Received September 3, 2019, accepted November 20, 2019, date of publication December 2, 2019,
date of current version December 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957034

Object-Level Segmentation of Indoor Point Clouds by the Convexity of Adjacent Object Regions

NAN LUO, QUAN WANG, QI WEI, AND CHUAN JING
School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Corresponding author: Quan Wang (qwang@xidian.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61802294, Grant 61702400, and Grant 61972302, in part by the China Postdoctoral Science Foundation under Grant 2018M633472, in part by the Key Technology Research and Development Program of Shaanxi under Grant 2019ZDLGY13-07, and in part by the Fundamental Research Funds for the Central Universities under Grant XJS18034.

ABSTRACT The issue of achieving an appropriate segmentation for indoor point cloud scenes remains difficult. Although available methods continue to improve the benchmark performance, more attentions need to be paid to deal with the drawbacks of inaccurate or incomplete segments in division. To push the research to the next level, this work proposes an learning-free algorithm for the segmentation of indoor point clouds which consists of two stages. The first stage extracts edges of RGBD point clouds and applies them in the voxel clustering process to avoid generating supervoxels which are situated across object boundaries. After this pre-segmentation, a two-phase merging procedure is presented in the second part. By conducting region growing on optimized supervoxels, a set of local regions is obtained. Then we propose to define the convexity-concavity of adjacent regions based on the observations of object structures and merge the convexly connected regions to achieve object-level segmentation. This algorithm is straightforward to implement and requires no training data. Experimental results show that it produces supervoxels with plausible boundaries and arrives at better object-level segmentation.

INDEX TERMS Object segmentation, supervoxels, indoor point clouds, convexity-concavity, merging of adjacent regions.

I. INTRODUCTION

Applications based on 3D point cloud have pervaded in various fields with the widespread use of 3D scanners and rapid growth of open-source project on point cloud data. As a crucial fundamental research topic in point cloud processing, segmentation divides point cloud into a series of local regions by investigating implicit information contained in the original data, making that each segmented region corresponds to an actual object or entity with specific function or semantic meaning [1], [2]. The segmentation results can be widely used in applications like object recognition [3], [4], scene understanding [5], [6], and 3D reconstruction [7], [8], etc.

Despite decades of research, dividing scenes into objects remains one of the most challenging topics in computer vision community, and two main issues are still unsettled. Firstly, due to the limitations of scanning devices, captured point clouds inevitably contain noises or outliers, which impacts the segmentation accuracy since most methods operate directly on the original data. Besides, large data amount of point cloud itself lowers the processing efficiency. Secondly, available methods usually just employ normal, curvature, or colour components of input data in segmentation, and this insufficiency of features leads to under- or over-segmentation of point clouds. Hence object segmentation with precise boundaries cannot be guaranteed.

The quality and speed of segmentation are vitally important to point cloud based studies, such as object recognition, scene analysis & reconstruction, etc. If integrated entities with smooth borders cannot be labeled correctly from the host point clouds, the subsequent algorithms would reconstruct incomplete rough object models, or cause failure in object recognition and scene analysis. In view of this, in this paper we investigate the structure property of indoor objects and propose a convexity guided segmentation algorithm for indoor point cloud which includes two stages: over-segmenting and merging. The over-segmenting stage detects and utilizes edges in point clouds to restrain the border of

The associate editor coordinating the review of this manuscript and approving it for publication was Shuhan Shen.

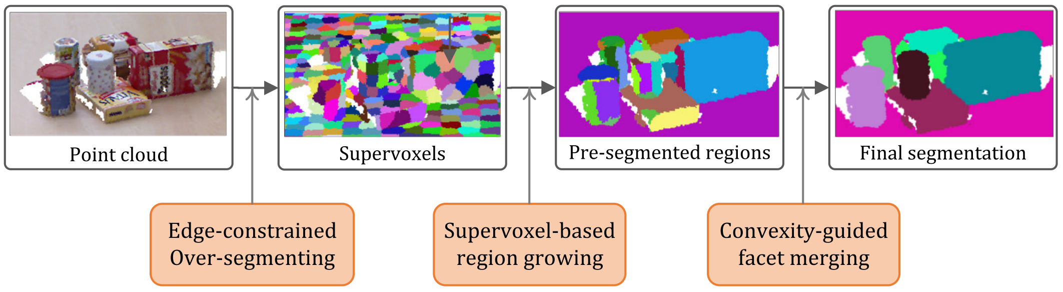

Point cloud
Supervoxels
Pre-segmented regions
Final segmentation
Edge-constrained Over-segmenting
Supervoxel-based region growing
Convexity-guided facet merging

FIGURE 1. Framework of proposed algorithm. The input point cloud is firstly over-segmented into a series of sparse supervoxels by edge-constrained clustering method. Then the region growing principle is applied on generated supervoxels to obtain local facets of objects. After that, we propose the convexity definition of adjacent facets and employ convexity-guided merging algorithm to achieve object-level segmentation.

clustered supervoxels [9], [10]. The second stage first pre-segments a set of local parts in region-growing principle, and then merges the adjacent parts with respect to their convexity-concavity to achieve perfect segments for the objects in a scene. The whole framework is illustrated in Fig.1.

Sec.2 summarizes the current research progress of point cloud segmentation. The details of two stages of proposed algorithm are described in Sec.3, with Sec.4 evaluates it experimentally and Sec.5 draws a conclusion.

II. RELATED WORK

Over the last decade, point cloud segmentation has gained sufficient research focus and achieved great improvements. A number of segmentation methods have sprung up which extend the traditional image segmentation techniques to divide point clouds on the basis of local surface features (i.e. coordinates, normals, or curvatures), e.g. region growing [11], clustering [12], model-fitting based method [13], etc. These ways are simple in principle and easy to implement, however, the drawbacks in robustness often lead to under- or over-segmentation in the final results. Graph-cut method transforms the point cloud to graph structure and then cut the graph via defining a minimum function, for instance, advanced Min-cut [14] and Grab-cut [15]. Besides, many researchers utilize probabilistic model, i.e. Markov Model [16] or Conditional Random Field [17], to solve the graph-cut problem for complicated point cloud scenes at the expense of efficiency. To promote the effectiveness, Stein et al. [18] firstly extract over-segmented supervoxels by voxel clustering, and then perform object partitioning using local convexity in region growing style. An octree-based segmentation [2] partitions voxelized point cloud into a group of incomplete regions and then add unassigned voxels to the regions, which speeds up the segmentation significantly.

Machine learning based segmentation, a newly developed category, extracts and fuses low-level features of point cloud, and then push them into the trained classifier for segmentation, such as Support Vector Machine [19], Random Forest [20], Deep Learning network [21], [22], etc. Among them, the deep neural network represented by CNN and RNN shows great potential in many fields because of its powerful ability of model learning and feature expression. In general, deep neural network requires regularized inputs, whereas point clouds are often unorganized and large in data volume, which brings challenges to these methods. Before 2017, most work operate on multi-view data [23] or volumetric point cloud [24]. Point-based approach draws great attention in 2017. PointNet [25] is the first end-to-end deep neural network operating directly on unordered point clouds. It introduces max-pooling to deal with the ordering issue of input points, and then merges the local and global features for pointwise classification. Kd-network [26] employs $k-d$ tree to tackle the problem of unorganized structure of point cloud. Later, PointNet++ [27], the layered version of PointNet, is proposed to manipulate the point cloud by layered learning to improve the segmentation. Deep learning based approach [28], [29] achieves good result in point cloud segmentation, however, new ideas are expected to promote the efficiency due to the large volume and redundancy of point cloud data.

Besides considering low-level data characteristics, many approaches enhance the segmentation with the aid of object structure or other semantic information, for instance, the local appearance [30] or the relationship between objects [31]. Image assisted method is usually applied for labeling of RGBD scenes [32], and similar interactive scene annotation system [33] has also been developed. Except that, other information, like density [34] or thermal of captured points [35], are used as assistance for labeling. Point cloud segmentation aims to realize object-level labeling or recognition, which is the initial preliminary understanding of scenes.

Among all these approaches, region growing is widely used in point cloud segmentation because of its simplicity. To improve the robustness and accuracy of segmentation, this work runs region growing on the basis of supervoxels to eliminate the influence of data noises, and then introduces the prior-knowledge of object structure and performs further merging on pre-segmented regions by the convexity-concavity of adjacent regions to realize accurate object-level segmentation.

III. SEGMENTATION ALGORITHM

Proposed work adopts a two-stage framework "over-segmenting and merging" for object segmentation of indoor

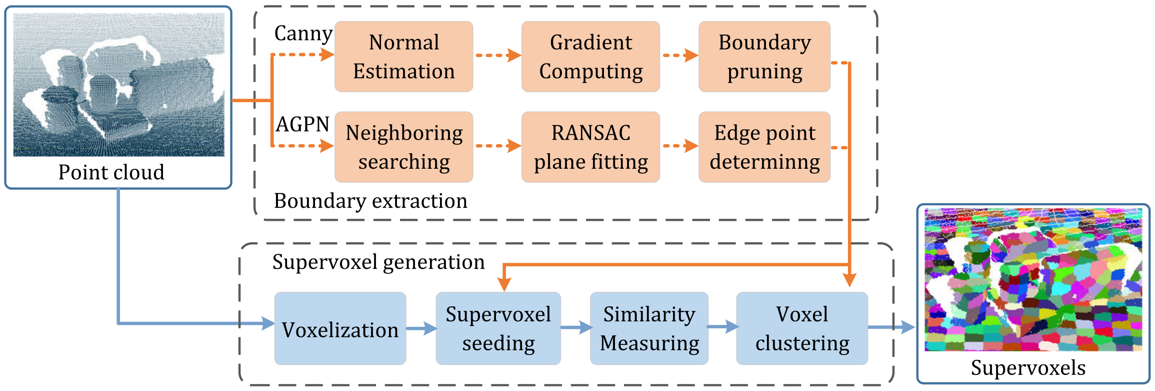


FIGURE 2. Pipeline of boundary constrained supervoxel over-segmentation, consisting of two stages: boundary extraction (upper dotted box) and supervoxel generation (lower dotted box). For RGBD point cloud, 'Canny' method extracts and prunes the object boundaries from the gradient of point cloud normal. For unorganized point cloud, AGPN analyzes the geometric properties of each point's neighborhood and determines the edges by the angular gap. The extracted boundaries are then utilized in the seeding and clustering steps to constrain the supervoxel generation.

point clouds. The first stage detects edges in a point cloud and employs the edges to constrain the supervoxel clustering process so that the generated supervoxels do not step over the object boundaries. The original point cloud is then concisely represented by the sparse supervoxels, which weakens the influence of data noises and simplifies the segmentation. After that, the second stage merges the over-segmented patches in region growing style with respect to the similarity between adjacent supervoxels, and then defines the convexity-concavity between merged regions to conduct further fusion for object-level segmentation.

