involving P_i. Consider Fig. 7(middle): the laptop lid has Q_{hv} relationship with the table, the laptop base, and the keyboard, while the laptop base leads to the highest likelihood since they have about the same size.

Then we define $\mathcal{T}_{\mathrm{S}}(P_i, P_j)$, the likelihood for patches P_i and P_j to belong to the same object using inference from the support map, as follows:

$$\mathcal{T}_{\mathrm{S}}(P_i, P_j) = \begin{cases} Q_x\langle P_i, P_j\rangle, & P_i \in Q_{\mathrm{v}}, P_j \in Q_{\mathrm{h}} \\ Q_x\langle P_j, P_i\rangle, & P_i \in Q_{\mathrm{h}}, P_j \in Q_{\mathrm{v}} \\ \max\limits_k \left(Q_x\langle P_i, P_k\rangle \cdot Q_x\langle P_j, P_k\rangle\right), & P_{i,j} \in Q_{\mathrm{v}}, P_k \in Q_{\mathrm{h}} \end{cases} \tag{9}$$

where Q_x can be either Q_{vh} or Q_{hv}, decided by the spatial relationship between two concerning patches. The top two conditions are essentially Q_{vh} or Q_{hv}, respectively, and the third indicates co-support relations between patches in Q_{v}, which exist if and only if two vertical patches have the same type of support relations (Q_x) with the same horizontal patch (see the example in Fig. 7(left)). Since support relations between patches are irrelevant to the segmentation status, in our system, they are computed during the preprocessing step.

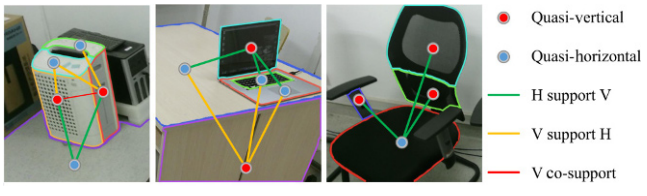


Fig. 7 Automatically detected support relations. Left: the quasi-vertical patches co-support the top quasi-horizontal patches.

Graph-cut. Given a set of user-specified strokes, only pixels with the same patch indices on both the point cloud view and the supplementary mesh view are regarded as valid, to avoid penetration

labeling. We then formulate support relations as well as distances, colors, and normals in the graph-cut formulation. Please refer to the ESM for details.

6.2 Finding similar objects

Duplicate objects often exist in a scene. It is redundant for a user to label them one by one. Our system employs a simple algorithm to automatically retrieve candidate objects similar to a given object in the scene. If the found objects are in fact not similar, users can delete the automatically generated labels.

Several methods for finding similar objects in point cloud scenes have already been proposed. For example, Kim et al. [9] introduced a method by first learning models of frequently occurring objects and then performing real-time matching. Mattausch et al. [24] took patch similarity and spatial consistency into account and automatically found all similar objects in a scene by clustering, which, however, takes dozens of seconds for large-scale indoor scenes.

We aim for a simple and efficient tool that can perform in real time, without pre-trained models or clustering. Our approach is based on the key assumption that similar objects should contain similar support structures, i.e., both the support relations and the accompanying patches should be similar. This assumption allows us to identify similar objects by straightforwardly comparing all unlabeled candidate support structures with a given object, based on patch similarity and support relations.

More specifically, given an already segmented object, denoted by $\mathcal{C}_0$, we first identify all candidate patches which have not been segmented and have a high similarity ($t_7 = 0.3$ in our implementation) to at least one patch in $\mathcal{C}_0$. For efficiency, we slightly change the similarity metric in Ref. [24] to define our similarity $\mathcal{A}_{i,j}$ for a given pair of patches P_i and P_j (see the ESM).

Then we group these candidate patches by connectedness to form patch constellations. For each pair of patches $\langle P_i, P_j\rangle$ in each constellation $\mathcal{C}_n$, we search for a corresponding pair $\langle P_k, P_l\rangle \in \mathcal{C}_0$ with the same type of support relation. Based on our key assumption, we compute the similarity likelihood $\mathrm{Sim}(\mathcal{C}_n, \mathcal{C}_0)$ between $\mathcal{C}_n$ and $\mathcal{C}_0$ as follows:

$$\mathrm{Sim}(\mathcal{C}_n, \mathcal{C}_0) = 1 - \prod_{P_i, P_j \in \mathcal{C}_n} (1 - \mathcal{I}_{i,j}) \qquad (10)$$

