Semantic Labeling and Instance Segmentation of 3D Point Clouds using Patch Context Analysis and Multiscale Processing

Shi-Min Hu, *Senior Member, IEEE,* Jun-Xiong Cai, Yu-Kun Lai, *Member, IEEE*

Abstract—We present a novel algorithm for semantic segmentation and labeling of 3D point clouds of indoor scenes, where objects in point clouds can have significant variations and complex configurations. Effective segmentation methods decomposing point clouds into semantically meaningful pieces are highly desirable for object recognition, scene understanding, scene modeling, etc. However, existing segmentation methods based on low-level geometry tend to either under-segment or over-segment point clouds. Our method takes a fundamentally different approach, where semantic segmentation is achieved along with labeling. To cope with substantial shape variation for objects in the same category, we first segment point clouds into surface patches and use unsupervised clustering to group patches in the training set into clusters, providing an intermediate representation for effectively learning patch relationships. During testing, we propose a novel patch segmentation and classification framework with multiscale processing, where the local segmentation level is automatically determined by exploiting the learned cluster based contextual information. Our method thus produces robust patch segmentation and semantic labeling results, avoiding parameter sensitivity. We further learn object-cluster relationships from the training set, and produce semantically meaningful object level segmentation. Our method outperforms state-of-the-art methods on several representative point cloud datasets, including S3DIS, SceneNN, Cornell RGB-D and ETH.

Index Terms—Segmentation, Labeling, Clustering, Patch Context, Semantics, Scene Understanding

1 Introduction

With the development of 3D acquisition techniques, in particular low-cost RGB-D cameras such as the Kinect [1] and Matterport cameras [2], 3D data becomes much more available. Multiple scans are usually needed to capture objects due to occlusions. When scans are aligned and fused, the resulting data is usually in the form of unstructured point clouds, making it a universal representation for scanned 3D data. Therefore, point cloud based scene understanding is essential for real-world applications such as autonomous driving, housekeeping robots, and virtual reality.

Machine learning has achieved great success on 2D image understanding. However, the point cloud of a typical 3D scene contains complex details, requiring millions of points, which are significantly more than the number of pixels in typical 2D images for machine learning. Moreover, the point distribution is *unstructured*, making it difficult for many deep learning methods, especially those based on convolutional neural networks (CNNs). To handle these problems, many existing works [3], [4], [5], [6], [7] use 3D voxel grids to represent 3D shapes on which CNNs can operate. However, the voxel resolution and shape precision for these methods are highly restricted due to the extra dimension. Therefore, they are only suitable for smaller-scale object-level 3D model analysis.

- *Shi-Min Hu and Jun-Xiong Cai are with the Department of Computer Science and Technology, Tsinghua University, China.*
  E-mail: shimin@tsinghua.edu.cn, caijunxiong000@163.com
- *Yu-Kun Lai is with the School of Computer Science & Informatics, Cardiff University, UK. E-mail: Yukun.Lai@cs.cardiff.ac.uk*

Recently, pioneering work PointNet [8] and PointNet++ [9] made a good attempt to learn CNNs directly on point clouds. They take unordered points as classification units. When applied to scene labeling, such methods split the whole point cloud into blocks with a reasonable number of points (e.g. 4096) to make it more manageable. However, suitable block partitioning can be difficult to determine. The work [10] uses a multiscale approach to make the PointNet block size more flexible. Nevertheless, sampled points are unordered, which restricts the pooling operations and the use of blocks also limits the effectiveness of learning context from the training data. By exploiting patch cluster based context information and multiscale processing, our method achieves an accuracy of 88.0% on the S3DIS (Stanford Large Scale 3D Indoor Spaces) dataset [2] which outperforms the PointNet method by 13%.

In this paper, we propose a novel learning based approach to semantic labeling and segmentation of point clouds. In particular, we use indoor scenes as examples because they are challenging due to occlusion, clutter and diverse object types, and have important real-world applications such as robot navigation and modeling. We combine an unsupervised clustering of patches with a supervised approach to learn reliable contextual information involving both patch clusters and object labels. This intermediate patch cluster representation allows each object category to contain highly diverse patches, and reliable contextual relationships to be learned as similar patches are considered together. Furthermore, to avoid over/under-segmentation in patch segmentation, instead of treating patch segmentation and labeling as two separate problems, we propose an effective multiscale approach that performs joint patch

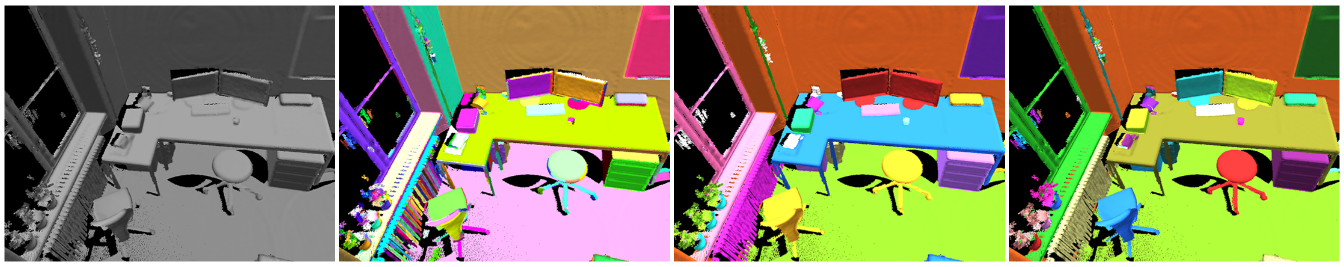

(a) input point cloud (b) patch segmentation (c) semantic labeling (d) semantic segmentation

Figure 1. Results of our semantic segmentation and labeling. Given the input point cloud (a), our method exploits patch clusters and patch cluster based contextual information learned from the training set to perform robust multiscale patch based segmentation (b). This provides a basis for (c) semantic labeling (with different colors corresponding to objects of different types) and (d) semantic instance segmentation (where colors are used to differentiate individual instances, regardless of their types).

segmentation and labeling, with suitable levels guided by learned contextual information. We further learn object-cluster relationships and use them to guide semantic segmentation, breaking down a whole 3D point cloud into individual object instances. The work [11] proposes a context-based method for object segmentation of images captured in uncontrolled environments, demonstrating the effectiveness of context. Yi et al. [12] propose a clustering-based method which uses hierarchical part segmentation for learning part-label context. However, their method requires explicit part level labeling that is usually not available for indoor scene data. Moreover, the context among objects in indoor scenes is less strict than context among object parts, and is therefore more complicated. Our method uses patch context analysis to provide a uniform framework for learning part-part, part-label and label-label context implicitly, avoiding the need for part level labeling.

To work effectively with a small amount of training data, we focus on the geometry information only, as color information can often vary substantially even for objects of the same type (e.g. a chair can be in wood color or painted white). An example of our patch segmentation, semantic labeling and semantic segmentation is given in Fig. 1. When segmenting the input point cloud into patches, our patch segmentation method with locally adapted levels reduces over-/under-segmentation. Based on this, we use the learned model to produce semantic labeling which assigns object labels to individual patches, as well as semantic segmentation which partitions the input point cloud into meaningful individual objects.

The main contribution of this paper is an efficient and effective learning based approach to semantic labeling and instance segmentation of unstructured 3D point cloud data. We introduce patch clusters as an intermediate representation for effectively representing and learning contextual information. Multiscale patch segmentation processing further helps segment point clouds in a suitable level and make the method generalize well to datasets with different characteristics. In the training stage, cluster-label relationships and cluster-object relationships are learned, which are used to guide joint patch segmentation and classification, with locally adapted segmentation levels automatically determined. Our learning based method for semantic segmentation and labeling works effectively for cluttered point cloud data with a small amount of training data.

Compared with traditional point cloud segmentation, our method achieves semantic level segmentation, avoiding over/under segmentation. Compared with semantic labeling, our method is efficient, works directly on unstructured point clouds, and outperforms state-of-the-art methods for both labeling accuracy and efficiency.

We also produce a point cloud benchmark based on the ETH dataset with manually labeled ground truth data at the point level for evaluation of both instance segmentation and semantic labeling techniques, which will be available at https://cg.cs.tsinghua.edu.cn/semantic.

2 Related Work

Geometry-based shape segmentation. Shape segmentation is an important problem for many downstream applications, including object recognition, scene modeling, robot navigation, etc. Traditional methods for point cloud segmentation focuses on the geometric information which can be directly analyzed from the input point cloud. When typical geometric primitives such as planes, cylinders etc. are of interest, segmentation can be achieved by explicitly fitting geometric primitives to the input point cloud [13]. For general unstructured point clouds, it is common practice to form a connected graph based on the neighborhood of each point. Point cloud segmentation can then be achieved by either pursuing coherence within each segment, often using region growing [14], or by analyzing clues for region boundaries which can be formulated as a graph-cut problem [15]. For segmenting objects into parts, van Kaick et al. [16] use convexity clues and shape signatures. Schnabel et al. [17] use an efficient RANSAC algorithm for shape segmentation. Attene et al. [18] perform hierarchical analysis for sampled surface point clouds. All these geometry based methods however do not take semantics into account. As a result, they may produce results with over-segmentation or under-segmentation, and often resort to user interaction to help guide the segmentation when high quality results are needed. For mesh segmentation where the input is a complete 3D mesh object and the purpose is to partition it into meaningful parts [19], learning based methods [20] have shown significant advantage of approaching semantic segmentation. The work [21] exploits co-analysis to improve mesh segmentation, demonstrating the benefits of building statistical models of shape parts. The method is unsupervised and therefore only segments shapes into parts based on geometric similarity, without assigning detailed semantic classes. Therefore, although aiming for semantic part segmentation, in their paper, the method is only applied to a collection of high-quality 3D meshes representing shapes of the same category and containing similar parts. The method cannot be directly generalized to process 3D scene point clouds which have much noise and occlusion, and contain objects with flexible layout. Moreover, each object category can have substantial variations of geometric shapes. In this work, we propose a novel supervised learning based method for semantic segmentation and labeling of complex 3D indoor scenes by exploiting shape and contextual information from the training data.

