A Technical Survey and Evaluation of Traditional Point Cloud Clustering Methods for LiDAR Panoptic Segmentation

Yiming Zhao Xiao Zhang Xinming Huang
Worcester Polytechnic Institute
100 Institute Rd, Worcester, MA, USA
{yzhao7, xzhang25, xhuang}@wpi.edu

Abstract

LiDAR panoptic segmentation is a newly proposed technical task for autonomous driving. In contrast to popular end-to-end deep learning solutions, we propose a hybrid method with an existing semantic segmentation network to extract semantic information and a traditional LiDAR point cloud cluster algorithm to split each instance object. We argue geometry-based traditional clustering algorithms are worth being considered by showing a state-of-the-art performance among all published end-to-end deep learning solutions on the panoptic segmentation leaderboard of the SemanticKITTI dataset. To our best knowledge, we are the first to attempt the point cloud panoptic segmentation with clustering algorithms. Therefore, instead of working on new models, we give a comprehensive technical survey in this paper by implementing four typical cluster methods and report their performances on the benchmark. Those four cluster methods are the most representative ones with real-time running speed. They are implemented with C++ in this paper and then wrapped as a python function for seamless integration with the existing deep learning frameworks. We release our code for peer researchers who might be interested in this problem [1].

1. Introduction

Panoptic segmentation is an ensemble of both the semantic segmentation for static stuff and the instance segmentation for countable objects. Point cloud panoptic segmentation needs to provide the semantic label id for each point and further assign a unique instance label to points that belong to the same object [27]. With the understanding of both the semantic and instance information from a single frame LiDAR scan, this task is able to deliver many useful clues for advanced autonomous driving functions, such as future prediction [16] and map building [7].

[1]https://github.com/placeforyiming/ICCVW21-LiDAR-Panoptic-Segmentation-TradiCV-Survey-of-Point-Cloud-Cluster

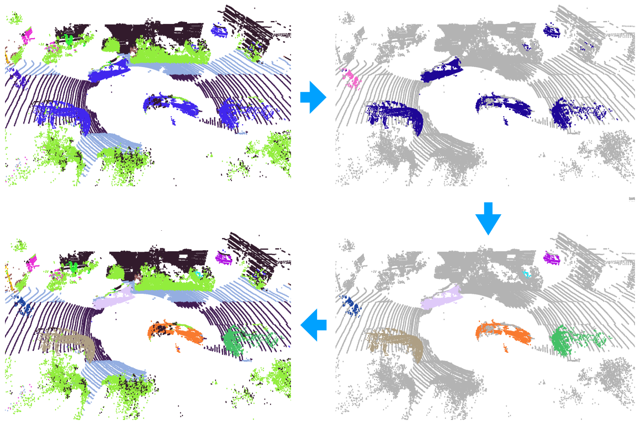
Figure 1: The demonstration of the panoptic segmentation task and how our pipeline solves it. After the semantic segmentation, the clustering algorithm works on object points to further segment each instance.

Motivation. Most existing works solve the LiDAR panoptic segmentation with end-to-end deep learning models. A neural network is in charge of both the point-wise classification and the point-wise clustering. Nowadays, almost all classification tasks are dominated by neural networks. However, considering the 3D geometry information implicitly encoded in the point cloud, it is questionable if the neural network is good at clustering at the same time. At least, instead of directly applying the neural network as the solution, we should evaluate several questions. For instance, do we really need the neural network for point-wise clustering? If needed, how much the neural network solution can outperform traditional methods? Answer those questions need a solid benchmark to investigate and compare existing traditional clustering methods on well-recognized datasets. This motivates us to prepare this technical survey paper of the traditional point cloud clustering method on the SemanticKITTI point cloud panoptic segmentation task.

Point cloud clustering is a topic studied in multiple domains, including robotics [43] and intelligent transportation [34]. This technique has been used in various applications. For example, some recent point cloud compression methods need clustering algorithms to downsample points [37, 38]. The non-learning clustering algorithm is also proved useful for some standard tasks like semantic segmentation [22] and object detection [44].

In this paper, we propose to only rely on an existing neural network for the semantic classification part, then process the point-wise clustering part with traditional LiDAR cluster algorithms. As far as we know, we are the first to propose solving panoptic segmentation with the traditional point cloud clustering algorithm. Therefore, instead of developing new techniques, we would rather conduct a comprehensive technical review to investigate the performance of all existing methods. Specifically, we pick the state-of-the-art semantic model Cylinder3D [45, 47] to provide the semantic label of each point. Then we run various LiDAR clustering algorithms to obtain the instance label of each object. Cylinder3D has open-source code with pretrained checkpoints, thus we can easily use the same semantic model to conduct a fair comparison among different clustering methods.

Under this setting, we evaluate various representative cluster methods implemented by us on the well-known semanticKITTI dataset [3, 27]. We discuss more details about each selected clustering method in Section III. The same as the standard panoptic segmentation, PQ (panoptic quality) is used to measure the panoptic performance of the clustering method. We also report other indicators used in SemanticKITTI when compare with methods on the leaderboard.