A. EDGE RESTRAINED SUPERVOXEL OVER-SEGMENTATION

Papon et al. [9] proposed the voxel cloud connectivity segmentation (VCCS) which voxelizes input points with the octree structure and then clusters the voxels into a group of irregular voxel sets, known as supervoxels. Voxelization can effectively reduce the computation time and the inference of noises. However, the supervoxels grown out of this approach often step over the boundaries of objects, which disaccords with the connotation of supervoxel. Proposed procedure solves this problem by extracting the object boundaries reside in the point cloud and applying them in supervoxel seeding and clustering processes, making the generated supervoxels strictly attached to object borders. The pipeline consisting of two main parts is shown in Fig.2.

1) BOUNDARY EXTRACTION

Boundaries correspond to the parts with significant variations of surface normals in point cloud, mainly lying in the intersections of different objects or adjacent facets of one object.

RGBD point cloud has the similar organized structure as image matrix, hence the variant of 'Canny' edge detecting algorithm based on depth image [36] is employed to determine the boundaries of entities. Different from detecting edges of gray images, 'Canny' boundary detection takes the normals of organized point cloud as input and chooses the

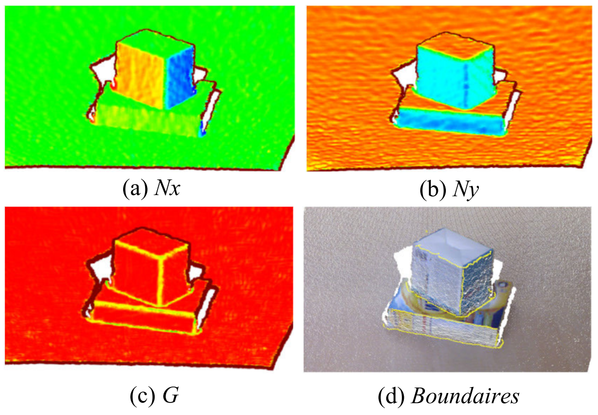


FIGURE 3. Illustration of boundary extraction of RGBD point cloud. (a)-(b) the normal components of point cloud in x and y directions. (c) the gradient of point cloud. (d) extracted boundires.

areas with dramatic normal changes as boundaries. This algorithm extracts the components of normals in x and y directions, N_x and N_y (as Fig.3(a) and Fig.3(b) show), which are later smoothed with Gaussian filter to eliminate the noises. Then it computes the corresponding first-order gradients to get the gradient of point cloud normals G (Fig.3(c)) for detecting the high response areas. At last, the non-maximum depressing process is applied on G to determine the final object boundaries (Fig.3(d)).

For unorganized point clouds, an AGPN (Analysis of Geometric Properties of Neighborhoods, [37]) procedure could be followed to detect the 3D edges by analyzing geometric properties of each point's neighborhood. For each unlabeled point p_o, AGPN firstly searches the neighboring point set P based on the Euclidean distance and fits a local plane on set P by RANSAC algorithm. Then P is divided into two parts, inliers and outliers, by the fitted plane, and point p_o is labeled as edge or non-edge according to the angular gap between itself and the inlier points. If p_o is an interior point (Fig.4(a) and (b)), the distribution of the angles between vectors $\overrightarrow{p_op_i}$ is consecutive, and there will be no substantial angular gap, then

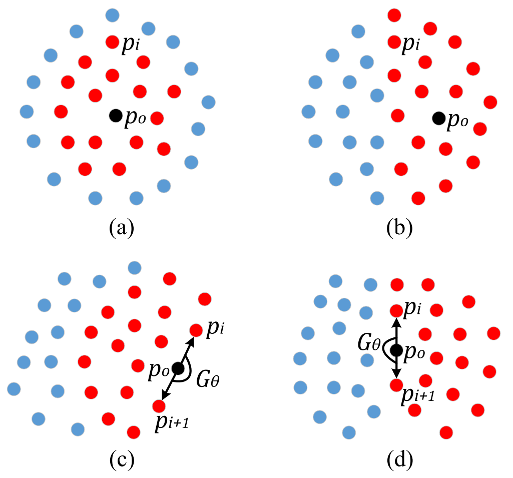


FIGURE 4. Distribution of the neighbors of an unlabeled point on a surface. (a)-(b) the neighborhood of an interior point, (c)-(d) the neighborhood of an edge point. p_o the current point, p_i the neighboring point, G_θ the angular gap. Inliers and outliers are respectively filled in red and blue.

p_o is labeled as non-edge. If p_o is an edge point, there will be a substantial angular gap G_θ (Fig.4(c) and (d)) between vectors $\overrightarrow{p_o p_i}$. Otherwise, point p_o is defined as noise or isolated point. After all the points are labeled, the boundaries are extracted.

2) SUPERVOXEL GENERATION

Supervoxels are the sparse homogenous representation of a point cloud. It brings great convenience to point cloud segmentation by expressing dense points with sparse local patches. The supervoxel generating procedure is stated as follows.

1) Voxelization. Partition point cloud space with predefined voxel resolution R_{voxel} and create voxel cloud with the centroid of each voxel. Then establish the adjacency graph considering 26-neighbours in voxel space. Voxel is considered as the 3D version of image pixel, and the parameter R_{voxel} specifies the resolution (i.e. voxel size) in which the cloud is voxelized. Voxelization is considered as the gridding of the original point cloud which can filter the data noise in the meantime. Large R_{voxel} means sparse voxels with less local features of point cloud, and too small R_{voxel} makes the voxelization lack of robustness to data noise. In practice, R_{voxel} is mainly decided by the density of point cloud, i.e. large for sparse point clouds and small for dense ones. By quantizing the continuous point cloud space to an ordered discrete voxel cloud space, we can establish 3D neighboring relations between voxels and then construct supervoxels.
2) Supervoxel seeding. Define the distance between supervoxels with a resolution R_{patch}, which is larger than R_{voxel} and used to control the size of generated supervoxels. Usually, R_{patch} is preferred not to oversize the smallest object in the scene to avoid possible under-segmentation. Then divide the voxel cloud space by R_{patch} and extract the closest voxel to the centroid of each occupied cell as the candidate seed, followed by a pruning process to determine the qualified seeds to initialize supervoxels. There are two principles for pruning. Firstly, isolate voxels should not be selected as seeds. Secondly, edge voxel or the voxel that has edge neighbors cannot be chosen as seeds, since it can easily lead to the circumstance that supervoxels stepping over the object borders.
3) Similarity measuring. Knowing that the visual color information does not always relate to object boundaries, in this work we only consider the spatial distance and geometric feature of voxels for similarity measuring. As Eq.(1) defines, D_s denotes the spatial distance between two voxels, which is normalized in the formula by R_{patch}, the controlling factor of supervoxel size. D_n is the sine value of two voxel normals. λ_s and λ_n are the weighting factors of two parts. For indoor point clouds, λ_s is empirically set to 0.2.

$$D = \sqrt{\lambda_s \frac{{D_s}^2}{3R_{patch}^2} + \lambda_n {D_n}^2}, \quad \lambda_s + \lambda_n = 1 \tag{1}$$

4) Voxel clustering. The supervoxel generating procedure is similar to that of VCCS algorithm [9]: beginning at each seed voxel, we flow outward to adjacent voxels and measure the similarity distance from each neighboring voxel to the supervoxel center by Eq.(1). If the distance is the smallest this voxel has seen, its label is set, and we add its neighbors which are not edge voxels to the search queue for this supervoxel. Then proceed to the next supervoxel, so that each supervoxel is considered at the same level. This process is iteratively operated until all voxels are checked.

By combining extracted boundary information with the voxel clustering process, the supervoxels could stop growing at the edges of point cloud, avoiding that the generated patches stepping over the object boundaries. Fig.5 gives examples of generated supervoxels on four point clouds from Object Segmentation Database (OSD-v0.2 [38]) with local details enlarged. In this experiment, R_{voxel} and R_{patch} are respectively set to $0.007m$ and $0.06m$. Compared with VCCS [9] (third row in Fig.5), proposed supervoxel generating procedure (bottom row in Fig.5) obtains supervoxels with more plausible borders which are consistent with object boundaries (indicated by yellow dashed lines). On the price of 18% more computations on boundary detecting (Fig.6(a)), it achieves comparable number of patches (Fig.6(b)) to provide better pre-segmentation for subsequent region fusion.