Point cloud based 3D modeling. Point cloud segmentation is often an integrated step in 3D scene acquisition and modeling. Shao et al. [22] segment individual RGB-D images into regions that correspond to objects, which are then used for modeling with 3D objects from a database. When the segmentation is not ideal, they resort to user input to provide essential guidance. Single RGB-D images have limited views and often contain occlusions. Their method fuses results from multiple RGB-D images when necessary. This provides a plausible solution, although the segmentation step does not benefit from more complete information that could be obtained by fusing multiple RGB-D images into a point cloud. Nan et al. [23] instead take 3D point clouds as input. To address the challenge of point cloud segmentation, they use object level classification to validate and correct segmentation in an iterative manner. However, for challenging cases with cluttered scenes and objects significantly different from models in the database, this greedy approach can produce sub-optimal results. Chen et al. [24] improve the robustness of indoor scene modeling from point clouds by using object-level context information learned from a synthetic scene database. The context information however is only effective when objects involved are correctly segmented. Wang et al. [25] decompose the input point cloud into primitives and achieve scene modeling using graph-matching of primitives. The method requires successful primitive fitting to work effectively, which can be challenging for cluttered scenes with complicated objects. Wang and Xu [26] detect local structures composed of planes, cylinders and primitive surfaces, leading to robust recovery for LiDAR scans. Li et al. [27] present a volumetric patch-based optimization technique for constructing a complete 3D model for an object from a single RGB-D image.

Indoor scenes often contain repeated objects with certain variability. Instead of modeling the entire scene, Kim et al. [28] learn primitive-based models in the training stage, which are then used to recognize repeated objects, allowing variability between parts. Their method requires in the learning stage several scans of interested objects, and does not handle cases where parts cannot be represented as primitives. The work [29] exploits self-similarity by identifying occurrence of similar near-planar patches using clustering. The method is only based on geometric similarity, so cannot identify semantically related objects with different shapes. Our method aims to produce object-level semantic segmentation and labeling, where the same object category may have substantial shape variations.

For real-time reconstruction and modeling, Wang and Kuo [30] reconstruct 3D scenes by using features to optimize camera poses. Yan et al. [31] propose a visual SLAM approach to aligning virtual and real worlds together, which is served for Augmented Reality applications. Zhang et al. [32] detect plane primitives and use them as constraints to improve the obtained point cloud. Xu et al. [33] develop an automatic scanning system that performs real-time RGB-D image fusion, segmentation and modeling. To cope with challenging segmentation, they use a PR2 robot to push the target item to make segmentation easier. Wang et al. [34] produce piecewise planar models (PPM) of complex scenes via multi-level energy minimization. They focus on plane reconstruction and texture reconstruction for large buildings. Our work focuses on semantic segmentation and labeling, addressing a key step between scene construction and scene modeling.

Semantic labeling and understanding of indoor scenes. Most research in semantic labeling works on RGB-D data using machine learning. Such work identifies specific objects from RGB-D images as 3D bounding boxes which requires estimating both objects and their orientations [35], [36], [37] and [1], [7] achieve pixel level labeling of object categories using a recurrent CNN. Such techniques however only work for RGB-D images rather than point clouds as they rely on a large amount of training data with regular topology.

Semantic labeling of 3D point clouds is a much more challenging task. Silberman et al. [1] propose a semantic region segmentation method for RGB-D image just for 4 labels (*ground*, *furniture*, *prop*, *structure*). Koppula et al. [38], [39] develop an approach to semantic labeling of 3D point clouds for indoor scenes. This method treats patch-based segmentation and labeling as two *separate* problems and uses a standard region growing technique for patch segmentation. It learns context information between *object labels*. At runtime, semantic labeling is formulated as an optimization problem, which is solved using mixed integer programming. However, the consistency of patch-based segmentation significantly affects the effectiveness of statistical learning. Moreover, objects are typically associated with patches with large variations, and similar patches can often be part of different object categories. This negatively affects the effectiveness of learning. Also, both training and applying the trained model are expensive, especially for problems with a larger number of patches. They use both color and geometry information, as well as camera related features. Our method assumes the input is in the simplest form, namely point cloud data with geometry only.

To address the limited availability of point cloud data, one approach is to propagate labels learned from ImageNet to 3D point clouds [40]. This avoids training using point cloud data directly, although the performance is still below that of [38]. The work [41] develops a method for semantic labeling of point cloud data without the need of handcrafted features. However, their method requires the input data to contain both the point cloud and the corresponding RGB-D images, so cannot be applied to our problem with only point clouds as input.

Some work considers specific higher-level semantics such as functionality and interaction of objects [42], and checking safety and stability of objects in the scene [43]. For robotic applications, Wu et al. [44] propose a learning based method for task-specific semantic labeling of RGB-D images where the desired level of semantic labels depends on the need of the task.

3 Our Method

Our method takes 3D point clouds as input, with each point containing the position $\mathbf{p}(p_x, p_y, p_z)$ and normal direction $\mathbf{n}(n_x, n_y, n_z)$. Each point in the training set also carries a manually assigned label l and object obj. Given a test point cloud, our aim is to assign a label to each point for semantic labeling, and for instance segmentation, segment the point cloud into semantically meaningful objects. As explained before, we focus on analyzing geometry information and ignore color information.

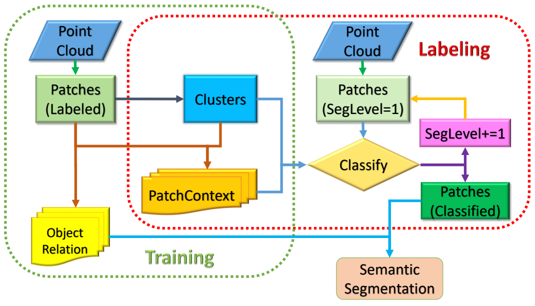


Figure 2. Pipeline of our method.

The pipeline of our method is summarized in Fig. 2. In the training stage, point clouds are first segmented into patches guided by provided object labels to ensure object boundaries are well preserved. Obtained patches are then clustered with each cluster representing patches with similar characteristics. Using clusters as an intermediate representation allows contextual rules to be identified more robustly. We further learn pairwise patch context information from the training dataset. In the test stage, we perform classification based on learned contextual rules and patches generated at different segmentation levels. Multiscale processing controls and selects appropriate segmentation levels for local regions guided by semantic labels and related context. For instance segmentation, this is followed by merging semantic patches according to the learned pairwise cluster-object model to obtain instance level segmentation.

3.1 Patch Generation using Dynamic Region Growing

In this step, we propose a simple *Dynamic Region Growing (DRG)* method to efficiently cluster a point cloud into a set of disjoint patches with smooth geometry. Unlike traditional region growing algorithms (e.g. [29]), our method dynamically updates the patch shape feature during region growing, making it more robust to normal noise and sampling, as demonstrate in Figs. 3 and 8.

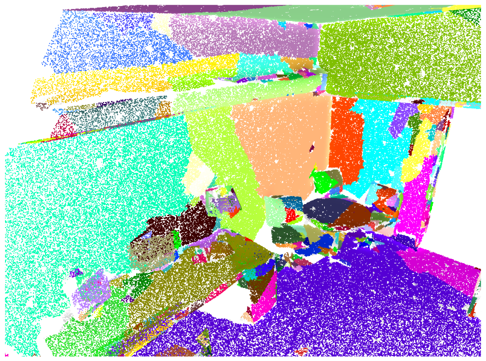
Region growing for the i^{th} patch $\mathbf{P}_i$ starts with a new seed point chosen from unassigned points with the minimum curvature[1]. Denote as $\mathbf{o}_i$ and $\mathbf{n}_i$ the center point and the normal of the growing patch $\mathbf{P}_i$, which are re-estimated every time when the patch size (i.e. the number of points in the patch) $|\mathbf{P}_i|$ reaches $c_1|\widetilde{\mathbf{P}}_i|$. Here, $|\widetilde{\mathbf{P}}_i|$ is the size of the patch when last estimated. c_1 is a constant multiplier set to 1.5 in our work. Considering a point $\mathbf{p}^{(x)} \in \mathbf{P}_i$ and a point $\mathbf{p}^{(y)} \notin \mathbf{P}_i$ closest to it, $\mathbf{p}^{(y)}$ can be added to $\mathbf{P}_i$ if the following conditions are satisfied:

$$\begin{aligned} \|\mathbf{n}^{(y)} \cdot \mathbf{n}_i\| &> k_1 \\ \|(\mathbf{p}^{(y)} - \mathbf{o}_i) \cdot \mathbf{n}_i\| &< k_2 \\ \|(\mathbf{p}^{(y)} - \mathbf{p}^{(x)}) \cdot \mathbf{n}_i\| &< c_2 \end{aligned} \quad (1)$$

where the threshold c_2 is set to $0.5cm$. Here, the first two conditions specify the constraint that the new point should fit with the patch normal plane, and the last condition requires consistency during the growing step.

k_1 and k_2 are key parameters that control the patch segmentation results. Looser constraints of k_1 and k_2 may cause ambiguous under-segmentation and stricter ones can result in over-segmentation. As under-segmentation causes objects to be mixed in patches, existing methods typically choose parameters leaning towards over-segmentation. We take a different, more robust approach. In the training stage, k_1 and k_2 are set to 0.5 and $5cm$. And we use the known point level labels to restrict region growing and only add points with a consistent label to the growing patch. In the test stage, *multiscale processing* is used to automatically select locally appropriate k_1 and k_2 to avoid under-segmentation or over-segmentation (see Sec. 3.4 for more details), using the learned patch-based semantic model for guidance. Although different strategies are used in the training and test stages since the label information is only available during training, both strategies aim to produce segmentation that better respects semantic boundaries. Fig. 3 compares our DRG result with the traditional region growing method [29],

1. This feature is calculated using the PCL library, and defined as the ratio of the minimum eigenvalue to the sum of three eigenvalues when applying PCA to points in the local neighborhoods.


(a) Traditional Region Growing

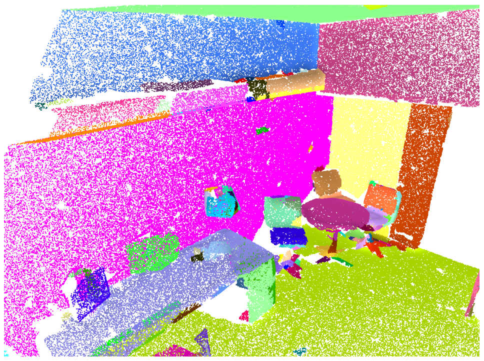

(b) Our Dynamic Region Growing (DRG)

Figure 3. Patch segmentation results of an office scene from S3DIS with $(k_1, k_2) = (0.8, 1cm)$. (a) the result of [29], (b) our DRG result. [29] produces over-segmentation due to normal noise, whereas DRG can tolerate noise to a considerable extent.

which shows our method is more robust to noise and suppresses over-segmentation.