where $\mathcal{I}_{i,j}$ is the *isomorphism likelihood* of the support structure between $\langle P_i, P_j\rangle$ and $\langle P_k, P_l\rangle$, calculated as $\mathcal{I}_{i,j} = (\min_{k,l} \mathcal{A}_{i,k})\mathcal{A}_{j,l}\mathcal{T}_{\mathrm{S}}(P_i, P_j)\mathcal{T}_{\mathrm{S}}(P_k, P_l)$. Here $\mathcal{A}_{i,k}$ and $\mathcal{A}_{j,l}$ represent the similarity between P_i and P_k and between P_j and P_l respectively.

If the similarity likelihood $\mathrm{Sim}(\mathcal{C}_n, \mathcal{C}_0)$ exceeds a threshold t_8 (= 0.25 in our implementation), we take it as a candidate similar object. We then use all patches in the found isomorphic support structures as seeds for a new object to perform another round of graph-cut.

As shown in Fig. 8 our algorithm manages to find several similar objects in two cases. In the left example, each chair is composed of different numbers of patches, but the algorithm is still able to find such similar isomorphic parts and perform graph-cut. The right example shows our ability to find different types of objects: tables (*green*), monitors (*cyan*), telephones (*yellow*), and keyboards (*purple*). Note that objects adjacent to some keyboards are mistaken by graph-cut, demanding additional strokes for refinement.

7 Results and evaluation

We have tested our system on different qualities of datasets from different sources. We also performed a user study to evaluate the performance of view suggestion. Please see the accompanying video in the ESM for interactive demos. Some representative suggested views and segmentation results are shown in Fig. 9.

Datasets. In order to test our performance of view suggestion and segmentation, we chose middle- or large-scale scenes with rich sets of objects. Specifically, we used 2 out of 3 scenes (office2, office3) in the room detection datasets of UZH [24], 1 of 5 scenes (office1) in the Floored Panorama dataset [37], and 2 of 2 synthetic scenes (office-room, living-room) with ground-truth trajectory in the ICL-NUIM dataset [45]. The ICL-NIUM data

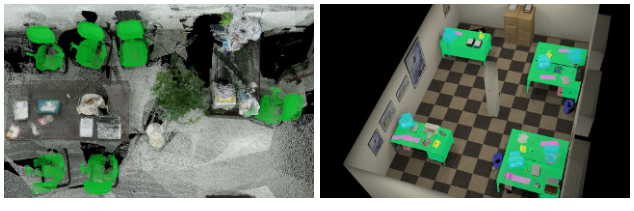


Fig. 8 Results of finding similar objects (shown in the same color).