B. CONVEXITY GUIDED MERGING OF ADJACENT REGIONS

Point cloud segmentation aims at dividing a point cloud scene into a group of regions with legible boundaries, and each region can correspond to an actual object in the scene. Above edge-assisted over-segmentation method deals with

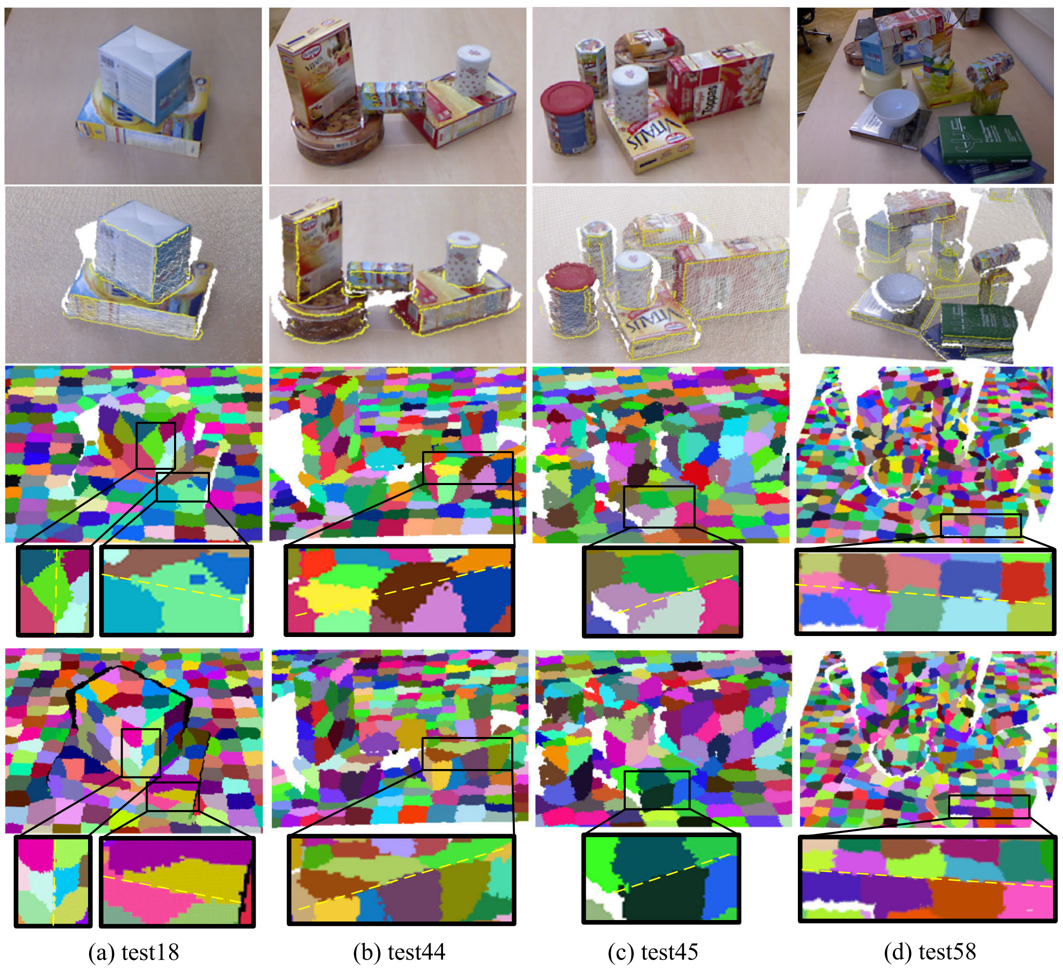


FIGURE 5. Comparisons of generated supervoxels on four point clouds. From top to bottom: original scene, extracted edges, VCCS supervoxels, supervoxels from proposed algorithm. The local details are marked and enlarged, with yellow dashed lines indicate object boundaries. Compared with VCCS (third row), proposed procedure (bottom row) obtains supervoxels with more plausible borders which are consistent with object boundaries. Parameters: $R_{voxel} = 0.007m$, $R_{patch} = 0.06m$.

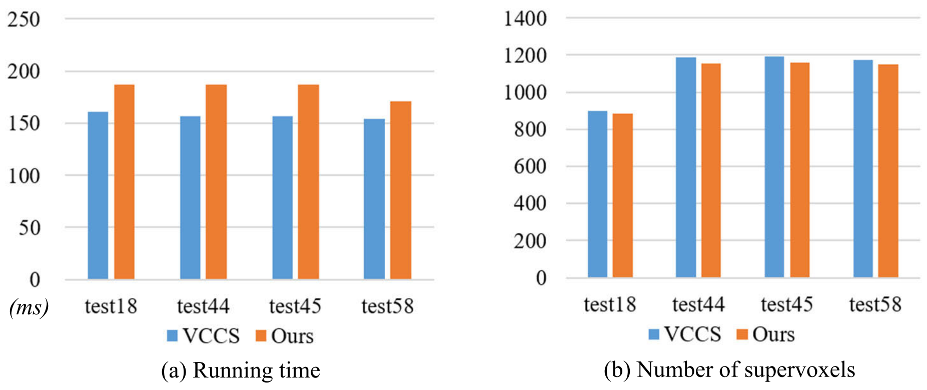


FIGURE 6. Comparisons of running time and generated supervoxel number on the four point clouds. (a)running time, (b)the number of supervoxels. Proposed supervoxel generating procedure consumes 18% extra computations on boundary detecting comparing to VCCS, and achieves comparable number of patches for subsequent region fusion.

the drawback of boundary crossing in local patches, on the basis of which the convexity-guided merging algorithm is presented in this section. The algorithm first carries out region growing on over-segmented supervoxels to gain a few candidate segments, each of which usually stands for one facet of an entity. Then defines the convexity-concavity between

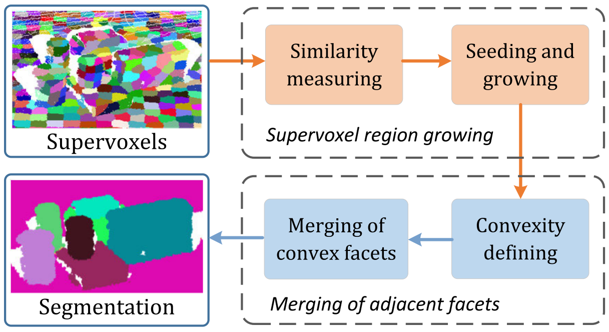


FIGURE 7. The pipeline of convexity-guided merging of adjacent regions. The first phase grows local facets from supervoxels, then the second phase merges them by convexity of adjacent facets to achieve object-level segmentation.

facets and merges the convexly adjoined ones to achieve object segmentation, as Fig.7 depicts.

1) REGION GROWING ON SUPERVOXELS

Establish the 128-dimensional feature space (defined in Eq.(2)) considering the normal and the PFH (Point Feature Histogram [39]) of supervoxels. Here the variation of normals implies the surface smoothness and that of PFHs expresses the geometric characteristic. Then the similarity between two supervoxels is calculated by Eq.(3), where S_n signifies the cosine value of the angle between the normal vectors of two supervoxels, S_{pfh} is the cosine value of the two corresponding PFH vectors, σ_n and σ_{pfh} are the weights. In region growing, we empirically set σ_n to 0.8.

$$F_s = [n_x, n_y, n_z, PFH_{1\ldots125}] \tag{2}$$

$$S = \sqrt{\sigma_n {S_n}^2 + \sigma_{pfh} {S_{pfh}}^2}, \quad \sigma_n + \sigma_{pfh} = 1 \tag{3}$$

Afterwards, choose the supervoxel with minimum residual as the initial seed, then measure the similarity of the seed to its adjacent supervoxels by Eq.(3) and add the qualified ones to the current segments. Meanwhile, push the neighbours that satisfy the residual requirement, i.e. less than a threshold, into the seed queue for region growing. Continue this procedure until the queue is empty, namely, the current segment completes growing. Then select a new seed and start the next round. Repeat above steps until all the supervoxels are assigned. At this point, we get the candidate segments.

2) MERGING OF ADJACENT FACETS

The grown segment generally corresponds to an individual facet rather than a whole object in the scene. To pursue object-level segmentation, the different facets of one object need to be further merged. Observations reveal that the adjacent facets of one indoor object are convexly connected in general, like a box, a can, or a book, while the contact faces of different things present a concave joining relationship, such as a box and its supporting table, a computer and the ground, etc. Based on this prior knowledge, object-level segmentation can be realized via fusing the convexly connected adjacent facets, while with concave ones defining the object borders.

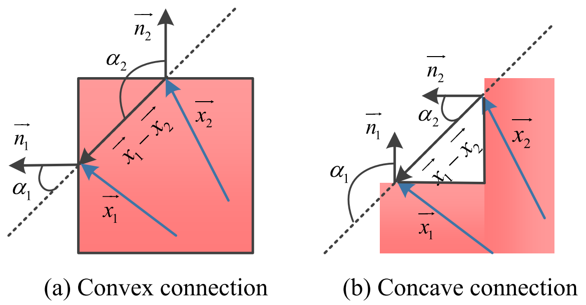


FIGURE 8. Definition of convexity-concavity between two supervoxels.

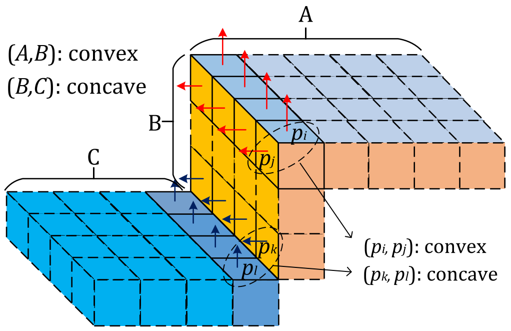


FIGURE 9. Definition of convexity-concavity of two adjacent facets. It is determined by the convexity-concavity of the border supervoxel pairs.

(1) Definition of convexity. The concavity-convexity of two adjacent facets is determined by the concavity-convexity of their border supervoxel pairs.

Fig.8 illustrates the convexity definition [18] between two contiguous supervoxels (given by Eq.(4)). $(\vec{x_1}, \vec{x_2})$ and $(\vec{n_1}, \vec{n_2})$ are respectively the centroid vectors and the normals of two supervoxels p_1 and p_2. (α_1, α_2) are the angles between each normal and the vector $(\vec{x_1} - \vec{x_2})$, whose relationship decides the convexity of supervoxels: if $\alpha_1 < \alpha_2 \iff (\vec{n_1} - \vec{n_2}) \cdot (\vec{x_1} - \vec{x_2}) > 0$, then two adjacent supervoxels are convexly connected. In practice, due to the data noises, two adjacent supervoxels from the same facet may be misjudged as concave relationship just because their similar normals show certain concavity. To avoid that, a threshold β_{th} is introduced to enhance the robustness: supervoxels will have convex relationship if the angle between their two normals is smaller than β_{th}.

$$c(p_i, p_j) = \begin{cases} \text{convex}, & (\alpha_1 < \alpha_2) \vee (\measuredangle(\vec{n_1}, \vec{n_2}) < \beta_{th}) \\ \text{concave}, & \textit{otherwise} \end{cases} \tag{4}$$

The convexity of two neighbouring facets (as illustrated in Fig.9) is given by Eq.(5), in which (p_i, p_j) denotes one element in the supervoxel-pair set $\mathbb{C}_{svp}(A, B)$ at the border of facets A and B. The connection of A and B is convex when more than half of the elements in $\mathbb{C}_{svp}(A, B)$ are judged as convex, otherwise it is concave. Since robustness has been taken into consideration in the convexity determination of border supervoxel-pair, presented convexity definition of neighbouring facets is robust to a certain degree.

$$c(A,B) = \begin{cases} \text{convex}, & \dfrac{|c(p_i,p_j) = \text{convex}|}{|\mathbb{C}_{svp}(A,B)|} \geq 0.5 \\ \text{concave}, & \textit{otherwise} \end{cases} \quad (5)$$

(2) Convexly merging. Firstly, assign label to each segmented facet, that is, assign supervoxels in the same facet with the same label while give different labels to the supervoxels from different facets. Secondly, find the adjacent neighbours for each facet and determine the supervoxel-pair set related to each neighbour. For each supervoxel p_j in one facet R_i, gather its adjacent supervoxels and compare their labels with that of the current supervoxel. If the labels differ, make this adjacent supervoxel p_l and the current supervoxel p_j a pair (p_j, p_l), and add it to the set $\mathbb{C}_{svp}(R_i, R_l)$, assuming R_l is the neighbouring facet corresponding to the adjacent supervoxel p_l. Afterwards, decide the convexity of the current facet to its neighbours by Eq.(5) and merge the convexly connected ones. This is realized by simply operating the following procedure: if facet R_i and its neighbouring facet R_l have convex relation, then reassign R_l the label of R_i. Through this way, the two facets are merged. Continue this process until all regions are checked, and then update the labels to gain the final merging result. Readers can refer to 'Algorithm-1' for more details.