3.2 Patch Clustering

To make probability analysis more robust and efficient, we cluster patches using an unsupervised clustering method, based on their feature similarity.

3.2.1 Feature descriptors

We first compute the Oriented Bounding Box (OBB) for each patch. OBB takes the patch normal as its orientation and projects points onto the fitting plane. Then, the *fitting rectangle* is computed within the fitting plane, defined as the rectangle with the minimum area covering all the projected patch points, for which the optimal solution is efficiently obtained using a Convex Hull Algorithm (with $O(n\,logn)$ time complexity) and Rotating Calipers Algorithm [45] (with $O(n)$ time complexity), where n is the number of points in the patch.

An example of OBBs for patches of a chair is given in Fig. 4 which provides a good approximation to the shape and allows patch features and relations to be calculated efficiently. We then define 9 feature descriptors that can be calculated from a patch, as summarized in Tab. 1. They are easy to compute and cover the location (height), shape, size and point distribution of a given patch; a similar set of features are widely used in existing research. As we will explain in the next subsection, these features are normalized in a consistent way without the need of manually specifying their contributions.

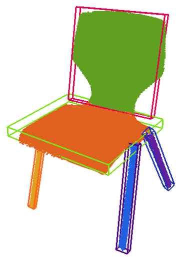


Figure 4. *OBBs.*

F	Description
f_1	Minimum height
f_2	Maximum height
f_3	Curvature of the patch
f_4	Angle between patch plane and horizontal plane
f_5	Area of the concave hull
f_6	Length of the fitting rectangle
f_7	Ratio between length and width of bounding box
f_8	Ratio between area f_5 and the fitting rectangle area
f_9	Point density – ratio between patch size and area f_5

Table 1
Feature descriptors of a patch.

3.2.2 Clustering

To allow each object to contain patches with diverse characteristics and learn rules from examples robustly, we introduce patch clusters as an intermediate representation to bridge between patches and objects. Patch clustering can be efficiently achieved using unsupervised clustering for all the patches in the training set.

For each patch $\mathbf{P}_i$, we have the 9-dimensional feature descriptor $\mathbf{F}_i = (f_1^{(i)}, f_2^{(i)}, ..., f_9^{(i)})$. Instead of clustering directly in this feature space, we notice that there are substantial variation of feature values for the same feature type (the height of an object can be 0 for objects on the floor, but can also be much larger), as well as between feature types (the range of angles are very different from those of the height for example). To address these, we normalize each feature $f_k^{(i)}$ with the standard deviation of the feature σ_k, and perform logarithm to cope with large difference in values. Specifically, we obtain a transformed vector $\mathbf{V}_i = (v_1^{(i)}, v_2^{(i)}, ..., v_9^{(i)})$ as follows:

$$v_k^{(i)} = \ln(\frac{f_k^{(i)}}{\sigma_k} + \varepsilon), \tag{2}$$

where ε is a small offset value which is set to $\frac{1}{e}$ to ensure the validity of $\ln$ function with 0 as input.

We define the feature discrepancy between the patches $\mathbf{P}_x$ and $\mathbf{P}_y$ as the ℓ_1 norm of the difference of feature vectors $\mathbf{V}_x$ and $\mathbf{V}_y$:

$$d(\mathbf{V}_x, \mathbf{V}_y) = |\mathbf{V}_x - \mathbf{V}_y|_1 = \sum_{i=1}^{9} |v_i^{(x)} - v_i^{(y)}|. \tag{3}$$

To perform unsupervised patch clustering, we use the K-means++ algorithm [46] (with $O(KN)$ time complexity where K is the number of clusters and N is the number of patches). Traditional clustering methods require the number of clusters and initial cluster centers to be specified, which can be data dependent and difficult to determine. Unlike these methods, K-means++ just takes a parameter of the maximum cluster distance which can be easily determined by features and has better adaptability for different datasets. The number of clusters K is determined using an iterative process where clusters are iteratively added until the maximum cluster distance within the same cluster exceeds a threshold (typically chosen as 1.7 in our experiments). Alternative spectral clustering method requires high computational and memory costs and fails to produce segmentation results due to memory overflow because typically a large number of patches need to be clustered. Even with efficient approximations such as Nyström approximation [47], the memory requirements remain high.

After clustering, we assign each patch $\mathbf{P}_i$ in the training set to a cluster c_i based on its closest cluster center. Each patch in the training set also contains its object category label l_i. We pre-compute a base confidence $conf_{base}$ for a cluster c to belong to object label l using the statistics from the training set, as well as the mean feature vector for each cluster $\bar{\mathbf{V}}^{(c)}$, which will be useful for the later classification task.

$$conf_{base}(c,l) = \frac{|\{\mathbf{P}_i \mid c_i = c, l_i = l\}|}{|\{\mathbf{P}_i \mid c_i = c\}|}, \tag{4}$$

$$\bar{\mathbf{V}}^{(c)} = mean\{\mathbf{V}_x | c_x = c\}, \tag{5}$$

where $|\cdot|$ is the number of elements in the set, and $mean\{\cdot\}$ is the mean vector of elements in the set.

3.3 Patch Context Analysis

Unlike previous work where contextual information is analyzed at the object level [24], [38] which is sensitive to segmentation correctness, and requires explicit modeling for objects and their parts, we propose a new model that represents patch-based contextual relationships using co-occurrence statistics of clusters and labels, which represent both the part-to-part and object-to-object context information simultaneously.

For every training 3D point cloud, we compute all pairs of adjacent patches. To achieve this, we estimate the distance $d(\mathbf{P}_x, \mathbf{P}_y)$ between two patches $\mathbf{P}_x$ and $\mathbf{P}_y$ as the minimum distance between their oriented bounding boxes $d(OBB_x, OBB_y)$ as they usually provide a good approximation to patches. $d(OBB_x, OBB_y)$ is defined as:

$$d(OBB_x, OBB_y) = \min_{\mathbf{X} \in OBB_x, \mathbf{Y} \in OBB_y} \|\mathbf{X} - \mathbf{Y}\|, \quad (6)$$

where OBB_x and OBB_y represent the volumes of the OBBs, and $\mathbf{X}$ and $\mathbf{Y}$ are arbitrary points within them. For efficient implementation, we check if $OBB_x \bigcap OBB_y \neq \emptyset$, and set $d(OBB_x, OBB_y)$ to 0 if this is the case. Otherwise, $d(OBB_x, OBB_y)$ can be efficiently calculated as

$$\min(\min_{\mathbf{X} \in \mathcal{V}_x, \mathbf{Y} \in \mathcal{S}_y} \|\mathbf{X} - \mathbf{Y}\|, \min_{\mathbf{X} \in \mathcal{S}_x, \mathbf{Y} \in \mathcal{V}_y} \|\mathbf{X} - \mathbf{Y}\|), \quad (7)$$

where $\mathcal{V}$ and $\mathcal{S}$ refer to the vertices and bounding surfaces of the corresponding OBB.

We set the adjacent distance threshold $d_1 = 0.50m$, the adjacent patch pair set $\mathcal{P}$ is defined as:

$$\mathcal{P} = \{(\mathbf{P}_x, \mathbf{P}_y) | d(OBB_x, OBB_y) \leq d_1\}. \quad (8)$$

For any patch pair $(\mathbf{P}_x, \mathbf{P}_y) \in \mathcal{P}$, we further analyze their spatial relationship r_{xy}, which can include one or more of the following possibilities: *adjacent* (patch distance is smaller than 0.5m), *close* (patch distance is smaller than 3cm), *orthogonal*, *parallel* and *co-planar*.

For each pair of patches $(\mathbf{P}_x, \mathbf{P}_y) \in \mathcal{P}$ in the training set, we obtain a patch context quintuple $q = (c_x, c_y, r_{xy}, l_x, l_y)$. We count the number of occurrences for each quintuple, denoted as γ_q. Among all the patch pairs in the training set belonging to the same clusters and with the same relationship as q, we further work out the proportion that has the same labels as in q, denoted as τ_q:

$$\tau_q = \frac{\gamma_q}{\sum_{\tilde{x}} \sum_{\tilde{y}} \gamma_{(c_x, c_y, r_{xy}, l_{\tilde{x}}, l_{\tilde{y}})}} \quad (9)$$

Let us denote the set of patch context quintuples as $\mathcal{Q}$. $q \in \mathcal{Q}$ if the following conditions are satisfied:

$$\gamma_q \geq t_1, \tau_q \geq t_2. \quad (10)$$

The former condition states that the quintuple should appear sufficient number of times to avoid accidentally accepting an erroneous quintuple, where t_1 is the minimum frequency required and the latter says that the regulation strength should be sufficiently high (i.e. the quintuple should state a dominant contextual relationship). We set $t_1 = 2$ and $t_2 = 0.75$ in our experiments.

3.4 Semantic Labeling

In the test stage, given an unknown 3D point cloud, the aim of semantic labeling is to segment it into a patch set $\{\mathbf{P}_i\}$ and assign the correct object label to each patch.

3.4.1 Single Level Semantic Labeling

We first describe single level semantic labeling. We use a region growing algorithm similar to the one in Sec. 3.1 although do not constrain region growing to points with the same label as label information is not available for the test dataset, which produces the patch set $\{\mathbf{P}_i\}$. We will then need to assign an object label l_i to each patch $\mathbf{P}_i$.