Contributions. We believe this survey paper will be beneficial for both academic research and industrial applications. We summarize the contributions of this paper below:

- *We propose a new framework for LiDAR panoptic segmentation.* We are the first to demonstrate the possibility of solving the LiDAR panoptic segmentation with a semantic network and a traditional clustering method. This solution outperforms all end-to-end network solutions published recently. The classic clustering method runs in millisecond-level on CPU, alleviates labeling effort on the instance part, and has a chance to adapt to new unseen scenarios better because it has no bias toward the training set. As a traditional method, it is not a network-style black-box, thus gives people a clear idea of when the method performs better or worse.

- *We set up a benchmark for LiDAR clustering algorithms.* An issue of previous research about the LiDAR clustering method is the ambiguity of evaluation metrics due to the existence of large non-object surfaces such as walls and ground. In this paper, we filter out all those non-object points by using a semantic model with published code and checkpoints [2]. So the clustering algorithm can only focus on points that belong to objects. We further use the well-recognized evaluation metrics [36, 30] of panoptic segmentation to directly evaluate and compare the effectiveness of different clustering methods. Thanks to the high quality of the SemanticKITTI dataset and leaderboard [3, 27], future research about LiDAR clustering topic can follow our benchmark as the de facto comparison baseline.

2. Literature Review

In this section, we give a brief literature review of the panoptic segmentation task as well as the point cloud cluster method. More technical details about selected cluster methods will be introduced in the next section.

2.1. Panoptic Segmentation

Panoptic segmentation is a newly proposed task to fully understand a single image frame [19]. A solution to panoptic segmentation is expected to simultaneously classify pixels that belong to stuff as well as recognize each individual countable thing. The uncountable amorphous region of identical texture is known as stuff, for example, road, water, sky, etc. Things are those countable objects like persons, cars, etc. As this task is a combination of semantic segmentation and instance segmentation, it is natural to extend many deep learning models originated from existing semantic or instance solutions with extra modifications to meet the requirement of the panoptic benchmark [8, 40].

2.2. LiDAR Panoptic Segmentation

LiDAR panoptic segmentation is a counterpart of image panoptic segmentation on the point cloud. The model needs to give the semantic label for each point and group points that belong to the same instance together [27]. The range image representation gives the convenience of directly modifying image-based methods on the point cloud, including both the one-stage DeepLab style method [10] and two-stage Mask R-CNN style method [35]. Considering specific 3D information encoded in point cloud inspired some specially designed network structures, such as a dynamic shifting module developed in [17].

As a very new task [3, 27] proposed in this deep learning era, many researches directly dive into the deep learning solutions. However, despite the semantic classification part, point cloud clustering is a long-existing research topic that also has a chance to contribute as part of the panoptic task.

2.3. Point Cloud Cluster Methods

The point cloud is a common representation of the 3D world. How to cluster objects from the point cloud is a

[2]https://github.com/xinge008/Cylinder3D

long-standing problem in the literature. Instead of the hot topic of dealing with 3D points using neural networks, we investigate traditional geometry-based point cloud clustering methods. We want to show those underestimated classic methods are valuable assets for solving real-world computer vision challenges if they are properly combined with recent deep learning models.

Clustering with Euclidean Distance. Using the Euclidean distance to cluster points is a straightforward idea explored in [20], authors developed a radially bounded nearest neighbor (RBNN) algorithm for the general point cloud. They further extended RBNN by considering the normal vector [21], which makes the algorithm prone to segment surfaces. Since LiDAR is mostly used in the outdoor scenario, segmenting the road surface is crucial. A novel ground segmenting algorithm was proposed in [9], other non-ground points were clustered with voxelized Euclidean neighbors. In [15], researchers provided a probabilistic framework to incorporate not only the Euclidean spatial information but also the temporal information from consecutive frames. The under-segmentation error and over-segmentation error used in [15] also shed a light on how we should compare and evaluate various point cloud clustering methods.

Clustering with Supervoxels or Superpoints. Without human knowledge, it is hard to define objects from the raw point cloud. Inspired by the concept of superpixels from the traditional image processing [1], some researchers are interested in finding super voxels in the Euclidean space. In [29], authors proposed a voxel cloud connectivity segmentation (VCCS) method which extends the definition of distance used in the iterative image pixel cluster SLIC [1]. From the view of clustering, super voxels or super points usually over segment objects as presented and discussed in [5, 23].

Clustering on Range Image. Besides the naive Euclidean distance [20, 21, 9], researchers explored more clues aiming at finding better criteria to separate neighbor points belong to different clusters. In [6, 42], the angle formed by two adjacent laser beams is considered to construct the discriminator. To make the algorithm fast enough for real-time applications, authors of [6] worked on the 2D range image representation of the LiDAR point cloud. This led to several works which proposed fast clustering methods on LiDAR range images by borrowing ideas from connected-component labeling (CCL) algorithms[43, 12]. The connected-component labeling (CCL) is a graph algorithm used in computer vision to detect connected regions in binary digital images [14]. Due to the inherent difference between binary images and LiDAR range images, authors of [43, 12] modified existing CCL methods, such as two-pass CCL [41] or run-based CCL [13], so that the clustering algorithms can run at millisecond level.