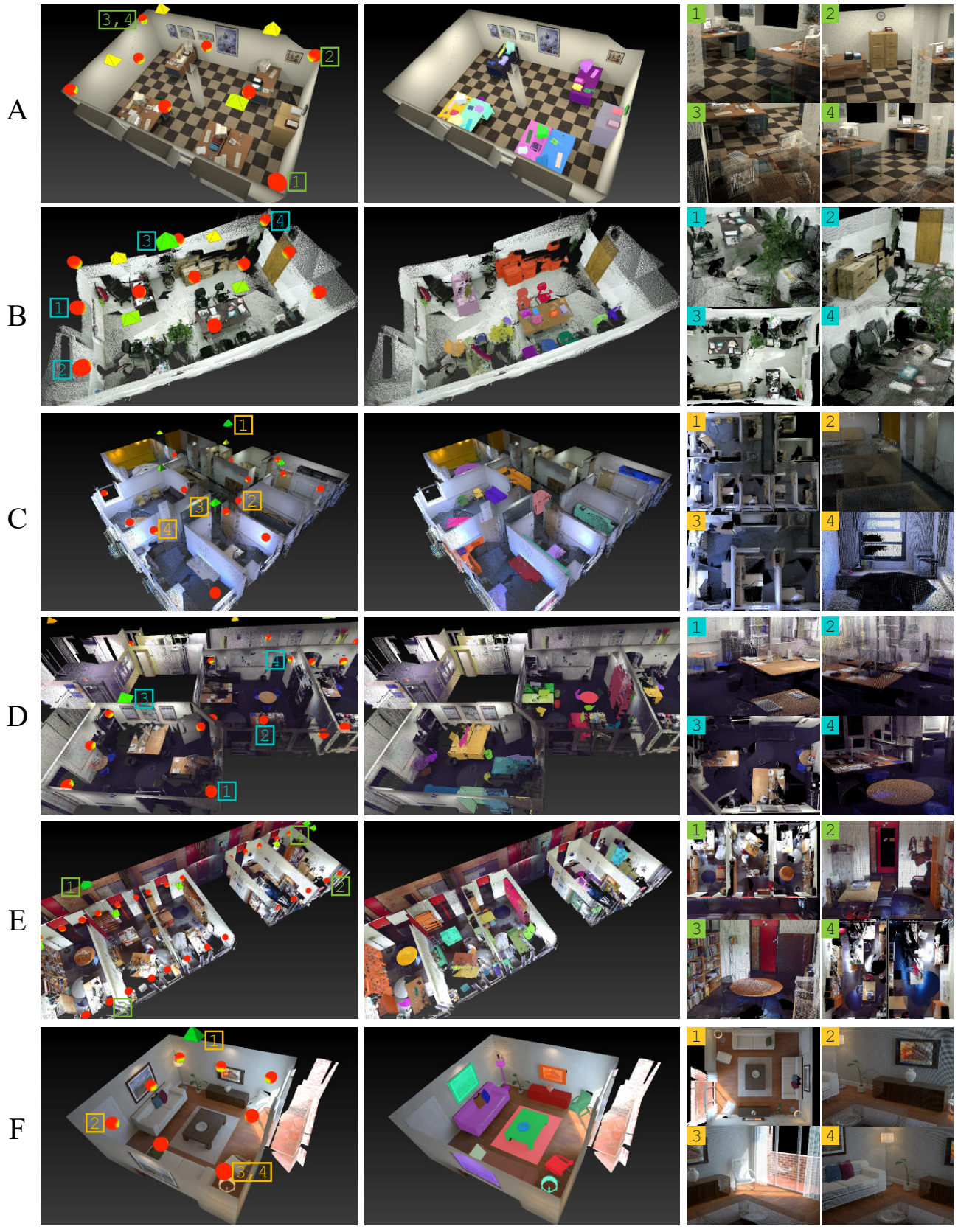
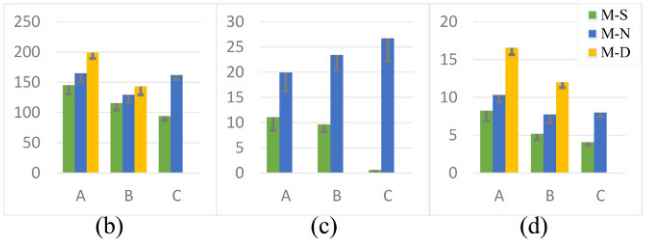


Fig. 9 Examples of results of our system. Left: view score map. Right: high score examples when no objects are labeled.

	Acc.	p_n^T	p_d^T	p_n^N	p_n^M	p_n^V	p_d^V
A	83.9%	**0.0682**	0.0015	0.0031	0.0270	0.0127	<0.0001
B	85.8%	**0.1379**	0.0042	0.0003	0.0247	0.0044	<0.0001
C	93.1%	<0.0001	—	0.0001	**0.0527**	< 0.0001	—

(a)



(b) (c) (d)

Fig. 10 Results of user study. Top, left: scenes with pre-defined target segmentations. (a) Acceptance rate of suggestions and hypothesis test results. The p-values are from paired sample t-tests between our method and no suggestion (n) or RGB-D (d), giving full time (T), the time spent on manual navigation (N), the counts of minimap navigation (M) and used views (V). Right: average and standard deviation of the means of observed values. (b) Time spent on the whole process in second. (c) Time spent on manual navigation in second. (d) The count of used views.

was generated by sampling point on surfaces of existing 3D mesh-based models, while others were acquired with LiDAR (with color information). Figure 9 illustrates all scenes tested. We additionally collected a scene captured by Microsoft Kinect2 to test for poor-quality scans.

7.1 Validity of view suggestion

To evaluate the effectiveness of view suggestion, we compared the performance of interactive segmentation with and without view suggestion. We also compared to stroke-based segmentation directly on RGB-D frames if available. These three methods, denoted as **M-S**, **M-N**, and **M-D**, were compared in a user study. Please refer to the accompanying video in the ESM to see how each method works.