We define a score s for label assignment as

$$s(\{l_i\}) = \sum_i conf_p(\mathbf{P}_i, l_i) + \sum_x \sum_y conf_{pc}(\mathbf{P}_x, \mathbf{P}_y, l_x, l_y), \quad (11)$$

where $conf_p$ considers the confidence of assigning label l_i to patch $\mathbf{P}_i$. We use a weighted sum of base confidence $conf_{base}(c, l_i)$ for cluster c to belong to label l_i, where the weight decreases when the patch $\mathbf{P}_i$ is more different from the cluster center:

$$conf_p(\mathbf{P}_i, l_i) = \sum_c w(\mathbf{P}_i, c) conf_{base}(c, l_i), \quad (12)$$

where the weight $w(\mathbf{P}_i, c)$ is defined as

$$w(\mathbf{P}_i, c) = \exp\left\{-\frac{d^2(\mathbf{V}_i, \bar{\mathbf{V}}^{(c)})}{2}\right\}. \quad (13)$$

Here, d is the distance between feature vectors of the patch $\mathbf{P}_j$ and cluster center c, as defined in Eq. 3. Although it is possible to replace the definition of d with Mahalanobis distance, our choice is more consistent with the distance measure used throughout the pipeline.

The binary term $conf_{pc}(\mathbf{P}_x, \mathbf{P}_y, l_x, l_y)$ measures how the patch pair $(\mathbf{P}_x, \mathbf{P}_y)$ satisfies patch context rules, and is defined as

$$conf_{pc}(\mathbf{P}_x, \mathbf{P}_y, l_x, l_y) = \begin{cases} \tau_q, & q \in \mathcal{Q} \\ 0, & \text{otherwise} \end{cases}, \quad (14)$$

where $q = (c_x, c_y, r_{xy}, l_x, l_y)$, c_x and c_y are the nearest clusters to patches $\mathbf{P}_x$ and $\mathbf{P}_y$, to make the problem more manageable. We turn this problem into a label assignment problem by minimizing the following energy:

$$E(\{l_i\}) = \sum_i E_i(l_i) + \sum_x \sum_y E_{x,y}(l_x, l_y), \quad (15)$$

where we define the unary energy term $E_i(l_i) = 1 - conf_p(\mathbf{P}_i, l_i)$, and the binary energy term $E_{x,y}(l_x, l_y) = 1 - conf_{pc}(\mathbf{P}_x, \mathbf{P}_y, l_x, l_y)$. The binary energy term however does not in general satisfy the property:

$$E_{x,y}(l_x, l_x) + E_{x,y}(l_y, l_y) \leq E_{x,y}(l_x, l_y) + E_{x,y}(l_y, l_x) \quad (16)$$

which is required for many multi-label graph-cut algorithms, since for certain pairs, the confidence of them being assigned different labels can be higher than the same label. We solve this optimization problem efficiently using the ICM (iterated conditional mode) method [48] which works for arbitrary energies.

3.4.2 Semantic Labeling with Multiscale Processing

Ideally, patch-based segmentation should produce consistent segmentation for both training and testing to allow learned models to be effectively applied. However, the result of the region growing algorithm is highly controlled by the parameters k_1 and k_2. In the training phase, the label information helps to instruct the segmentation so by setting looser thresholds, the obtained result can largely avoid under- or over-segmentation. However, in the test phase, setting a single set of thresholds that works in all the cases is difficult. For example, using larger k_1 and

k_2 cannot segment a board from the wall as they have similar depth, while using smaller k_1 and k_2 can easily over-segment complex shapes such as a sofa and produce many small pieces. To address this challenge, we propose to use a *multiscale semantic labeling* procedure that uses labeling to guide the segmentation process and chooses locally suitable scales for segmentation and labeling.

Our multiscale semantic labeling takes a set of scale levels, each determined by k_1 and k_2 parameters. $Level_i = (k_1^{(i)}, k_2^{(i)})$, $i = 1, 2, 3, \ldots$.

We initialize the 0th level by treating the entire point cloud data as a single root patch $\mathbf{P}^{(0)}$. We then consider each successive level h, starting from $h = 1$.

We first segment each patch $\mathbf{P}_i^{(h-1)}$ in the $(h-1)$th level into subpatches, using the hth level parameters. This produces a candidate patch set $S^{(h)}$. Not all the patches in $S^{(h)}$ will be accepted. We use semantic labeling to help determine whether such further segmentation is beneficial.

We classify the subpatches in $S^{(h)}$ using the single level semantic labeling described in Sec. 3.4.1. Assume the parent patch for $\mathbf{P}_i^{(h)}$ in the hierarchy is $\mathbf{P}_j^{(h-1)}$, and the object label for $\mathbf{P}_i^{(h)}$ is $l_i^{(h)}$.

We define the score of assigning l_i to a patch $\mathbf{P}_i$ as the sum of the subset of terms in the global score $s(\{l_i\})$ with $\mathbf{P}_i$ and label l_i involved. Note the following definition works for any level:

$$s(\mathbf{P}_i, l_i) = conf_p(\mathbf{P}_i, l_i) + \sum_y conf_{pc}(\mathbf{P}_i, \mathbf{P}_y, l_i, l_y). \quad (17)$$

We further define a normalized score $\tilde{s}(\mathbf{P}_i, l_i)$ using all the possible labels:

$$\tilde{s}(\mathbf{P}_i, l_i) = \frac{s(\mathbf{P}_i, l_i)}{\sum_l s(\mathbf{P}_i, l)}. \quad (18)$$

A subpatch $\mathbf{P}_i^{(h)}$ is accepted if both of the following two conditions are satisfied:

$$l_i^{(h)} \neq l_j^{(h-1)} \quad (19)$$

$$\tilde{s}(\mathbf{P}_i^{(h)}, l_i^{(h)}) > \tilde{s}(\mathbf{P}_j^{(h-1)}, l_j^{(h-1)}). \quad (20)$$

The former states that the subpatch has a different object label from its parent patch. Otherwise the patch should not be further segmented to avoid over-segmentation. The latter says the normalized confidence score for the subpatch should be higher than the parent patch.

The $Level_h$ patches $\{\mathbf{P}_i^{(h)}\}$ are composed of patches from the three sources: 1) accepted subpatches in $S^{(h)}$, 2) patches in $\{\mathbf{P}_i^{(h-1)}\}$ with all the subpatches rejected, and 3) remaining points re-segmented using the parameters in the $(h-1)$th level. The last case is introduced to handle $(h-1)$th level patches with subpatches partly accepted. The result (patches and their assigned labels) of the last level is treated as the result of multiscale semantic labeling.

In our work, we set 5 different segmentation levels. As we will show later, these parameters do not need to be carefully chosen and the performance is significantly better than the results obtained using single level patch segmentation.

3.5 Semantic Instance Segmentation

The aim of semantic segmentation is to break down the input point cloud into disjoint semantic regions, each corresponding to an independent object made up of patches shared with the same label. We utilize the labeling result to help obtain semantic segmentation of the input scene. Specifically, we merge the labeled patches into the object level with their classified labels and patch cluster-object relationships.

Similar to the patch context for semantic labeling, for each pair of patches $(\mathbf{P}_x, \mathbf{P}_y) \in \mathcal{P}$ in the training set, we further obtain a patch cluster-object relationship quintuple $q' = (c_x, c_y, r'_{xy}, l, pred)$. c_x and c_y are the clusters associated with patches $\mathbf{P}_x$ and $\mathbf{P}_y$. r'_{xy} describes the relationship of patches $\mathbf{P}_x$ and $\mathbf{P}_y$, which is similar to patch relationship r_{xy} defined previously, although with the *adjacent* relationship excluded as we only consider close patches for merging.

Denote by o_x and o_y the object IDs patches $\mathbf{P}_x$ and $\mathbf{P}_y$ belong to. $pred$ states whether the two patches belonging to the same object. $pred = 1$ if $o_x = o_y$ and 0 otherwise. l refers to the label of the patch pair, and a patch pair is considered for merging only when they share the same label, as patches belonging to different object categories obviously belong to different object instances. (For experiments where the label information is unavailable, l is ignored and fixed to 0 in this formulation).

We count the number of occurrences for each quintuple, denoted as $\gamma'_{q'}$. Among all the patch pairs in the training set belong to the same clusters, with the same relationship and label as q', we further work out the proportion with the same object relationship (whether or not belonging to the same object) as q', defined as:

$$\tau'_{q'} = \frac{\gamma'_{q'}}{\sum\limits_{pred=0,1} \gamma'_{(c_x, c_y, r'_{xy}, l, pred)}} \quad (21)$$

Define quadruple $t = (c_x, c_y, r'_{xy}, l)$ for describing patch (cluster)-object relationships. Denote by $\mathcal{T}_+$ the set of positive relationships, showing regions that should be merged. $t \in \mathcal{T}_+$ if the following conditions are satisfied:

$$\gamma'_{(c_x, c_y, r'_{xy}, l, 1)} > t_3, \tau'_{(c_x, c_y, r'_{xy}, l, 1)} > t_4. \quad (22)$$

Similarly, denote by $\mathcal{T}_-$ the set of negative patch-object relationships showing regions that should not be merged. $t \in \mathcal{T}_-$ if the following conditions are satisfied:

$$\gamma'_{(c_x, c_y, r'_{xy}, l, 0)} > t_3, \tau'_{(c_x, c_y, r'_{xy}, l, 0)} > t_4. \quad (23)$$

We set $t_3 = 2$ and $t_4 = 0.9$ in our experiments. The semantic segmentation algorithm starts with treating each patch as an object on their own. Let $\mathcal{P}_x$ be the set of patches that belong to o_x, the object that contains patch $\mathbf{P}_x$. Then, initially, $\mathcal{P}_x = \{\mathbf{P}_x\}, \forall x$. We process each patch pair iteratively in the decreasing order of $\tau'_{q'}$. Then, for every pair of patch $\mathbf{P}_x$ and $\mathbf{P}_y$ being considered, we compute the patch-object relationship between any patch $\mathbf{P}_{\tilde{x}} \in \mathcal{P}_x$ and $\mathbf{P}_{\tilde{y}} \in \mathcal{P}_y$, denoted as $\mathcal{T}_{xy}$. Then we merge $\mathcal{P}_x$ and $\mathcal{P}_y$ into the same object, if the following are satisfied:

$$l_x = l_y, \mathcal{T}_{xy} \cap \mathcal{T}_+ \neq \emptyset, \mathcal{T}_{xy} \cap \mathcal{T}_- = \emptyset. \quad (24)$$

In other words, their semantic labels are identical, and there are rules for patches in these sets to be merged, but not rules to stop them from merging. This process repeats until all the patch pairs are processed.

4 Results and Discussions

We now present our results for semantic labeling and segmentation of indoor scenes. We tested our algorithm on a computer with an Intel Xeon CPU E5-2620 2.0GHZ CPU and 32GB memory.