The above convexity-based merging algorithm originating from the observations of object structures optimizes the supervoxel-based region growing results. As shown in Fig.10, region growing pre-segments the point cloud into a group of local facets (middle row) from the clustered supervoxels (top row), then the merging phase fuses the adjacent facets to reach object-level segmentation (bottom row). In summary, proposed algorithm deals with the issue of point cloud over-segmentation, so that each segment semantically represents one whole object in the indoor environments. More detailed comparison will be discussed in next section.

Algorithm 1 Convexly Merging of Adjacent Regions

```
Input: Grown regions $\mathbb{R}\{R_1, R_2, \ldots, R_m\}$,
       all supervoxels $\mathbb{P}\{p_1, p_2, \ldots, p_n\}$
Output: Merged segments $\mathbb{S}\{S_1, S_2, \ldots, S_k\}$
    /*——Initialization——*/
 1: $\mathbb{S} = \emptyset$; //store the final segmentation
 2: map<int, int> svp; //<region No., count of svp>
 3: map<int, int> convex_svp; //<region No., count of convex svp>
 4: Assign label to each region, Label($\mathbb{R}$, $\mathbb{P}$);
    /*——Start the merging loop——*/
 5: for each $R_i$ in $\mathbb{R}$ do
      /*collect convex svp of adjacent regions*/
 6:   Collect the supervoxels pList in $R_i$;
 7:   for each $p_j$ in pList do
 8:     Gather its adjacency adjList from $\mathbb{P}$;
 9:     for each $p_l$ in adjList do
10:       Get the label $l$ of $p_l$;
11:       if ($i$!=$l$) then
            /*labels are different, find a svp*/
12:         Increase the count in svp related to $R_l$;
13:         Determine the convexity of $(p_j, p_l)$ by Eq.(4);
14:         if convex then
15:           Increase the count in convex_svp related to $R_l$;
16:         end if
17:       end if
18:     end for
19:   end for
      /*merge convexly connected facets*/
20:   for $k$=0 to svp.size()-1 do
21:     Get the region No. $j$ of the $k$th element in svp;
22:     Get the corresponding count $n_1$ in svp;
23:     Get the corresponding count $n_2$ in convex_svp;
24:     if ($n_1/n_2 \geq 0.5$) then
25:       Relabel and merge $R_j$ to $R_i$;
26:       Clear($R_j$);
27:     end if
28:   end for
29:   Clear svp and convex_svp for next iteration;
30: end for
    /*——Prepare the final result——*/
31: for $i$=0 to |$\mathbb{R}$|-1 do
32:   if $|R_i|$ ==0 then
33:     continue;
34:   else
35:     $S_k \leftarrow R_i$;
36:     $\mathbb{S} \leftarrow \mathbb{S} + S_k$;
37:   end if
38: end for
```

IV. EVALUATION AND DISCUSSION

In this section we evaluate proposed algorithm qualitatively and quantitatively on two indoor point cloud datasets, the Object Segmentation Database (OSD-v0.2 [38]) and NYU Indoor Database (NYU-v2 [40]). In addition to these results, we also compared it with four available methods, including the smoothness constrained region growing (RG [11]), the locally convex connected patches (LCCP [18]), and two deep learning based methods PointNet++ [27] and RSNet [29]. We compared the segmentations against groundtruth using four standard measures: *Weighted Overlap* (WOv) [40], *Mean Intersection of Union* (MIoU) [41], over- (f_{os}) and under-segmentation error (f_{us}) [38]. All the experiments are conducted on machine equipped with 32GB memory and Titan-Xp GPU.

A. PARAMETER ANALYSIS

Before evaluating proposed algorithm on two datasets, we carried out experimental analysis for parameters R_{voxel},

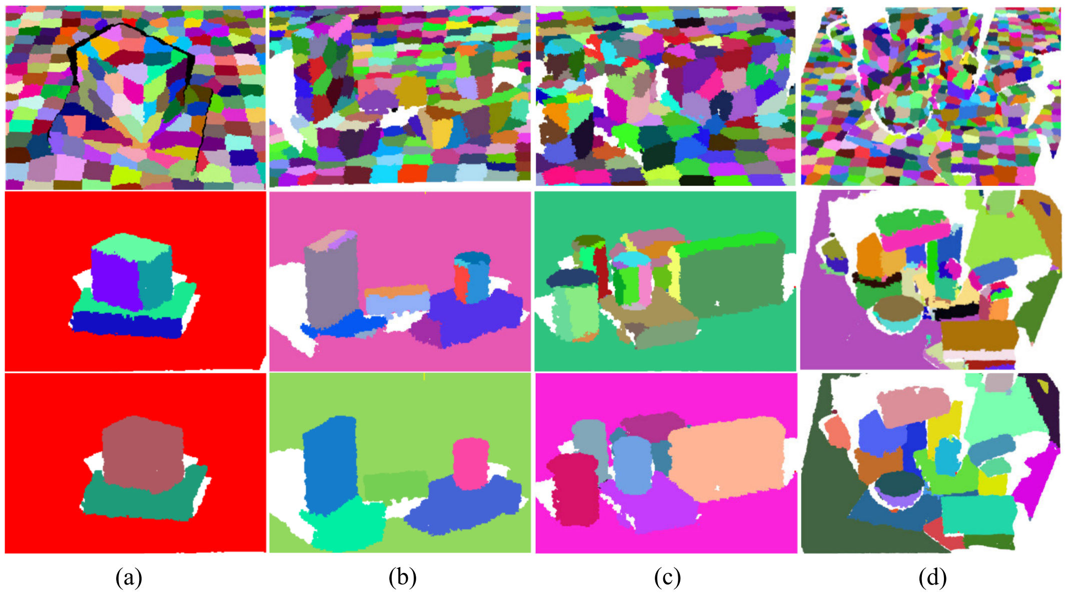


FIGURE 10. Examples of proposed merging process. From top to bottom: supervoxels, result from region growing, final result by facet merging. As the results show, supervoxel-based region growing pre-segments the scenes into local facets which are then merged for labeling different objects separately. Parameters: R_{voxel} = 0.007, R_{patch} = 0.06, β_{th} = 10°.

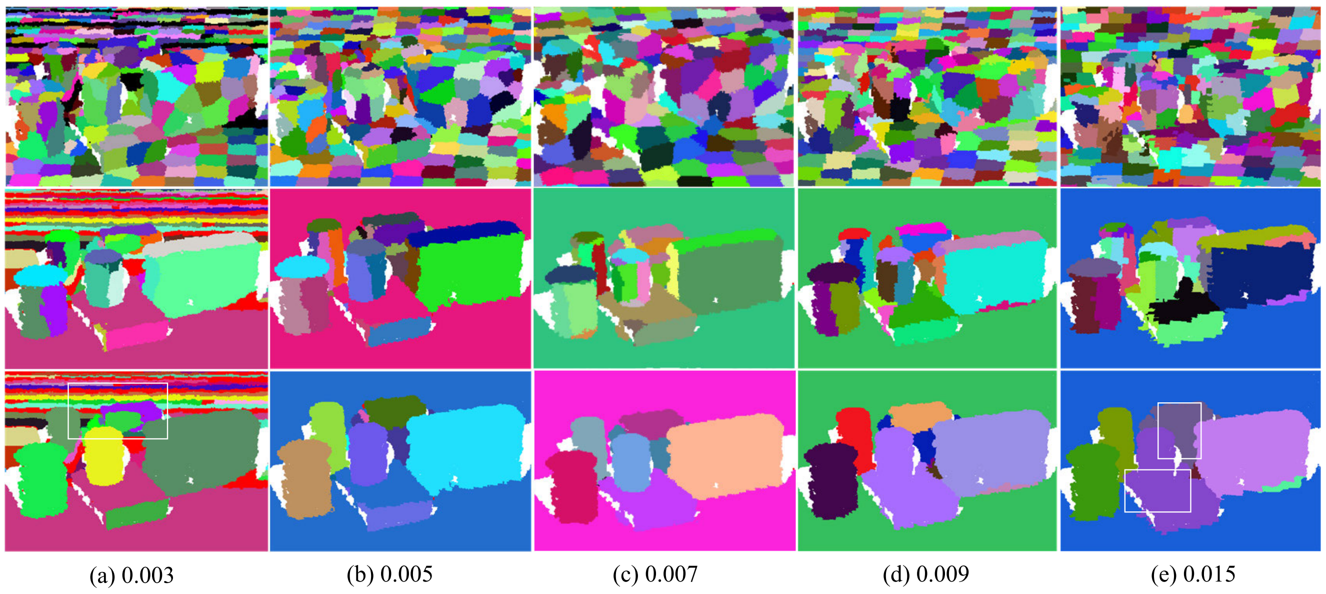


FIGURE 11. Examples of the influence of parameter R_{voxel} on segmentation when R_{patch} = 0.06 and β_{th} = 10°. From top to bottom: generated supervoxels, result from supervoxel region growing, the final segmentation by facet merging. (a)-(e): R_{voxel} = 0.003, 0.005, 0.007, 0.009, 0.015.