Interfaces. For each of the methods compared, we provided a minimap for easy navigation. For **M-D**, while interactive segmentation was done on RGB-D images, we provided an additional point cloud viewer for examining the segmentation status. The only difference between **M-S** and **M-N** was the availability of view suggestion. For **M-S**, we encouraged the participants to use the view suggestion feature as much as possible, though minimap navigation and common 3D browsing were still allowed. All methods were tested in full-screen mode (1080 p) to eliminate the influence of window size. For fair comparison, we used the same preprocessing, graph-cut, and similar-object-finding algorithms for all methods.

Participants and apparatus. We recruited 12 university students, who had basic knowledge of graphics but different levels of experience in 3D browsing. In a training session, which lasted nearly 15 minutes on average, each participant was briefed and trained how to use the three methods. The entire user study was conducted on the same PC with i7 2.10 GHz CPU, a GTX 660Ti GPU, and 16 GB RAM.

Tasks. We prepared 3 scenes with target segmentations. Scene A in Fig. 10(a) came from the ICL-NUIM dataset with 42 objects and 35 RGB-D key-frames. Scene B was our own recorded Kinect2 data, including 22 objects and 33 key-frames. Scene C was a large-scale multiple-room scene with 21 objects but no associated RGB-D images. These scenes are given in rows 1–3, respectively, in Fig. 9. The participants were asked to interactively segment these scenes using the three methods, whose presentation was counterbalanced to alleviate bias due to familiarity with the scenes. For Scene C, we tested **M-S** and **M-N** only, due to its lack of RGB-D frames. We pre-defined a ground-truth segmentation for each scene with manually labeled objects and required the participants to reproduce the ground-truth segmentations (shown on another monitor simultaneously) as closely as possible. Since it was difficult to reproduce the segmentations exactly, each scene was considered as completed if the similarity (ratio of patches) between the current and target segmentations reached 90%.

Measures. Our system recorded the following information for comparison: time spent on manual navigation, processing, and the whole progress (including planning, drawing, processing, and navigation); number of uses of view suggestion and minimap navigation operations; number of views used; number of executions of graph-cut and similar-object-finding algorithms; number and total length of input strokes. Additionally, we also recorded the acceptance ratio of view suggestion. We conducted a questionnaire survey at the end of the study.

Results. The acceptance rates of our suggested

views for Scenes A, B, and C were 83.9%, 85.8%, and 93.1%, respectively, indicating the good quality of suggested views. Figures 10(b)–10(d) show the statistics of some observed values. The table in Fig. 10(a) gives evaluation and hypothesis testing results, where p is the p-value calculated through student's t-test [46], which is helpful for evaluating statistically significant differences among small-scale paired-samples.

Using one-way analysis of variance (ANOVA), we found a significant difference in the average total time between the tested methods ($p = 0.0288$ for all methods on Scenes A and B, $p = 0.0058$ for **M-S** and **M-N** on all the scenes). In all tested scenes, our method significantly outperformed **M-D** ($p < 0.005$ from t-test) in both the number of views used and the total time. In Scene C, our method was significantly faster than **M-N** ($p < 0.001$), although they achieved comparable performance on minimap navigation ($p = 0.0527$). We observed that minimap navigation is crucial for large-scale scenes. In Scenes A and B, compared to **M-N**, our method significantly reduced the time for camera manipulation ($p < 0.005$) and the usage of minimap navigation ($p < 0.03$), but achieved comparable performance in total time ($p > 0.05$). This might be because the expense of manual navigation in such single-room scenes is relatively low. Our method also used significantly fewer views than **M-N** ($p < 0.02$) when segmenting all the tested scenes.

Our participants reported that with **M-D**, the RGB-D image views were more useful for object recognition due to their dense views, but (re-) checking the labeled objects, either through back-and-forth traversal or with the help of the auxiliary point cloud view, was not convenient. Worse, the number of RGB-D key-frames can significantly increase as scenes get bigger. Some participants also reported that they subconsciously referred to the good RGB-D views or results of view suggestion to improve labeling efficiency when using **M-N**. In fact, the performance of interactive segmentation without view suggestion can be easily affected by familiarity with the scenes and the proficiency of 3D browsing.

7.2 Robustness and efficiency

Parameters. In the preprocessing step for extracting storeys and sampling viewpoints, we used different sets of parameters for different datasets due to their different data quality and scene scale. The other parameters for patch construction, view suggestions, and interactive segmentation, remained unchanged for all the tested scenes.