4.1 Running Times

We compare our method with the method for semantic labeling of 3D point clouds [38], [39] (see Tab. 3). This method takes an existing patch segmentation as input. For the Cornell RGB-D dataset [38] where the average patch number is around 50 per scene, their training time is about 2-3 hours. For our benchmark with a moderate number of 486 patches on average, their training time increases dramatically to 7 days. Moreover, their test time for each scene also increases massively to 10 hours. In contrast, our method takes point cloud data as input and our multiscale approach jointly solves patch segmentation and semantic labeling. Even for our benchmark data with an average of 1.5 million points per scene, our training time is only 8 minutes, and the test time is 17 seconds for the single-level case and 45 seconds for multiscale processing, which are *hundreds* of times faster.

To provide a breakdown of running times for our multiscale approach, the running time for patch segmentation is 5 seconds for the first time and 2 seconds for subsequent levels as KD-tree initialization is one-off. The running times for feature extraction and semantic labeling/segmentation are 8 seconds and 2 seconds respectively.

For larger datasets, such as S3DIS, our method still works efficiently, while the method [38] is unable to produce results due to running out of memory.

4.2 Semantic Labeling Results

We use our benchmark based on the ETH database, Cornell RGB-D, SceneNN and S3DIS to evaluate semantic labeling. To demonstrate our method, we use single-level segmentation with two parameters settings. $Level_A$: (k_1, k_2) = $(0.5, 5cm)$ uses looser thresholds so tends to produce fewer, bigger patches. $Level_B$: (k_1, k_2) = $(0.9, 1cm)$ on the other hand uses tighter thresholds and produces a larger number of smaller patches. We also use our multiscale patch segmentation, with the local segmentation level guided by semantic labeling. For multiscale segmentation, we use 5 levels with the following settings for (k_1, k_2): $(0.5, 5cm)$, $(0.6, 4cm)$, $(0.7, 3cm)$, $(0.8, 2cm)$ and $(0.9, 1cm)$. There is no need to carefully choose these parameters due to the robustness of our multiscale segmentation, so we simply choose evenly distributed thresholds. The *same* parameter setting is used for all experiments.

ETH benchmark. ETH point cloud dataset [29] is a high-quality point cloud dataset containing 18 office point cloud scenes constructed from laser point cloud data. As [29] does not provide labels, we manually labeled the point cloud data in two ways to support evaluation: 1) semantic label for each point, useful for evaluating semantic labeling, 2) individual object segmentation, useful for evaluating instance segmentation. Specifically, we use a large number of semantic labels, ***floor, wall, door, board, chair, desk, keyboard, monitor, mouse, cup, CPU, laptop, window, desk cabinet, basket, trash, phone, vase, lamp, eraser, radiator, bookcase, cabinet, pipe, box, headset, bag, table, plants, trunk, sofa, pillow, paper, clothes_rack, outlet, clothes, plastic_bag, slipper*** and ***fan***.

Out of the 18 office scenes, we take 12 scenes for training and the rest 6 scenes for testing. We select 25 common classes to test our labeling performance. All the selected classes are shown in Tab. 2. The remaining classes appear fewer than 5 times across the whole dataset, and therefore are meaningless to test learning capabilities. For this dataset, we use label guided segmentation in the training set, which generates 5493 patches for training as well as 278 clusters and 17361 patch context relationships.

Tab. 2 shows the performance of our semantic labeling. For each class, we report the F-measure (the harmonic mean of precision and recall) to give a concise, balanced performance indicator. Our algorithm performs well on majority of the classes. Fig. 5 compares our semantic labeling results without and with patch-based contextual information. The multiscale approach is used for both results. We can see that the contextual information helps resolve ambiguities, such as the board, the keyboard and the phone, which are incorrectly labeled without using context. It helps improve the performance from 86.98% to 93.81%. Please refer to Tab. 2 with detailed performance comparisons which shows the improvement of patch context.

Fig. 6 shows an example of patch segmentation results. The result of $Level_A$ segments the chair and the wall reasonably well, although certain objects such as the board and the keyboard are under-segmented, since they are near co-planar to the wall or desk with only a small depth difference. The result of $Level_B$ on the other hand segments the board correctly, although objects such as the desk or monitor might be over-segmented, leading to incorrect semantic labeling. With under-segmentation, it is not possible to correct this in the later stages of the pipeline. Although over-segmentation may still allow correct labels to be assigned, while the unreasonable patch boundaries would affects the performance of semantic labeling. Our multiscale patch segmentation produces reasonable patches across the whole input point cloud data. The effectiveness of multiscale processing is also demonstrated in Fig. 7 where the overall accuracy of our method increases with an increasing number of levels. Multiscale segmentation helps improve the overall point accuracy rate from 83.34% to 93.81% for our benchmark based on the ETH dataset. Such performance gain gradually diminishes when the number of levels reaches 5. Similar performance gain is achieved with different datasets.

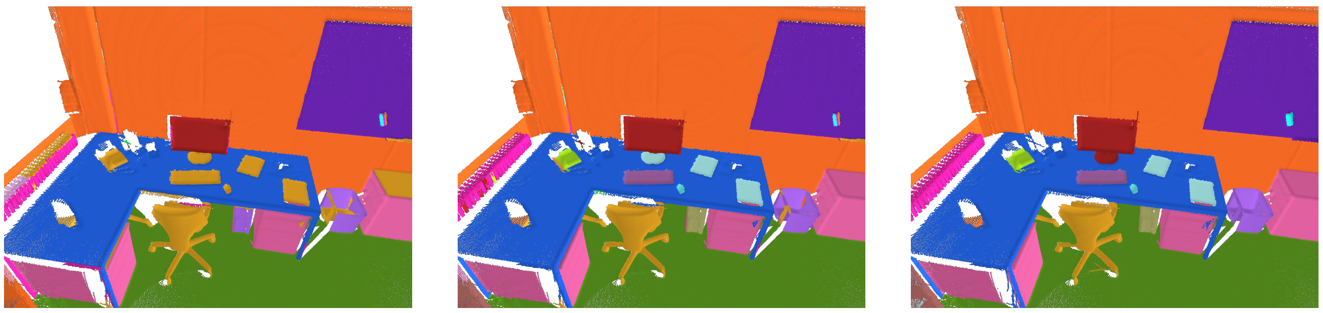
Note that [38] requires patch segmentation as input, to avoid potential loss of performance due to poor patch segmentation, we use *label guided segmentation* as the initial patch segmentation for their method. If a standard region growing based patch segmentation were used, the performance would be lower.

To test the robustness of our patch segmentation method,

F-measure	floor	wall	door	board	chair	desk	key-board	monitor	mouse	cup	CPU	window	desk cabinet
$Level_A$	98.76	88.05	00.00	00.00	77.23	86.13	00.00	82.55	00.00	83.13	00.00	72.59	60.04
$Level_A + PC$	98.75	89.19	99.68	00.69	92.62	86.70	00.38	89.70	00.00	78.08	78.66	82.11	72.27
$Level_B$	97.16	85.16	41.31	68.87	72.08	73.08	00.00	23.80	00.00	00.00	00.00	38.81	71.76
$Level_B + PC$	98.05	92.76	96.91	94.20	86.33	76.61	35.09	84.72	00.00	51.29	37.02	62.18	71.98
Koppula [38]	78.90	80.61	00.00	00.28	68.27	79.64	61.98	86.89	50.03	89.38	36.48	00.00	00.00
$MultiScale$	99.48	96.08	00.00	98.48	71.27	93.76	00.00	82.97	00.00	77.41	00.00	77.40	71.31
$MultiScale + PC$	99.59	97.56	99.87	98.55	92.51	96.52	91.63	87.83	63.93	70.04	53.43	82.51	75.18

F-measure	basket	trash	phone	vase	eraser	radiator	book cabinet	pipe	plant	paper	clothes rack	clothes	Overall
$Level_A$	70.37	79.76	00.00	00.00	00.00	60.02	44.47	48.50	53.52	00.00	22.88	00.00	79.69
$Level_A + PC$	84.88	83.48	36.43	39.02	00.00	80.63	71.09	51.83	56.06	26.13	40.90	41.34	85.12
$Level_B$	53.20	45.97	00.00	00.00	00.00	09.81	28.53	00.00	00.00	00.00	00.00	00.00	78.19
$Level_B + PC$	72.76	71.44	01.37	32.43	11.09	57.65	54.63	04.67	03.04	10.68	09.84	47.29	86.73
Koppula [38]	00.04	00.00	00.00	00.00	00.00	00.00	00.00	00.00	03.40	00.00	00.00	00.00	66.25
$Multiscale$	72.30	80.88	00.00	00.00	48.77	59.95	52.29	45.50	53.58	07.32	16.89	00.00	86.98
$Multiscale + PC$	87.09	84.55	82.83	15.14	70.34	78.16	79.55	49.11	66.12	51.20	62.81	40.09	93.81

Table 2
Semantic labeling performance (F-measure) on our benchmark based on the ETH dataset with 25 classes. $Level_A$, $Level_B$: results with single level segmentation using two parameter settings (looser and tighter thresholds). $Multiscale$: our multiscale segmentation and labeling. +PC means the method uses pairwise patch context information.



(a) $Multiscale$ (b) $Multiscale + PC$ (c) $Groundtruth$

Figure 5. Comparison of our semantic labeling results without (a) and with (b) patch context (as indicated by +PC). The multiscale approach is used in both cases. In this example, patch context helps significantly improve labeling for on-desk objects such as the keyboard, mouse and phone.