R_{patch} and β_{th} that impact the segmentation results to determine the optimal settings.

R_{voxel} and R_{patch} are the parameters involved in supervoxel generating process, which impact the quality of clustered supervoxels and the supervoxel-based segmentation. In this work, we performed the parameter evaluation by fixing one and varying the other to be analyzed. By setting R_{patch}=0.06 and β_{th}=10°, we tested the impact of R_{voxel} on supervoxel generation, region growing of supervoxels, and the final segmentation. Fig.11 examples the intermediate results and the final segmentation of proposed method under different R_{voxel}. It is observed that too small R_{voxel} (e.g. Fig.11(a)) makes the voxelization and the clustered supervoxels easily disturbed by data noise, which drags the performance of region growing and the final segmentation. Though large R_{voxel} can reduce the data volume of voxels and promote the efficiency of supervoxel generating, it could cause sawtooth phenomenon in supervoxels and result in incorrect merging (e.g. Fig.11(e)) due to the inadequate feature representing of original data for supervoxel clustering. Fig.12 calculates the average WOv and MIoU of the final segmentation on 10 randomly selected samples from

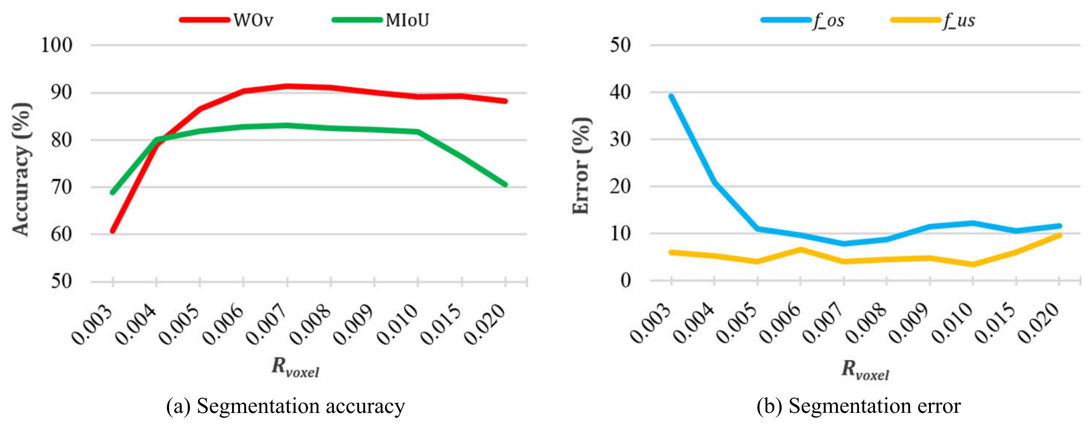


FIGURE 12. The curves of segmentation measures over R_{voxel} on 10 testing samples from OSD-v0.2 database when $R_{patch} = 0.06$ and $\beta_{th} = 10°$. (a) segmentation accuracy curves (WOv and MIoU), (b) segmentation error curves (f_{us} and f_{os}).

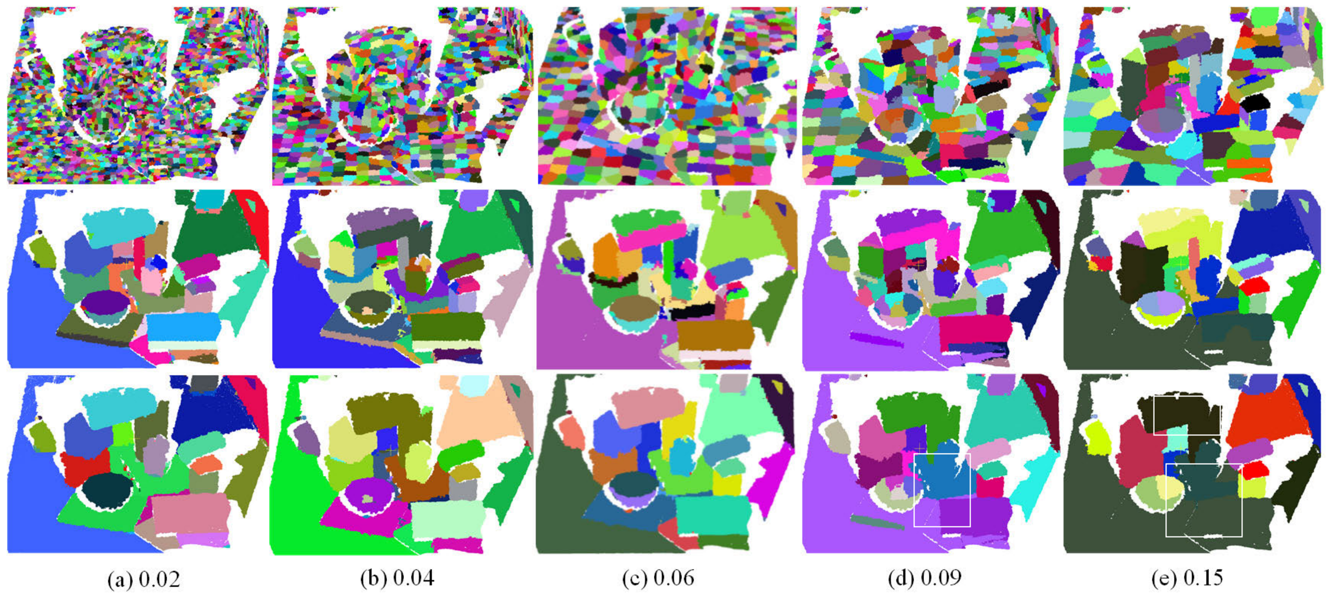
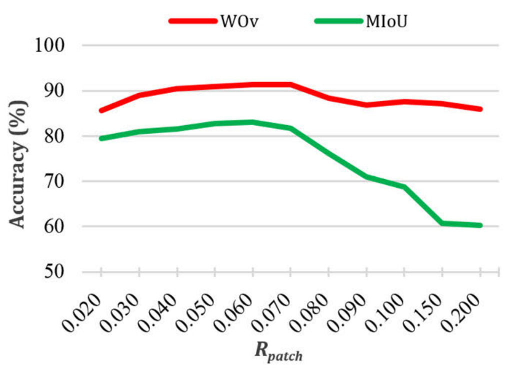


FIGURE 13. Examples of the influence of parameter R_{patch} on segmentation when $R_{voxel} = 0.007$ and $\beta_{th} = 10°$. From top to bottom: generated supervoxels, result from supervoxel region growing, the final segmentation by facet merging. (a)-(e): $R_{patch} = 0.02, 0.04, 0.06, 0.09, 0.15$.

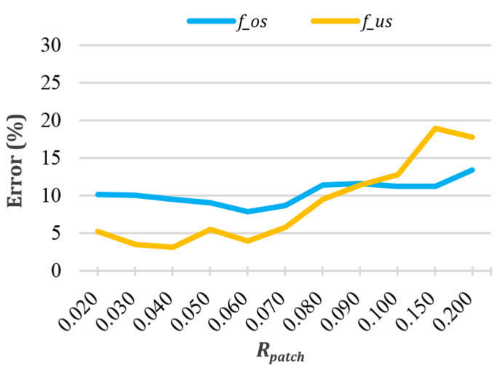
OSD-v0.2 dataset, as well as the f_{os} and f_{us} under varying R_{voxel}. The statistics are consistent with the intuitive observations, which reveals that proposed algorithm achieves good segmentation (high WOv and MIoU, low f_{us} and f_{os}) on OSD-v0.2 when R_{voxel} is in [0.005, 0.010]. More generally, it suggests that a ratio of 6 to 12 between R_{patch} and R_{voxel} is plausible.

Fig.13 illustrates the multi-stage results of proposed method under various setting of R_{patch} when R_{voxel} is fixed at 0.007. It shows that increasing R_{patch} brings about larger size of supervoxel and obvious under-segmentation in the final result (e.g. Fig.13(d) and (e), and the f_{us} curve in Fig.14(b)), while smaller patch size can lead to better segmentation. The intuitive results are verified by the figures in Fig.14, from which a conclusion can be drawn that the segmentation accuracy (WOv and MIoU in Fig.14(a)) on OSD-v0.2 drops when R_{patch} is larger than 0.07, and the segmentation error rises in the meantime. It is suggested that a relatively small R_{patch} is preferred to segment the objects of different sizes in the scene. By considering both the R_{voxel} and R_{patch} evaluations, we recommend the parameter setting of $R_{voxel} = 0.007, R_{patch} = 0.06$ for dataset OSD-v0.2 consisting of close desk scenes with high point density. For NYU-v2 which focuses on comparatively large room scenes with relatively low point density, the parameters should be adjusted accordingly. In our experiments, we selected $R_{voxel} = 0.01, R_{patch} = 0.08$ for NYU-v2 dataset on the basis of similar tests.

β_{th} is the tolerance threshold introduced to enhance the robustness of convexity determining in supervoxel fusing phase. It shows no obvious influence on the segmentation results when β_{th} is less than 45°, and begins to cause under-segmentation when setting it larger values, which is a trend that also can be estimated from its definition. Our tests on both databases show the similar conclusion, and we



(a) Segmentation accuracy



(b) Segmentation error

FIGURE 14. The curves of segmentation measures over R_{patch} on 10 testing samples from OSD-v0.2 database when $R_{voxel} = 0.007$ and $\beta_{th} = 10°$. (a) segmentation accuracy curves (WOv and MIoU), (b) segmentation error curves (f_{us} and f_{os}).