Timing. On the same machine used in the user study, the preprocessing stage can cost dozens of minutes for the tested scenes. For a 140 m^2 one-floor room with 16 million points, it took about 1 minute per viewpoint to traverse through scenes to get the descriptors and 30 minutes in total; this could potentially be made faster by switching to quicker rendering tools. Since the score of each view can be calculated separately, we can also easily accelerate the method by parallelization. In the online stage, it only cost 50 ms to suggest an appropriate view, the graph-cut process only took 100 ms, and finding similar objects took 200 ms with respect to a labeled object. In summary, the online processing costs meet the requirement of real-time response.

Limitations. In some scenes with complicated objects such as plants or trees, our system may "prefer" to suggest views involving such objects (Fig. 11(left)), since they are taken as fragmentary patches (each including several leaves or stems) in patch construction, leading to high residual entropy on these views. However, once such objects are labeled, their influence will disappear. Since patches are indivisible in our implementation, if two or more objects are gathered into one patch (e.g., due to similar color and depth), it is impossible to segment the individual objects. Also the boundaries of objects may not be accurate if the points on boundaries are ambiguous. This might be addressed by introducing an interactive repair tool for boundary refinement. In addition, our proposed support relations and the algorithm for finding similar objects may fail if objects are combined as multiple discrete components (see Fig. 11(right)). Fortunately, users can modify the labeling by providing additional

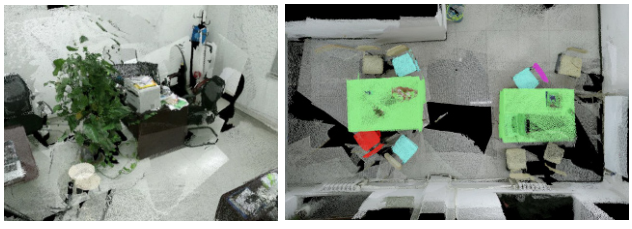


Fig. 11 Less successful cases. Left: an example view with a tree with a high score. Right: failure to find similar objects in isolated patches (the chair in red).

inputs for segmentation.

8 Conclusions and future work

We have presented a novel system for interactive segmentation of large-scale indoor scenes, represented as cluttered point clouds. The key contributions of our work are the problem and solution of automatic view suggestion for interactive segmentation. Other contributions include support relations for graph-cut-based segmentation and finding similar objects. Our extensive evaluations show the advantages of our approach over alternative methods. In future, besides addressing the discussed limitations, we are also interested in applying a similar idea of view suggestion for other applications, e.g., interactive editing of large-scale scenes.

Acknowledgements

This work was supported by the Joint NSFC–ISF Research Program (Project No. 61561146393), the National Natural Science Foundation of China (Project No. 61521002), the Research Grant of Beijing Higher Institution Engineering Research Center, and the Tsinghua–Tencent Joint Laboratory for Internet Innovation Technology. The work was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CityU113513 and CityU11300615).

Electronic Supplementary Material Supplementary material is available in the online version of this article at http://dx.doi.org/10.1007/s41095-017-0078-4.

References

[1] Lai, K.; Bo, L.; Ren, X.; Fox, D. Detection-based object labeling in 3D scenes. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1330–1337, 2012.

[2] Johnson, A. E.; Hebert, M. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 21, No. 5, 433–449, 1999.

[3] Zheng, B.; Zhao, Y.; Yu, J. C.; Ikeuchi, K.; Zhu, S. C. Beyond point clouds: Scene understanding by reasoning geometry and physics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3127–3134, 2013.

[4] Holz, D.; Behnke, S. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In: *Intelligent Autonomous Systems 12*. Lee, S.; Cho, H.; Yoon, K.-J.; Lee, J. Eds. Springer Berlin Heidelberg, 61–73, 2013.

[5] Rabbani, T.; van den Heuvel, F. A.; Vosselmann, G. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* Vol. 36, No. 5, 248–253, 2006.

[6] Boykov, Y.; Funka-Lea, G. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision* Vol. 70, No. 2, 109–131, 2006.

[7] Golovinskiy, A.; Funkhouser, T. Min-cut based segmentation of point clouds. In: Proceedings of the IEEE 12th International Conference on Computer Vision Workshops, 39–46, 2009.