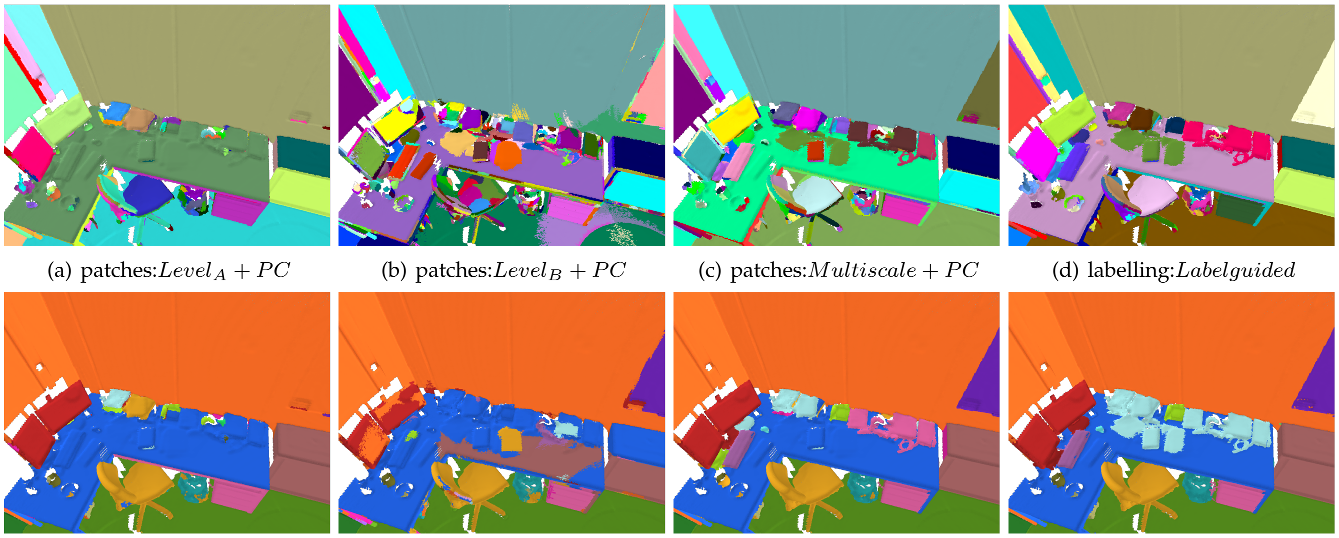
	Our benchmark based on ETH dataset			
	Average Points	Average Patches	Training Time(12 scenes)	Test Time
Koppula [38]	1.5million	486	7days	10h
single level		-	8min	17s
multiscale		-	8min	45s
	Cornell RGB-D dataset			
Office Home	Average Points	Average Patches	Training Time (4 - folds)	Test Time
Koppula [38]	0.5million	46	2.5h	15min
	0.6million	51	3.2h	14min
our method	0.5million	46	4.2s	200ms
	0.6million	51	3.5s	200ms
	SceneNN dataset [49]			
	Average Points	Average Patches	Training Time(62 scenes)	Test Time
Koppula [38]	1.7million	947	Failed	Failed
Our method		-	25min	29s
	Stanford 3D dataset [2]			
	Average Points	Average Patches	Training Time(6 - folds)	Test Time
Koppula [38]	0.8million	317	Failed	Failed
Our method		-	47min	35s

Table 3
Running time comparison for semantic labeling on both ETH dataset, Cornell RGB-D dataset, SceneNN and Stanford 3D dataset.

Fig. 8 shows the patch segmentation results using our method for the original scene, the scene with added normal noise, and the downsampled scene. For each point, the added normal noise has a Gaussian distribution with 0 mean and 30° standard deviation. For downsampling, we keep half of the points from the original scene. Note that such operations are applied to the *entire* dataset, including the training set, to simulate real-world scenarios where the quality of data is poorer. The patch segmentation results as well as the overall accuracies are reported. Our dynamic region growing method is robust, with the drop of accuracy under 1% for the noisy dataset, and under 2% for the downsampled dataset.

Cornell RGB-D dataset. Cornell RGB-D dataset contains 25 office scenes and 34 home scenes. As our aim is object level semantic segmentation and labeling, to test the capability of handling diverse shapes within each category, we generate *object level* labels by grouping relevant part labels. For example, both chair back rest and chair base patches are considered to belong to chairs. We thus have 9 object classes for office scenes: *table*, *floor*, *monitor*, *chair*, *keyboard*, *CPU*, *paper*, *printer* and *book*, and 10 classes for home scenes: *floor*, *table*, *bed*, *shelf*, *pillow*, *sofa*, *chair*, *quilt*, *book* and *laptop*.

We compare our method with [38] using the code provided by the authors. The average performance for both office and home scenes is reported in Tab. 4. We use the same metrics as in [38], where micro P/R (precision/recall) means the percentage of patches correctly labeled, macro results refer to the average precision and recall of every class. The *max_class* is the baseline in [38] corresponding to labeling every patch with the class containing the maximum number of patches. We use the complete pipeline in their paper (color+geometry+context). Note that their method



(a) patches:$Level_A + PC$ (b) patches:$Level_B + PC$ (c) patches:$Multiscale + PC$ (d) labelling:$Labelguided$

(e) labelling:$Level_A + PC$ (f) labelling:$Level_B + PC$ (g) labelling:$Multiscale + PC$ (h) labelling:$Groundtruth$

Figure 6. Comparison of our semantic labeling and patch segmentation results using patch context with different patch segmentation levels (a)(e) single-level $Level_A$, (b)(f) single-level $Level_B$, (c)(g) multiscale processing. Top row: patch segmentation result, bottom row: labeling result.

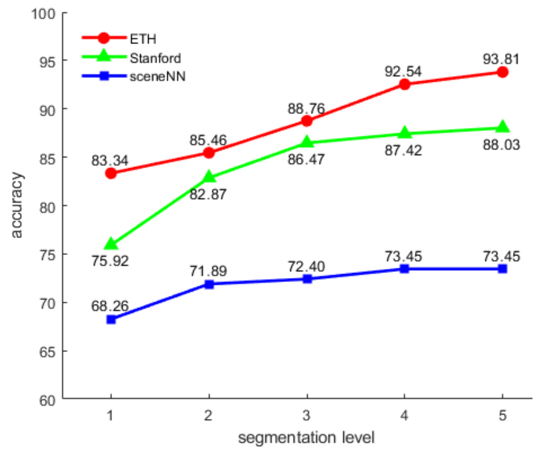


Figure 7. Overall accuracy of labeling results on different benchmarks with different levels of multiscale processing. Higher level means stricter thresholds. ETH dataset has high-quality normals and benefits from high levels of patch segmentation. SceneNN results have less improvement due to incomplete labeling and thus limited context. S3DIS results improve rapidly in low levels due to their RGB-D camera input with relatively high noise.

	Office Scenes		
	micro	macro	
method	P/R	Precision	Recall
max_class	26.91	26.91	11.11
Koppula [38]	77.01	73.08	58.82
Our Method	80.88	74.30	69.02
	Home Scenes		
	micro	macro	
method	P/R	Precision	Recall
max_class	23.01	23.01	10.00
Koppula [38]	47.26	39.57	34.06
Our Method	52.80	50.36	41.05

Table 4
Labeling results on the Cornell RGB-D dataset. For consistency, we report the performance measured at point level, rather than patch level as in [38].

also utilizes camera related features which our method does not exploit. As our method is based on geometry only, we ignore the color information in our pipeline. Nevertheless, our method achieves better performance than [38] (with an overall improvement of 3.87% and 5.54% respectively in terms of micro P/R), demonstrating the effectiveness of our method.

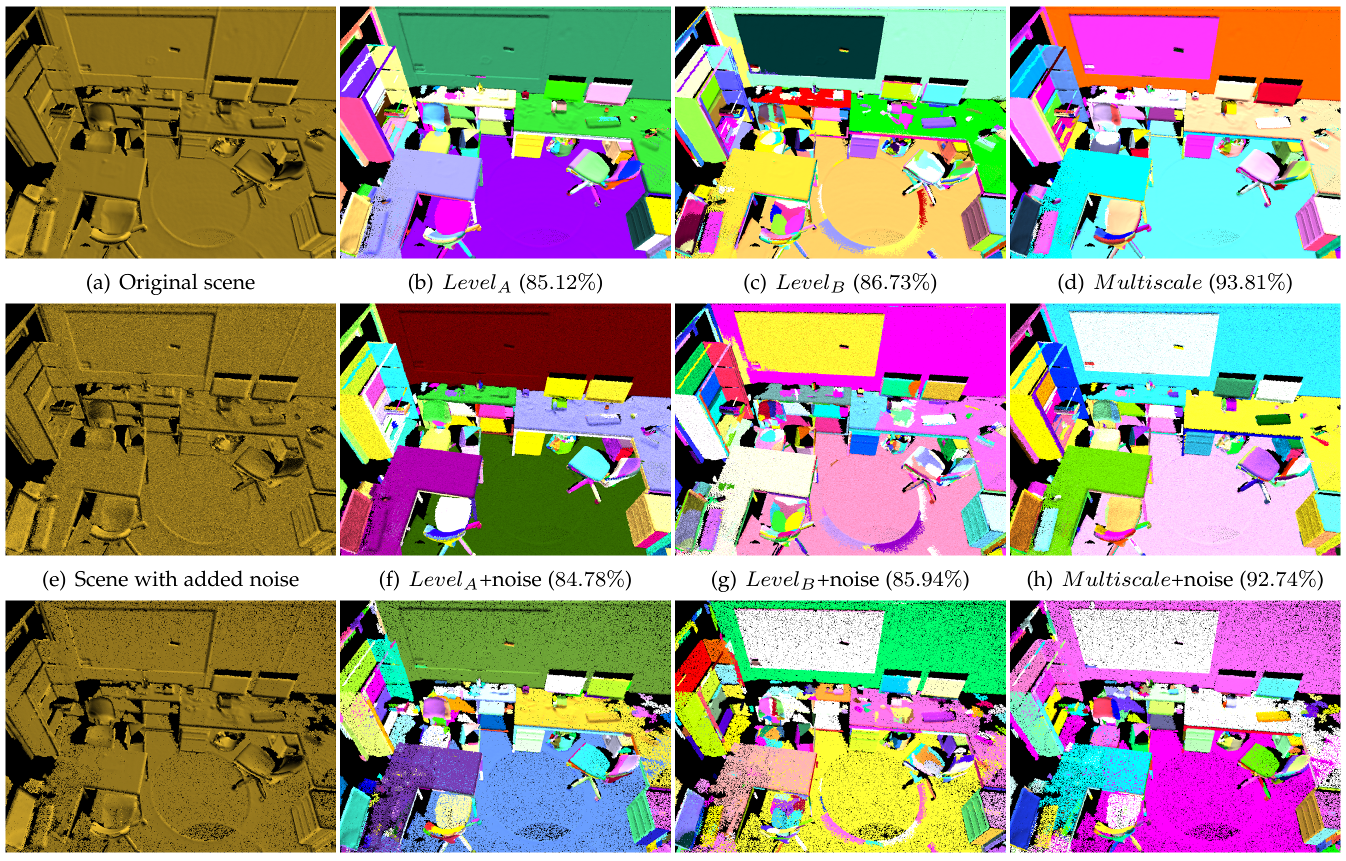
SceneNN dataset. The SceneNN dataset [49] is a labeled point cloud dataset that contains 95 scenes, fused from RGB-D images. We take 62 scenes for training and the remaining 33 for testing, with random partitioning. The SceneNN dataset contains 360 different labels. We merge labels of different styles in the same category into one label. For example, *chair1*, *chair2* and *seat* are treated as *chair*. We then select 15 most common classes with at least 20 instances (*chair*, *monitor*, *desk*, *table*, *wall*, *floor*, *trash bin*, *door*, *shelf*, *pillow*, *keyboard*, *computer*, *bed*, *window* and *fridge*) for testing.