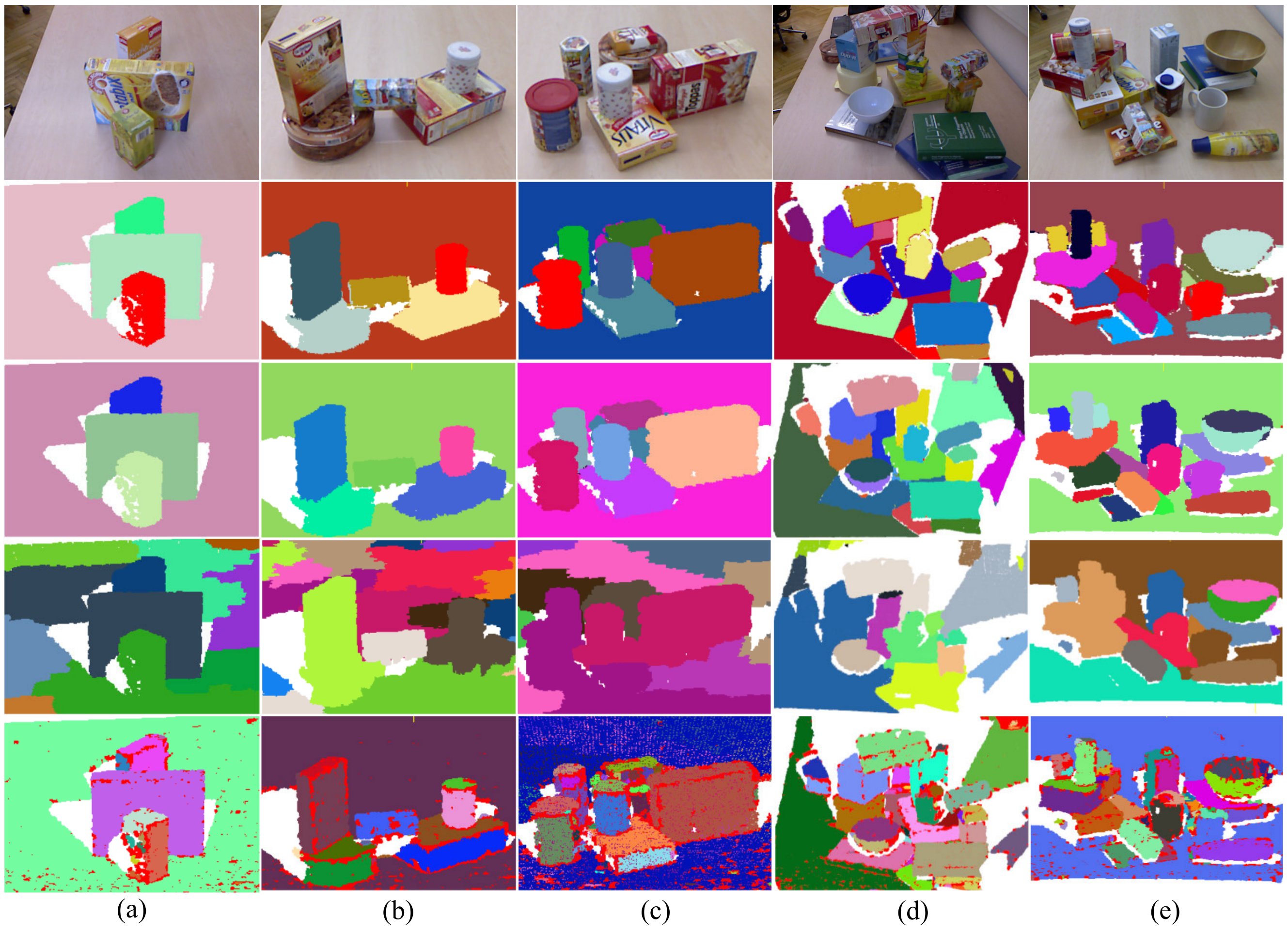


FIGURE 15. Comparison of segmentation result on point clouds from OSD-v0.2 database. From top to bottom: scene images, groundtruth, results from ours, LCCP, and RG. RG over-segments the each object to several parts with miscellaneous points. LCCP under-segments the point cloud, especially for simple scenes. Ours achieves the best segmentation and separates each object correctly, comparing to the groundtruth. Parameters: $R_{voxel} = 0.007$, $R_{patch} = 0.06$, $\beta_{th} = 10°$.

empirically set $\beta_{th} = 10°$ for both two databases in our evaluations by following Stein et al.'s setting [18].

B. OBJECT SEGMENTATION DATABASE (OSD)

The Object Segmentation Database (OSD-v0.2), developed by Richtsfeld et al. [38] in 2012, consists of 111 clustered scenes of objects on a table. The scenes contain multiple objects that have mostly box-like or cylindrical shape, with partial and full occlusions and heavy clutter in 2D as well as 3D. This makes the groundtruth data relatively non-ambiguous. However, this database contains no semantic labels for the objects included, making it nonapplicable

TABLE 1. **Comparison of different segmentation methods on the OSD-v0.2 database. Our results were produced with voxel size $R_{voxel} = 0.007$, seed size $R_{patch} = 0.06$, and concavity tolerance angle $\beta_{th} = 10°$.**

Method	WOv	mIoU	f_us		f_os		Average time (s)
	Mean	Mean	Mean	SD	Mean	SD	
Ours	**92.1%**	**83.6%**	4.0%	5.2%	**7.9%**	**9.3%**	0.29
Ours without convexity	86.4%	65.6%	**1.8%**	**2.6%**	13.6%	11.6%	0.28
LCCP[18]	91.4%	40.8%	24.7%	12.3%	8.6%	9.8%	**0.24**
RG[11]	83.1%	64.5%	6.6%	7.4%	16.9%	11.0%	17.61

TABLE 2. **Comparison of different segmentation methods on the NYU-v2 database. For the learning-free methods LCCP, RG, and ours, we directly gathered the average segmentation time for each point cloud. For learning-based methods PointNet++ and RSNet, the training time (15.4 hours and 42.8 hours, respectively) and average testing time for each point cloud were both collected. Our results were produced with voxel size $R_{voxel} = 0.01$, seed size $R_{patch} = 0.08$, and $\beta_{th} = 10°$.**

Method	WOv	mIoU	f_us		f_os		Average time
	Mean	Mean	Mean	SD	Mean	SD	
Ours	**74.4%**	**32.9%**	38.5%	14.7%	**25.6%**	8.9%	0.42s
Ours without convexity	67.8%	31.2%	**35.2%**	14.1%	39.3%	9.6%	0.41s
LCCP[18]	67.6%	17.9%	55.8%	16.4%	32.3%	8.5%	**0.34s**
RG[11]	66.2%	16.5%	54.0%	11.0%	33.8%	8.7%	85.14 s
PointNet++[27]	59.4%	10.9%	54.9%	**8.7%**	40.6%	**8.0%**	15.4h+0.88s
RSNet[29]	56.5%	10.6%	52.0%	9.9%	43.5%	8.3%	42.8h+2.28s

for PointNet++ and RSNet, so we just compare proposed method with LCCP and RG on this database.

Fig.15 shows the qualitative comparisons of three segmentation algorithms on five point clouds from OSD-v0.2. It is observed that RG is susceptible to noises and causes miscellaneous points at the object boundaries, which leads to outliers in segments (the bottom row). This is due to that RG only considers smoothness constraint in its growing process and cannot directly deal with sharp surfaces or edges. Owing to the supervoxel pre-segmentation, LCCP is robust to data noises and able to obtain smooth segmenting regions. However, it causes obvious under-segmentation (the fourth row). The major reason is that the merging strategy in LCCP just utilizes the convexity of adjacent supervoxels as merging standard, hence noisy patches would fail to or unexpectedly split the connected surfaces. Proposed algorithm, in contrast, successfully segments each single object from the table scenes (the third row). The objects are separated by the concave boundaries, and objects that have convex shape are labeled as one segment from others. Also, the distraction from data noises is commendably disposed of. These advancements own to the introduction of supervoxels and the two-phase merging strategy. Hollow objects, such as bowls, cups, etc., can be observed to show multiple segments. This is because that the normals change strongly on these concave surfaces. Comparing to the groundtruth (the second row), proposed algorithm realizes accurate segmentation of contained objects in point cloud scenes.

These qualitative observations are also supported by the quantitative results. Figures in Table1 demonstrate that our approach is able to compete with other two methods in segmentation of cluttered scenes with simple objects. Compared to the points-based method RG, we achieve better point precision (WOv and mIoU), the over- (f_{os}) and under-segmentation error (f_{us}) are also lower. Comparing to the supervoxel-based method LCCP, we obtain similar WOv and f_{os}, but better mIoU and f_{us}. This is because that LCCP struggles to segment simple objects from background, especially for simple scenes. Proposed convexity-guided facet merging strategy is an indispensable step for segmentation. Comparing to the method without merging the convexly connected facets (i.e. 'Ours without convexity' in Table1), proposed strategy greatly improves the segmentation precision and deals with the over-segmentation problem.

In terms of efficiency, supervoxel-based methods, LCCP and ours, show great advantage. Compared with point-based RG, the complexity of segmentation is significantly reduced when taking supervoxel over-segmentation as a preprocessing step. Compared to LCCP, proposed algorithm consumes 20.8% more time in supervoxel generation and facet merging, but this compromise, in return, greatly promotes the accuracy of object segmentation.