[8] Sedlacek, D.; Zara, J. Graph cut based point-cloud segmentation for polygonal reconstruction. In: *Advances in Visual Computing*. Bebis, G.; Boyle, R.; Parvin, B.; Koracin, D. et al. Eds. Springer Berlin Heidelberg, 218–227, 2009.

[9] Kim, Y. M.; Mitra, N. J.; Yan, D.-M.; Guibas, L. Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 138, 2012.

[10] Nan, L.; Xie, K.; Sharf, A. A *search-classify* approach for cluttered indoor scene understanding. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 137, 2012.

[11] Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor segmentation and support inference from RGBD images. In: *Computer Vision–ECCV 2012*. Fitzgibbon, A.; Lazebnik, S.; Perona, P.; Sato, Y.; Schmid, C. Eds. Springer Berlin Heidelberg, 746–760, 2012.

[12] Nguyen, A.; Le, B. 3D point cloud segmentation: A survey. In: Proceedings of the 6th IEEE Conference on Robotics, Automation and Mechatronics, 225–230, 2013.

[13] Shao, T.; Xu, W.; Zhou, K.; Wang, J.; Li, D.; Guo, B. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 136, 2012.

[14] Yuan, X.; Xu, H.; Nguyen, M. X.; Shesh, A.; Chen, B. Sketch-based segmentation of scanned outdoor environment models. In: Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling, 19–26, 2005.

[15] Silberman, N.; Fergus, R. Indoor scene segmentation using a structured light sensor. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 601–608, 2011.

[16] Xiao, J.; Owens, A.; Torralba, A. SUN3D: A database of big spaces reconstructed using SfM and object labels. In: Proceedings of the IEEE International Conference on Computer Vision, 1625–1632, 2013.

[17] Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; Yu, F. ShapeNet: An information-rich 3D model repository. *arXiv preprint* arXiv:1512.03012, 2015.

[18] Hinterstoisser, S.; Lepetit, V.; Ilic, S.; Holzer, S.; Bradski, G. R.; Konolige, K.; Navab, N. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: *Computer Vision–ACCV 2012*. Lee, K. M.; Matsushita, Y.; Rehg, J. M.; Hu, Z. Eds. Springer Berlin Heidelberg, 548–562, 2012.

[19] Chen, K.; Lai, Y.-K.; Wu, Y.-X.; Martin, R.; Hu, S.-M. Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information. *ACM Transactions on Graphics* Vol. 33, No. 6, Article No. 208, 2014.

[20] Silberman, N.; Sontag, D.; Fergus, R. Instance segmentation of indoor scenes using a coverage loss. In: *Computer Vision–ECCV 2014*. Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer International Publishing, 616–631, 2014.

[21] Chen, K.; Lai, Y. K.; Hu, S.-M. 3D indoor scene modeling from RGB-D data: a survey. *Computational Visual Media* Vol. 1, No. 4, 267–278, 2015.

[22] Shen, C.-H.; Huang, S.-S.; Fu, H.; Hu, S.-M. Adaptive partitioning of urban facades. *ACM Transactions on Graphics* Vol. 30, No. 6, Article No. 184, 2011.

[23] Zhang, H.; Xu, K.; Jiang, W.; Lin, J.; Cohen-Or, D.; Chen, B. Layered analysis of irregular facades via symmetry maximization. *ACM Transactions on Graphics* Vol. 32, No. 4, Article No. 121, 2013.

[24] Mattausch, O.; Panozzo, D.; Mura, C.; Sorkine-Hornung, O.; Pajarola, R. Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum* Vol. 33, No. 2, 11–21, 2014.

[25] Valentin, J.; Vineet, V.; Cheng, M.-M.; Kim, D.; Shotton, J.; Kohli, P.; Nießner, M.; Criminisi, A.; Izadi, S.; Torr, P. SemanticPaint: Interactive 3D labeling and learning at your fingertips. *ACM Transactions on Graphics* Vol. 34, No. 5, Article No. 154, 2015.

[26] Wong, Y.-S.; Chu, H.-K.; Mitra, N. J. SmartAnnotator an interactive tool for annotating indoor RGBD images. *Computer Graphics Forum* Vol. 34, No. 2, 447–457, 2015.

[27] Christie, M.; Olivier, P. Camera control in computer graphics: Models, techniques and applications. In: Proceedings of the ACM SIGGRAPH ASIA 2009 Courses, Article No. 3, 2009.