We perform experiments with both our method and [38]. The results are shown in Tab. 6. Our method achieves fairly good performance with an overall accuracy of 73.5%. Patch context analysis contributes to an improvement of 5%. Note that for this dataset, many points are unlabeled. So the multiscale processing has less effect as shown in Fig 7. The method [38] failed in the training stage due to out of memory.

S3DIS dataset. The S3DIS dataset [2] includes 6 areas and 271 rooms captured by Matterport cameras, with ground truth point-level labeling of 13 object categories, including chairs, tables, etc. We compare our method with state-of-the-art deep learning based methods [8], [10]. We use the Intersection over Union (IoU) measure for consistency with existing methods. As shown in Tab. 7, our method achieves significantly better accuracy than [8], [10]. Methods with +RGB mean they use color in addition to 3D information. Detailed per semantic class IoU results are presented in Tab. 5 which shows our method is better than existing method for nearly all the semantic classes. Fig. 9 shows an example. Our method produces results similar to the groundtruth.

4.3 Instance Segmentation Results

We are not aware of existing semantic instance segmentation methods on point cloud data. The method [50] uses a learning based approach to perform indoor scene segmentation for unknown objects. This method takes RGB-D


(a) Original scene (b) $Level_A$ (85.12%) (c) $Level_B$ (86.73%) (d) $Multiscale$ (93.81%)

(e) Scene with added noise (f) $Level_A$+noise (84.78%) (g) $Level_B$+noise (85.94%) (h) $Multiscale$+noise (92.74%)

(i) Scene with downsampling (j) $Level_A$+ds. (83.49%) (k) $Level_B$+ds. (83.57%) (l) $Multiscale$+ds. (91.95%)

Figure 8. Experiments testing the robustness of our DRG patch segmentation. First row: results of the original scenes, second row: results for scenes with added normal noise, third row: results for scenes with downsampling. The overall accuracies are reported in brackets.

IoU	mean IoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board
PointNet [8]	40.0	84.0	87.2	57.9	37.0	19.6	29.3	35.3	51.6	42.4	11.6	26.4	12.5
PointNet+RGB [8]	43.5	81.5	86.7	64.8	29.4	16.3	39.1	48.1	52.5	42.5	05.4	37.6	30.4
Engelmann [10]	43.0	86.5	94.9	58.8	37.7	25.6	28.8	36.7	47.2	46.1	18.7	30.0	16.8
Engelmann+RGB [10]	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	58.1	47.4	06.9	39.0	30.0
Ours	64.6	98.2	98.7	75.2	63.6	37.1	54.3	73.2	72.1	72.7	09.9	56.7	63.1

Table 5
Comparison of IoU between our method and deep learning methods for each semantic class in the S3DIS dataset.

Accuracy	mean IoU	overall accuracy	avg. class accuracy
$MultiScale$	41.2	68.3	51.7
$MultiScale + PC$	45.8	73.5	54.7

Table 6
Accuracy and mean IoU results of our method on SceneNN dataset.

Accuracy	mean IoU	overall accuracy	avg. class accuracy
PointNet [8]	40.0	72.1	52.9
PointNet+RGB [8]	43.5	75.0	55.5
Engelmann [10]	43.0	75.4	55.2
Engelmann+RGB [10]	49.7	81.1	66.4
Ours	64.6	88.0	75.6

Table 7
Comparison of labeling accuracy between our method and deep learning methods on the S3DIS dataset.

images as input. Nevertheless, since it uses a graph-based model, it can be generalized to point cloud data. However, although the method works well on simpler RGB-D datasets with fewer patches, when applied to our benchmark, the SVM appears to predict every graph edge as belonging to different objects. This is because in the training set of pairwise relationships, this case is dominant, and SVM suffers from imbalanced training data. While this problem may be relieved by imbalanced learning techniques [51], such extension is non-trivial and beyond the scope of this paper.

We perform semantic segmentation on ETH dataset using the same training/test separation. We use statistical measures including Rand Index (RI), Global Consistency Error (GCE) and Local Consistency Error (LCE) to quantitatively evaluate different results [19]. Note that some other quantities such as cut discrepancy used for mesh segmentation evaluation do not generalize well to point cloud data as cuts are not clearly defined. For RI, larger is better where for GCE and LCE, smaller is better. As can be seen from Tab. 8, by exploiting the object label information in the training set, more specific and effective patch-object relationships can be learned. This along with contextual information leads to improved object level segmentation results. Our multiscale segmentation only works with label information and achieves the highest performance, significantly better than the baseline region growing method (implemented in

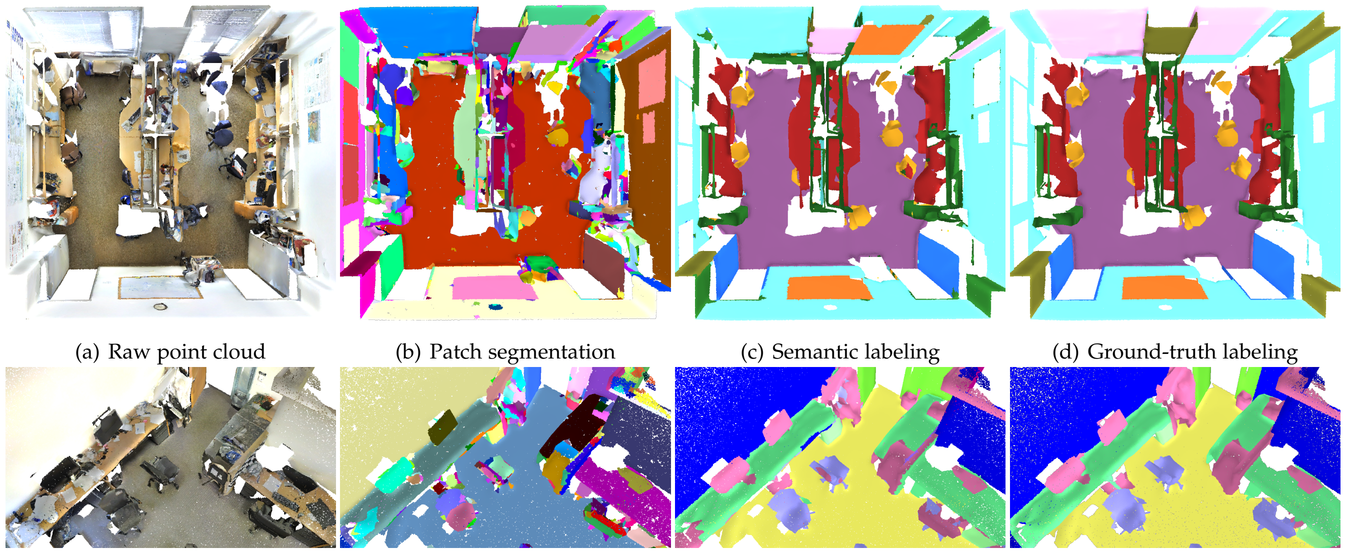

(a) Raw point cloud (b) Patch segmentation (c) Semantic labeling (d) Ground-truth labeling
(e) Raw point cloud (closeup) (f) Patch segmentation (closeup) (g) Semantic labeling (closeup) (h) Ground-truth labeling (closeup)

Figure 9. Semantic labeling results on the S3DIS dataset. (a) raw point cloud, (b) patch segmentation result with multiscale processing, (c) semantic labeling result, (d) groundtruth, (e)-(h): closeup views of results in (a)-(d).

PCL 1.8). Both multiscale segmentation and patch context contribute to this improvement. Fig. 10 shows semantic segmentation results of a scene using different settings, with incorrect segmentation highlighted. By using label information, the learned model can effectively separate the two monitors. The multiscale processing further improves robustness and can effectively segment objects such as chairs.

	RI	GCE	LCE
$RegionGrow[PCL1.8]$	85.03	16.55	10.56
$Level_A$ (without label)	89.21	14.23	09.40
$Level_A$ (with label)	90.97	11.24	04.49
$Level_B$ (without label)	91.20	17.97	15.48
$Level_B$ (with label)	92.75	16.31	08.15
$Multiscale$ (with label)	**97.16**	**07.98**	**03.63**

Table 8
Instance segmentation results. $Level_A$, $Level_B$: instance segmentation results using single-level segmentation with two different settings (looser and tighter thresholds). $Multiscale$: our multiscale instance segmentation result. *without label*: semantic label information is unavailable. *with label*: semantic label information is available and used in the pipeline for semantic instance segmentation.