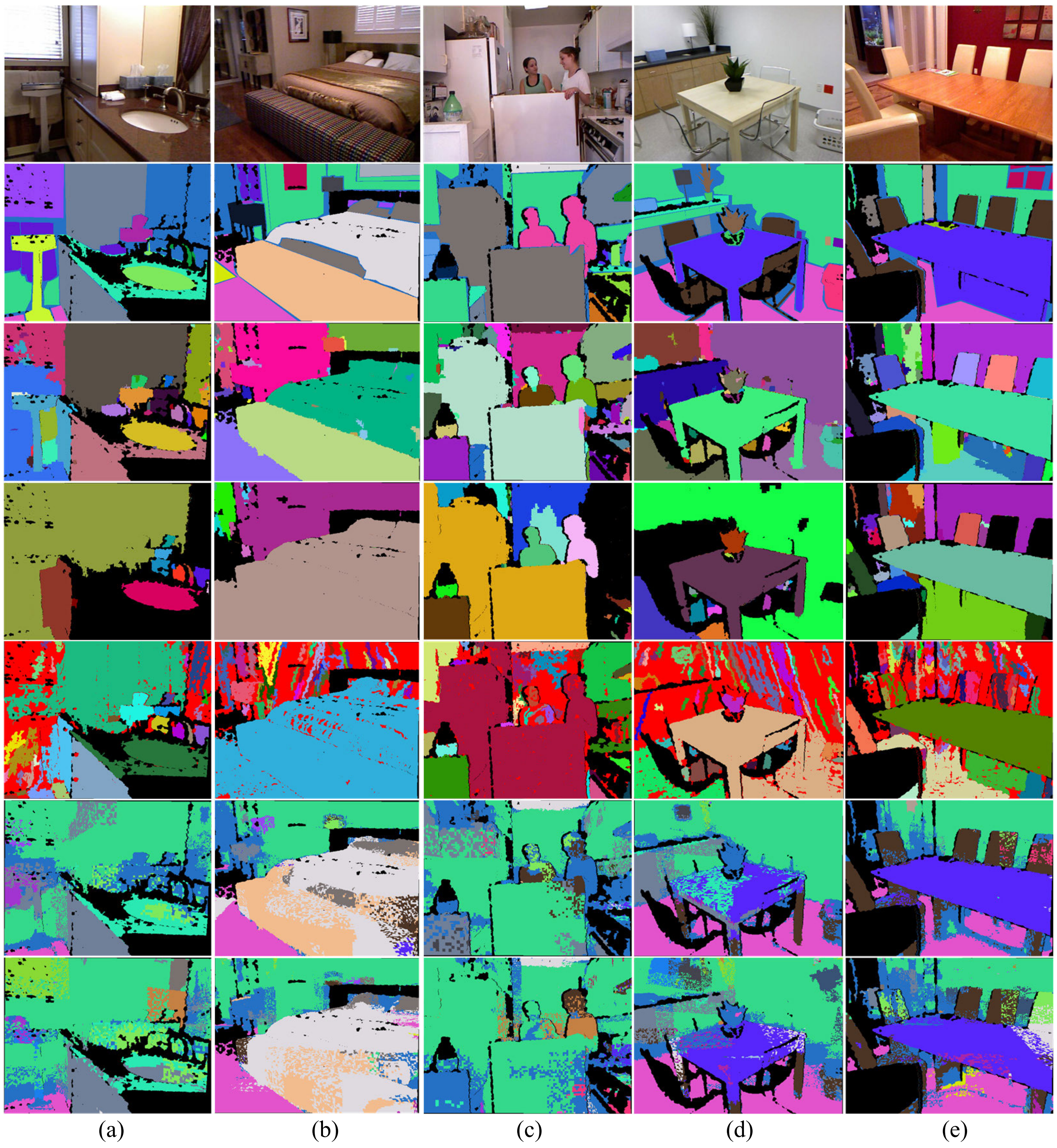


FIGURE 16. Comparison of segmentation result on point clouds from NYU-v2 database. From top to bottom: scene images, groundtruth, results from ours, LCCP, RG, PointNet++ and RSNet. LCCP under-segments the point cloud, RG over-segments the each object to several parts with miscellaneous points. The deep learning based methods PointNet++ and RSNet generate ambiguous segmentation labels and unspecific segmenting boundaries. Ours achieves the best segmentation and separates each object correctly, comparing to the groundtruth. Parameters: $R_{voxel} = 0.01$, $R_{patch} = 0.08$, $\beta_{th} = 10°$.

C. NYU INDOOR DATABASE(NYU)

The NYU Indoor Dataset (NYU-v2) presented by Silberman et al. [40] is a large complex dataset which consists of 1449 scenes under realistic cluttered conditions with 894 semantic labels for objects. In this work we evaluated proposed algorithm with all the four methods mentioned before on this dataset. All the segmentations were conducted on the raw point clouds in 3D space and then projected onto the RGB image for comparing to the labeled groundtruth.

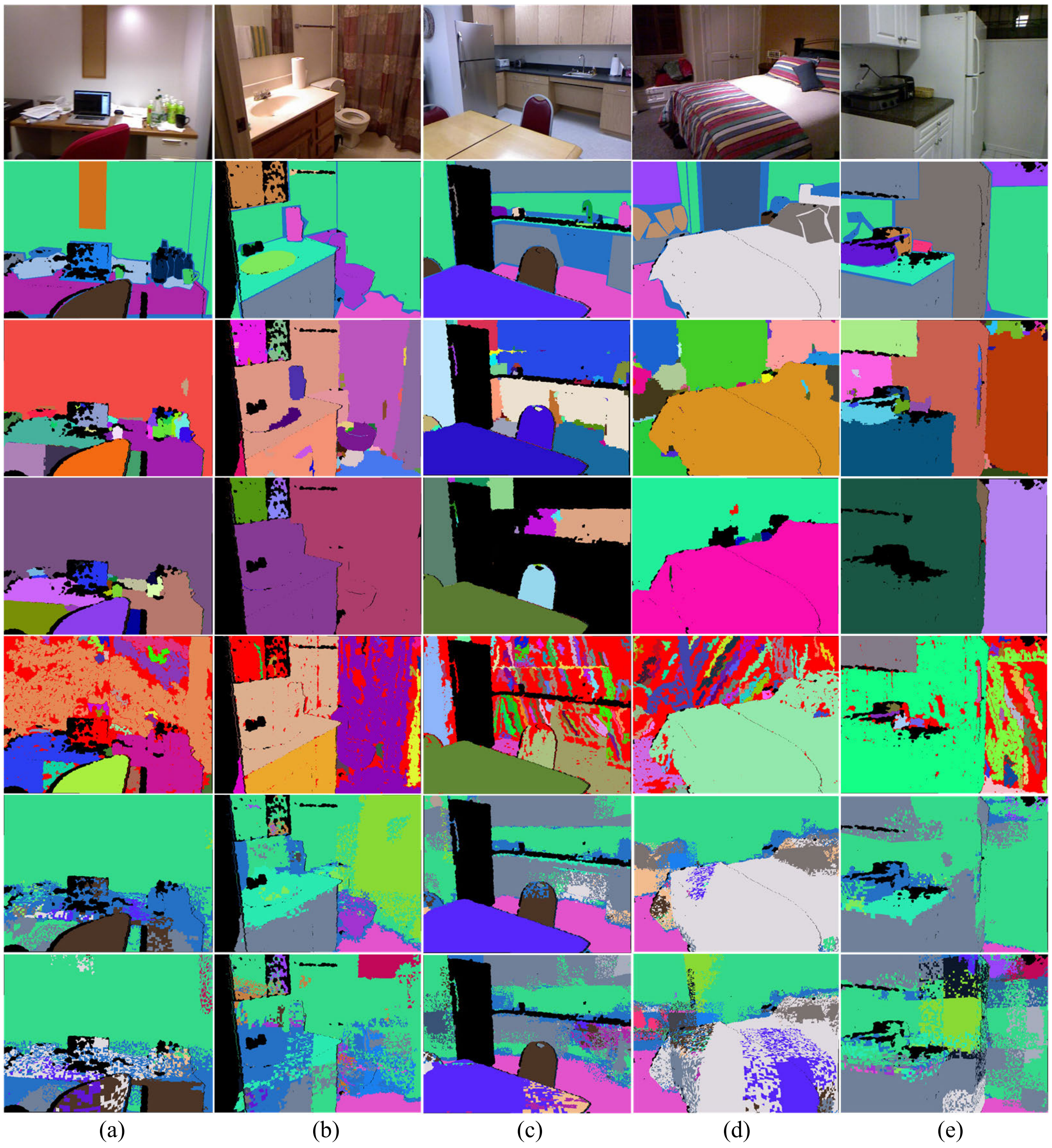


FIGURE 17. More results on point clouds from NYU-v2 database. From top to bottom: scene images, groundtruth, results from ours, LCCP, RG, PointNet++ and RSNet. Parameters: $R_{voxel} = 0.01$, $R_{patch} = 0.08$, $\beta_{th} = 10°$.

In contrast to the simple objects in the OSD-v0.2 database, objects in NYU-v2 database are composed of multiple complex parts. Example scenes in Fig.16 show that proposed algorithm (the third row) still performs good in object separation as expected. The wash basin, towel, and the tissue box in scene (a) are separated from the table. The wall is also segmented into several parts by the convexity. Scene (b) presents how a bedroom is partitioned into floor, bed, wall, and lamp. Scene (c) shows the segmentation in a kitchen. The persons are partitioned into head and upper body. The refrigerator and the objects on the table are correctly labeled. Scene (d) and (e) reveal that proposed method can output meaningful segments for conference rooms. In contrast, LCCP shows obvious under-segmentation and RG cannot deal with the noisy backgrounds. Meanwhile, The deep learning based methods PointNet++ and RSNet show no superior results on NYU-v2 database either. The segmentation labels for objects are ambiguous and hardly demonstrate specific segmenting boundaries. This is major due to the large amount of semantic labels and the noisy point clouds in NYU-v2 database.

It is evident that our labeling results disagree with the groundtruth in several cases, however, we do not consider either of the labeling wrong in general, but see them as different views on the data which are equally justifiable in many cases. Also, our algorithm is designed to segment objects by their geometrical convexity, very thin objects like the paintings on the background wall in scene (b) and (e) are not considered as independent instances. The results for more scenes are illustrated in Fig.17.

The quantitative results in Table2 show that proposed algorithm is able to generate good results on the challenging dataset. Our method achieves better WOv and mIoU, the under- and over-segmentation error (f_{us} and f_{os}) are relatively reasonable, which reflect the actual segmentation. Since the groundtruth is a kind of category tag, and the objects in different positions are labeled the same, the quantitative results on NYU-v2 database is not as competitive as that on OSD-v0.2 database. For the segmentation time, learning-free methods LCCP, RG, and ours can process each point cloud within half a second, comparing to that PointNet++ and RSNet respectively require 0.88 seconds and 2.28 seconds. And the learning-based approaches need large amounts of training time, 15.4 hours and 42.8 hours in our tests, which will be a major issue especially then the training sets are becoming increasingly large. Compared with LCCP, ours requires 23.5% more time to improve the segmentation, which is similar to the results on the OSD-v0.2 database.

D. DISCUSSION

As stated in Section II, semantic prior-knowledge, such as object structure or object relationship, can be utilized to enhance the segmentation. Proposed framework introduces the convexity of adjacent facets which is derived from the convexity of supervoxels, and uses the structure semantics to assist the segmentation. Benefiting from this novelty, proposed algorithm fuses the pre-segmented facets and achieved object-level segmentation (Fig.10). Experimental results (Fig.15/16/17) prove its advantages to available methods. Deep learning-based methods PointNet++ and RSNet can obtain good performance on relatively simple and accurate point clouds, but the segmenting confidence on complex datasets with major noises shows that further attention is required. RG operates directly on the original point cloud, whereas LCCP employs clustering on supervoxels with respect to the convexity of supervoxels. This idea inspires us the potential ways of describing the structure semantics of sophisticated objects, which can be applied to promote the efficiency and semantic accuracy of point cloud segmentation, not limited to the conventional methods or deep learning-based methods.