[28] Scott, W. R.; Roth, G.; Rivest, J.-F. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys* Vol. 35, No. 1, 64–96, 2003.

[29] Secord, A.; Lu, J.; Finkelstein, A.; Singh, M.; Nealen, A. Perceptual models of viewpoint preference. *ACM Transactions on Graphics* Vol. 30, No. 5, Article No. 109, 2011.

[30] Vázquez, P.-P.; Feixas, M.; Sbert, M.; Heidrich, W. Viewpoint selection using viewpoint entropy. In: Proceedings of the Vision Modeling and Visualization Conference, 273–280, 2001.

[31] Andújar, C.; Vázquez, P.; Fairén, M. Way-Finder: Guided tours through complex walkthrough models. *Computer Graphics Forum* Vol. 23, No. 3, 499–508, 2004.

[32] Li, T.-Y.; Lien, J.-M.; Chiu, S.-Y.; Yu, T.-H. Automatically generating virtual guided tours. In: Proceedings of the Computer Animation, 99–106, 1999.

[33] Christie, M.; Languénou, E. A constraint-based approach to camera path planning. In: *Smart Graphics*. Butz, A.; Krüger, A.; Olivier, P. Eds. Springer Berlin Heidelberg, 172–181, 2003.

[34] Salomon, B.; Garber, M.; Lin, M. C.; Manocha, D. Interactive navigation in complex environments using path planning. In: Proceedings of the Symposium on Interactive 3D Graphics, 41–50, 2003.

[35] Choi, S.; Zhou, Q.-Y.; Koltun, V. Robust reconstruction of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5556–5565, 2015.

[36] Newcombe, R. A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A. J.; Kohli, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality, 127–136, 2011.

[37] Ikehata, S.; Yang, H.; Furukawa, Y. Structured indoor modeling. In: Proceedings of the IEEE International Conference on Computer Vision, 1323–1331, 2015.

[38] Furukawa, Y.; Curless, B.; Seitz, S. M.; Szeliski, R. Manhattan-world stereo. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, 1422–1429, 2009.

[39] Asha, V.; Bhajantri, N. U.; Nagabhushan, P. GLCM-based chi-square histogram distance for automatic detection of defects on patterned textures. *International Journal of Computational Vision and Robotics* Vol. 2, No. 4, 302–313, 2011.

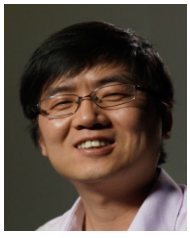
[40] Früh, C.; Zakhor, A. Constructing 3D city models by merging aerial and ground views. *IEEE Computer Graphics and Applications* Vol. 23, No. 6, 52–61, 2003.

[41] Fisher, M.; Savva, M.; Li, Y.; Hanrahan, P.; Nießner, M. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Transactions on Graphics* Vol. 34, No. 6, Article No. 179, 2015.
[42] Marton, Z. C.; Rusu, R. B.; Beet, M. On fast surface reconstruction methods for large and noisy point clouds. In: Proceedings of the IEEE International Conference on Robotics and Automation, 3218–3223, 2009.
[43] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society* Vol. 58, No. 1, 267–288, 1996.
[44] Boykov, Y.; Veksler, O.; Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 23, No. 11, 1222–1239, 2001.
[45] Handa, A.; Whelan, T.; McDonald, J.; Davison, A. J. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1524–1531, 2014.
[46] Gosset, W. S. The probable error of a mean. *Biometrika* Vol. 6, No. 1, 1–25, 1908.

Sheng Yang received his B.S. degree in computer science from Wuhan University in 2014. He is currently a Ph.D. candidate in computer science in Tsinghua University. His research interests include computer graphics and point cloud processing.

Jie Xu is a Ph.D. student at the Computer Science and Artificial Intelligence Laboratory in Massachusetts Institute of Technology. His research interests include computer graphics and geometric processing.

Kang Chen received his B.S. degree in computer science from Nanjing University in 2012. He is currently a Ph.D. candidate in the Institute for Interdisciplinary Information Sciences, Tsinghua University. His research interests include computer graphics, geometric modeling and processing.



Hongbo Fu is an associate professor in the School of Creative Media, City University of Hong Kong. He received his Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 2007 and B.S. degree in information sciences from Peking University in 2002. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an associate editor of *The Visual Computer*, *Computers & Graphics*, and *Computer Graphics Forum*.