5 Conclusion and Future Work

In this paper, we propose a novel approach to semantic segmentation and labeling of 3D point clouds. We segment input point clouds into patches, and introduce patch clusters as an intermediate representation between patches and semantic labels. We further develop a novel multiscale patch segmentation and classification approach using learned patch contextual information as guidance. Representing contextual information in the form of patches and their relationships makes the model less sensitive to segmentation inconsistencies. We demonstrate that our method produces improved performance over state-of-the-art methods. Our current method relies on similar patches for learned models to work effectively. Our current patch segmentation method is efficient, but may produce over-segmentation for highly curved surfaces, and fail to produce consistent segmentation for non-rigidly deformed objects. Since our method is patch-based and exploits patch context, it is tolerant with a certain level of occlusions, but the performance can be affected when the scene is highly occluded. We would like to address these in the future, e.g. by generalizing the patch segmentation method. With recent effort, point cloud datasets [2],

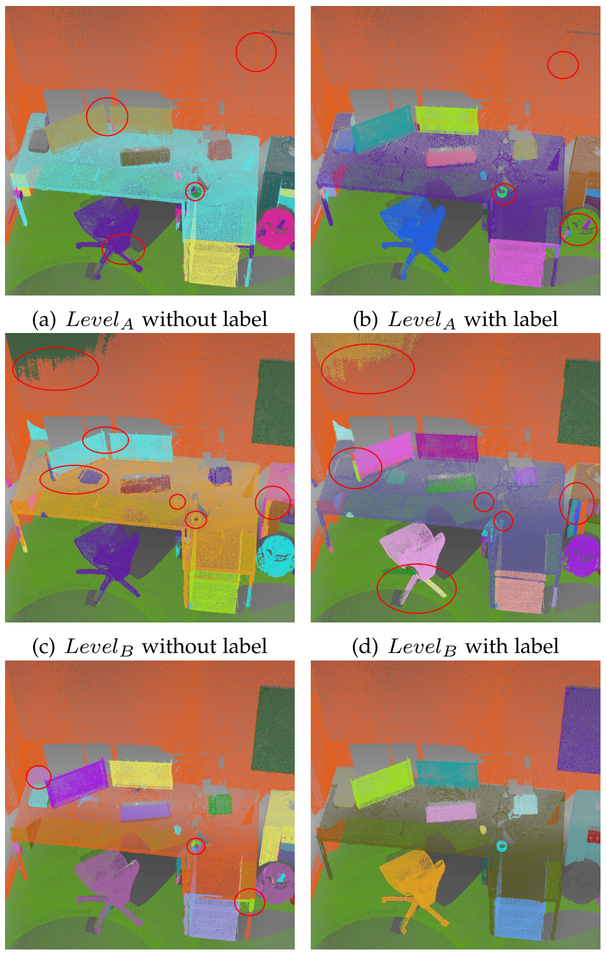

(a) $Level_A$ without label (b) $Level_A$ with label
(c) $Level_B$ without label (d) $Level_B$ with label
(e) $Multiscale$ with label (f) $GroundTruth$

Figure 10. Results of our semantic segmentation, compared with the ground truth segmentation. Incorrect segmentation is highlighted.

[49], [52] are being released. They provide potential for deep learning methods like PointNet [8]. We would like to combine CNN methods with patch shape analysis to further improve our work in the future.

Acknowledgments

This work was supported by the National Key Technology R&D Program (Project Number 2017YFB1002604), the Natural Science Foundation of China (Project Number 61521002, 61761136018) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. We would like to thank the authors and contributors of the ETH dataset [29], the Cornell RGB-D dataset [38], the SceneNN dataset [49] and the S3DIS dataset [2] for making the datasets available. We also thank authors of [38] for providing their code.

References

[1] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision*, 2012, pp. 746–760.

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1534–1543.

[3] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[4] H. Su, F. Wang, E. Yi, and L. J. Guibas, "3D-assisted feature synthesis for novel views of an object," in *International Conference on Computer Vision*, 2015, pp. 2677–2685.

[5] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *International Conference on Computer Vision*, 2015, pp. 945–953.

[6] K. Guo, D. Zou, and X. Chen, "3D mesh labeling via deep convolutional neural networks," *ACM Transactions on Graphics*, vol. 35, no. 1, p. 3, 2015.

[7] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *European Conference on Computer Vision*, 2014, pp. 345–360.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," pp. 77–85, 2017.

[9] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Neural Information Processing Systems*, 2017, pp. 5099–5108.

[10] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3D semantic segmentation of point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 716–724.

[11] R. Mottaghi, X. Chen, X. Liu, N. G. Cho, S. W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[12] L. Yi, L. Guibas, A. Hertzmann, V. G. Kim, H. Su, and E. Yumer, "Learning hierarchical shape segmentation and labeling from online repositories," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 70, 2017.

[13] P.-A. Fayolle and A. Pasko, "Segmentation of discrete point clouds using an extensible set of templates," *The Visual Computer*, vol. 29, no. 5, pp. 449–465, 2013.

[14] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," in *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'*, vol. 36, no. 5, 2006, pp. 248–253.

[15] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *International Conference on Computer Vision Workshops*, 2009, pp. 39–46.

[16] O. van Kaick, N. Fish, Y. Kleiman, S. Asafi, and D. Cohen-Or, "Shape segmentation by approximate convexity analysis," *ACM Transactions on Graphics*, vol. 34, no. 1, p. 4, 2014.

[17] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[18] M. Attene and G. Patanè, "Hierarchical structure recovery of point-sampled surfaces," *Computer Graphics Forum*, vol. 29, no. 6, pp. 1905–1920, 2010.

[19] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," in *ACM Transactions on Graphics*, vol. 28, no. 3, 2009, p. 73.

[20] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D mesh segmentation and labeling," *ACM Transactions on Graphics*, vol. 29, no. 4, p. 102, 2010.

[21] O. Sidi, O. Van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, "Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering," *ACM Transactions on Graphics*, vol. 30, no. 6, p. 126, 2011.

[22] 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*, vol. 31, no. 6, p. 136, 2012.

[23] L. Nan, K. Xie, and A. Sharf, "A search-classify approach for cluttered indoor scene understanding," *ACM Transactions on Graphics*, vol. 31, no. 6, p. 137, 2012.

[24] K. Chen, Y. Lai, Y.-X. Wu, R. 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*, vol. 33, no. 6, 2014.

[25] J. Wang, Q. Xie, Y. Xu, L. Zhou, and N. Ye, "Cluttered indoor scene modeling via functional part-guided graph matching," *Computer Aided Geometric Design*, vol. 43, pp. 82–94, 2016.

[26] J. Wang and K. Xu, "Shape detection from raw LiDAR data with subspace modeling," *IEEE Transactions on Visualization & Computer Graphics*, vol. 23, no. 9, pp. 2137–2150, 2017.

[27] D. Li, T. Shao, H. Wu, and K. Zhou, "Shape completion from a single RGBD image," *IEEE Transactions on Visualization & Computer Graphics*, vol. 23, no. 7, pp. 1809–1822, 2017.

[28] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas, "Acquiring 3D indoor environments with variability and repetition," *ACM Transactions on Graphics*, vol. 31, no. 6, p. 138, 2012.

[29] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola, "Object detection and classification from large-scale cluttered indoor scans," *Computer Graphics Forum*, vol. 33, no. 2, pp. 11–21, 2014.

[30] C. Wang and X. Guo, "Feature-based RGB-D camera pose optimization for real-time 3D reconstruction," *Computational Visual Media*, vol. 3, no. 2, pp. 95–106, 2017.

[31] Z. Yan, M. Ye, and L. Ren, "Dense visual SLAM with probabilistic surfel map," *IEEE Transactions on Visualization & Computer Graphics*, vol. 23, no. 11, pp. 2389–2398, 2017.

[32] Y. Zhang, W. Xu, Y. Tong, and K. Zhou, "Online structure analysis for real-time indoor scene reconstruction," *ACM Transactions on Graphics*, vol. 34, no. 5, p. 159, 2015.

[33] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen, "Autoscanning for coupled scene reconstruction and proactive object analysis," *ACM Transactions on Graphics*, vol. 34, no. 6, p. 177, 2015.

[34] W. Wang, L. Hu, and Z. Hu, "Energy-based multi-view piecewise planar stereo," *Science China Information Sciences*, vol. 60, no. 3, p. 32101, 2017.

[35] S. Song and J. Xiao, "Sliding shapes for 3D object detection in depth images," in *European Conference on Computer Vision*, 2014, pp. 634–651.

[36] ——, "Deep sliding shapes for amodal 3D object detection in RGB-D images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 808–816.

[37] Z. Ren and E. B. Sudderth, "Three-dimensional object detection and layout prediction using clouds of oriented gradients," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1525–1533.

[38] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3D point clouds for indoor scenes," in *Neural Information Processing Systems*, 2011, pp. 244–252.

[39] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena, "Contextually guided semantic labeling and search for three-dimensional point clouds," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 19–34, 2013.

[40] Y. Wang, R. Ji, and S.-F. Chang, "Label propagation from imagenet to 3D point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3135–3142.

[41] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 3050–3057.

[42] R. Hu, O. van Kaick, B. Wu, H. Huang, A. Shamir, and H. Zhang, "Learning how objects function via co-analysis of interactions," *ACM Transactions on Graphics*, vol. 35, no. 4, p. 47, 2016.

[43] B. Zheng, Y. Zhao, J. Yu, K. Ikeuchi, and S.-C. Zhu, "Scene understanding by reasoning stability and safety," *International Journal of Computer Vision*, vol. 112, no. 2, pp. 221–238, 2015.

[44] C. Wu, I. Lenz, and A. Saxena, "Hierarchical semantic labeling for task-relevant RGB-D perception." in *Robotics: Science and systems*, 2014.

[45] G. T. Toussaint, "Solving geometric problems with the rotating calipers," in *Proc. IEEE Melecon*, vol. 83, 1983, p. A10.

[46] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[47] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nyström method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[48] M. Schmidt and K. Alahari, "Generalized fast approximate energy minimization via graph cuts: Alpha-expansion beta-shrink moves," *arXiv:1108.5710*, 2011.

[49] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with annotations," in *International Conference on 3D Vision*, 2016, pp. 92–101.

[50] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *International Conference on Intelligent Robots*, 2012, pp. 4791–4796.

[51] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, no. 9, pp. 1263–1284, 2008.

[52] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, p. 10.

Shi-Min Hu is currently a professor in the department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the PhD degree from Zhejiang University in 1996. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conferences. He is Editor-in-Chief of Computational Visual Media, and on editorial boards of several journals, including IEEE Transactions on Visualization and Computer Graphics, Computer Aided Design and Computer & Graphics. He is a senior member of IEEE and ACM.

Jun-Xiong Cai received his B.S. degree in software engineering from Jilin University, China in 2015. He is currently a Ph.D. candidate in computer science in Tsinghua University. His research interests include computer graphics, geometry processing, deep learning and robot programming.

Yu-Kun Lai received his bachelor's and Ph.D. degrees in computer science from Tsinghua University, China in 2003 and 2008, respectively. He is currently a Reader in the School of Computer Science & Informatics, Cardiff University, UK. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial board of The Visual Computer. He is a member of IEEE.