However, proposed algorithm has its limitations. Firstly, there are two more steps in our algorithm, boundary extraction and convexity fusion, hence more running time is required. Note that boundaries extraction and supervoxels generation can be parallelized using GPUs, which should lead to a 10-fold speed-up. Secondly, there are few small pieces in the segmented regions for NYU scenes, which is caused by the supervoxel-based region growing process and can be avoided by setting up large value of minimum-size. Overall, proposed algorithm achieves good segmentation for both simple and complex indoor point cloud scenes.

V. CONCLUSION

This work presents a convexity guided object segmentation algorithm for indoor point clouds. It follows the ''over-segmenting and merging'' principle which starts from the over-segmented supervoxels and then achieves object-level segmentation via the two-phase merging procedure.

The merit of the algorithm lies in fusing the pre-segmented adjacent facets with respect to their convexity relationship which is derived from the prior-observations of object structures. The convexity between two facets is determined by the convexity of their neighbouring supervoxel-pairs. Also the boundary information of point cloud is detected and introduced into the supervoxel clustering to avoid generated supervoxels crossing object boundaries. Improved supervoxels assure that the grown regions have legible boundaries and the followed region merging procedure output reliable segments with reasonable semantics. These contributions effectively advanced the accuracy of object segmentation.

In future work, we plan to promote the adaptability of convexity judging criterion, model the actual object according to the adjacency relationship among its surfaces, and then combine object recognition or retrieving with segmentation to pursue superior accuracy. Furthermore, investigating supervoxel-based deep learning segmentation approach to promote the training efficiency is also an interesting topic.

REFERENCES

[1] F. Husain, H. Schulz, B. Dellen, C. Torras, and S. Behnke, ''Combining semantic and geometric features for object class segmentation of indoor scenes,'' *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 49–55, Feb. 2017.

[2] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, ''Octree-based region growing for point cloud segmentation,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 104, pp. 88–100, Jun. 2015.

[3] S. A. A. Shah, M. Bennamoun, F. Boussaid, and L. While, ''Evolutionary feature learning for 3-D object recognition,'' *IEEE Access*, vol. 6, pp. 2434–2444, 2018.

[4] X. Ning, Y. Wang, W. Hao, M. Zhao, L. Sui, and Z. Shi, ''Structure-based object classification and recognition for 3D scenes in point clouds,'' in *Proc. Int. Conf. Virtual Reality Vis.*, Aug. 2014, pp. 166–173.

[5] F. Verdoja, D. Thomas, and A. Sugimoto, ''Fast 3D point cloud segmentation using supervoxels with geometry and color for 3D scene understanding,'' in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 1285–1290.

[6] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S. Zhu, ''Beyond point clouds: Scene understanding by reasoning geometry and physics,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3127–3134.

[7] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys, "3D modeling on the go: Interactive 3D reconstruction of large-scale scenes on mobile devices," in *Proc. Int. Conf. 3D Vis.*, Oct. 2015, pp. 291–299.
[8] A. Arnaud, M. Gouiffès, and M. Ammi, "On the fly plane detection and time consistency for indoor building wall recognition using a tablet equipped with a depth sensor," *IEEE Access*, vol. 6, pp. 17643–17652, 2018.
[9] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, "Voxel cloud connectivity segmentation—Supervoxels for point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2027–2034.
[10] Y. Lin, W. Cheng, D. Zhai, L. Wei, and J. Li, "Toward better boundary preserved supervoxel segmentation for 3D point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 143, pp. 39–47, Sep. 2018.
[11] T. Rabbani, F. van den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 36, no. 5, pp. 248–253, 2006.
[12] D. Holz, S. Holzer, and R. B. Rusu, "Real-time plane segmentation using RGB-D cameras," in *Proc. RoboCup, Robot Soccer World Cup XV*, 2011, pp. 306–317.
[13] Y. Wang and H. Shi, "A segmentation method for point cloud based on local sample and statistic inference," in *Geo-Informatics in Resource Management and Sustainable Ecosystem*, vol. 482. Berlin, Germany: Springer, 2015, pp. 274–282.
[14] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Sep. 2009, pp. 39–46.
[15] N. K. Sallem and M. Devy, "Extended GrabCut for 3D and RGB-D point clouds," in *Advanced Concepts for Intelligent Vision Systems*. Cham, Switzerland: Springer, 2013, pp. 354–365.
[16] J. R. Schoenberg, A. Nathan, and M. Campbell, "Segmentation of dense range information in complex urban scenes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 2033–2038.
[17] R. B. Rusu, A. Holzbach, N. Blodow, and M. Beetz, "Fast geometric point labeling using conditional random fields," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 7–12.
[18] S. C. Stein, M. Schoeler, J. Papon, and F. Wörgötter, "Object partitioning using local convexity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 304–311.
[19] J. Zhang, X. Lin, and X. Ning, "SVM-based classification of segmented airborne LiDAR point clouds in urban areas," *Remote Sens.*, vol. 5, no. 8, pp. 3749–3775, Jul. 2013.
[20] D. Wolf, J. Prankl, and M. Vincze, "Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 4867–4873.
[21] H. Jing and Y. Suya, "Point cloud labeling using 3D convolutional neural network," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 2670–2675.
[22] Z. Shu, C. Qi, S. Xin, C. Hu, L. Wang, Y. Zhang, and L. Liu, "Unsupervised 3D shape segmentation and co-segmentation via deep learning," *Comput. Aided Geom. Des.*, vol. 43, pp. 39–52, 2016.
[23] Y. Li, A. Dai, and L. Guibas, "Database-assisted object retrieval for real-time 3D reconstruction," *Comput. Graph. Forum*, vol. 34, pp. 435–446, May 2015.
[24] Z. Wu, S. Song, A. Khosla, Y. Fisher, Z. Linguang, and T. Xiaoou, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.
[25] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.
[26] R. Klokov and V. Lempitsky, "Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 863–872.
[27] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst. 30 (NIPS)*, 2017, pp. 5099–5108.
[28] S. Hang, V. Jampani, D. Sun, S. Maji, V. Kalogerakis, and M. H. Yang, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2530–2539.
[29] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2626–2635.
[30] S. Khan, M. Bennamoun, F. Sohel, R. Togneri, and I. Naseem, "Integrating geometrical context for semantic labeling of indoor scenes using RGBD images," *Int. J. Comput. Vis.*, vol. 117, no. 1, pp. 1–20, 2016.
[31] K. Chen, Y. K. 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 Trans. Graph.*, vol. 33, no. 6, pp. 1–12, 2014.
[32] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3D point clouds for indoor scenes," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.*, Granada, Spain, 2011, pp. 244–252.
[33] Y.-S. Wong and H.-K. Chu, "Annotating RGBD images of indoor scenes," presented at the SIGGRAPH Asia Indoor Scene Understanding Where Graphics Meets Vis., Shenzhen, China, 2014.
[34] T. Hackel, J. D. Wegner, and K. Schindler, "Fast semantic segmentation of 3D point clouds with strongly varying density," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inform.*, vol. 3, no. 3, pp. 177–184, 2016.
[35] G. G. Demisse, D. Borrmann, and A. Nüchter. "Interpreting thermal 3D models of indoor environments for energy efficiency," *J. Intell. Robot. Syst.*, vol. 77, no. 1, pp. 55–72, 2015.
[36] C. Choi, A. J. B. Trevor, and H. I. Christensen, "RGB-D edge detection and edge-based registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1568–1575.
[37] H. Ni, X. Lin, X. Ning, and J. Zhang, "Edge detection and feature line tracing in 3D-point clouds by analyzing geometric properties of neighborhoods," *Remote Sens.*, vol. 8, no. 9, p. 710, 2016.
[38] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4791–4796.
[39] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," *KI-Künstliche Intell.*, vol. 24, no. 4, pp. 345–348, 2010.
[40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2012, pp. 746–760.
[41] G. G. Alberto, O. E. Sergio, and O. Sergiu, "A review on deep learning techniques applied to semantic segmentation," Apr. 2017, *arXiv:1704.06857*. [Online]. Available: https://arxiv.org/abs/1704.06857

NAN LUO was born in Xianyang, Shannxi, China, in 1987. He received the B.S. degree in computer science and technology, and the M.S. and Ph.D. degrees in computer architecture from Xidian University, Xi'an, in 2009, 2012, and 2017, respectively.

He is currently a Postdoctoral Research Fellow with the Computer Engineering Department, Xidian University. He has been a member of the China Computer Federation(CCF), since 2012. His research interests include point cloud processing, as well as 3D scene reconstruction and understanding.

QUAN WANG was born in Shaanxi, in July 1970. He received the B.Eng., M.Sc., and Ph.D. degrees in computer application, computer device and equipment, computer application from Xidian University, Xi'an, China, in 1992, 1997, and 2008, respectively. He is currently a Professor with Xidian University.

Since 1992, he has been with the School of Computer Science and Technology, Xidian University. From 1998 to 1999, he was a Visiting Scholar at Fujitsu (OITA) Software Lab, Japan. From 2014 to 2018, he was the Dean of the School of Computer Science and Technology, Xidian University. He is currently the Assistant President of Xidian University, a distinguished member of the China Computer Federation(CCF), the Councilor of the ACM Xi'an Section, the Vice President of the Image Science and Engineering Branch of China Instrument and Control Society, and the Vice Director of the Committee of Embedded System of CCF. His current research interests include input and output technology and system, as well as image processing and understanding.

QI WEI was born in Linfen, Shanxi, China, in 1993. She received the B.S. degree in computer science and technology from Shanxi University, in 2015, and the M.S. degree in computer architecture from Xidian University, in 2018.

She is currently an Employee with Huawei Technologies Corporation. Her research interests include image-based 3D reconstruction and point cloud segmentation.

CHUAN JING was born in Mianyang, Sichuan, China, in 1993. He received the B.S. degree in computer science and technology from Northeastern University at Qinhuangdao, in 2016, and the M.S. degree in computer architecture from Xidian University, in June 2019. His current research focuses on the registration and semantic segmentation of 3D point clouds.