Post-Process for Over-Segmentation. Over-segmentation means a single object is wrongly clustered into multiple parts. Successfully merging over segmented parts will improve the clustering quality for some models. In [34], a Gaussian process regression method is proposed to improve the accuracy as a post-process step. Some heuristic methods inspired by the 3D information are also attempted in [25, 26]. Those heuristic methods mainly consider the fact that two objects can not stay too close.

Evaluation Metrics. Earlier publications in this field only evaluated the clustering method on private datasets with unique settings [20, 21, 9]. Recent papers [15, 34, 43, 42, 12] tried to use the popular KITTI dataset [11]. However, there are still some problems in those papers. For example, [15] evaluated cluster performance on the KITTI tracking dataset but only considered objects within 15 meters.

The lack of a well-recognized benchmark is a problem that limits further research about point cloud clusters. In this paper, we propose to utilize the panoptic segmentation task on SemanticKITTI as a benchmark. All the clustering methods should share the same semantic segmentation model to let the cluster only focus on points that belong to instances. We use the open-sourced state-of-the-art Cylinder3D [45, 47] as the semantic model in this paper. A series of indicators designed for panoptic segmentation, including PQ, $PQ^{\dagger}$, etc, are able to measure and compare different cluster solutions as they share the same semantic part.

3. Selected Reviewed Methods

Existing point cloud clustering methods can be broadly summarized as four types, the Euclidean-based cluster in 3D space, clustering point cloud into supervoxel or superpoint, modified one-pass connected-component labeling on range image and modified two-pass connected-component labeling on range image. In this method review section, we pick the most typical algorithm in each type and give a more detailed introduction.

3.1. Euclidean Cluster

Euclidean cluster is a straightforward clustering method. It firstly constructs the kd-tree on the entire point cloud, then groups all neighbor points within a radius threshold as one instance. We illustrate the Euclidean cluster in Alg. 1. There is one parameter, the radius threshold. Intuitively, a larger threshold will group close objects together, but a smaller threshold is more sensitive to object parts with few points. Thus we choose a moderate 0.5 meter as the threshold in this paper. More details about the Euclidean cluster can be referred in [31, 32].

Algorithm 1 Euclidean Cluster

```
Input  : Point cloud $\mathbf{P}$, Euclidean distance threshold $\mathbf{d_{th}}$
Output: A list of labels for each point $\mathbf{C}$
Create a kd-tree to represent $\mathbf{P}$;
$\mathbf{C} = list([0, .., 0])$, $label = 1$;
foreach point $P_i$ in $\mathbf{P}$ do
    if $C_i > 0$ then
        Continue;
    end
    $\mathbf{S}(\mathbf{P_i}) \leftarrow$ a set of points near $p_i$ in a sphere with radius $r < d_{th}$;
    $C_i = label$;
    foreach point $P_i$ in $\mathbf{S}(\mathbf{P_i})$ do
        if $C_i == 0$ then
            $C_i = label$;
        end
    end
    $label = label + 1$;
end
return C
```

3.2. Supervoxel Cluster

Superpixel SLIC [1] is a well-known traditional image processing operator that groups local pixels to a larger one with common characteristics. Supervoxel [29] is designed on RGB-D point cloud as a counterpart of superpixel on the 2D image.

Compared with superpixel, there are three major differences of supervoxel. The first one is about initial seeds. In supervoxel, the seeding of clusters is done by partitioning 3D space, rather than the projected image plane. The second difference is an extra constrain that the iterative clustering algorithm enforces strict spatial connectivity of occupied voxels when considering points for clusters. The third one is the definition of the distance used in k-means. In supervoxel, instead of the distance on 2D image, the 3D Euclidean distance and the angle of the normal vector are further considered along with the color similarity. Note that the definition of supervoxel distance in point cloud library (PCL) [32] is different with original paper [29]. We choose the one implemented in PCL in this paper.

The distance D is defined in Eq. 1. The spatial distance D_s is normalized by the seeding resolution, color distance D_c is the euclidean distance in normalized RGB space, and normal distance D_n measures the angle between surface normal vectors. w_c, w_s, and w_n, are the color, spatial, and normal weights, respectively.

$$D = \sqrt{w_c D_c^2 + \frac{w_s D_s^2}{3R^2} + w_n D_n^2}. \tag{1}$$

In this paper, we are working on the LiDAR point cloud without RGB color information, thus the color distance D_c is set as zero for all points. The left iterative k-means of supervoxel is the same as superpixel SLIC [1] on 2D images. Eq. 1 helps the method balance the local normal and the local Euclidean distance.

3.3. Depth Cluster

Depth cluster [6] is a fast one-pass CCL (connected-component labeling) algorithm on the LiDAR range image. The CCL algorithm on the binary image needs to check if two neighbor pixels both have intensity one. However, the CCL on the LiDAR range image is required to define the condition that determines if two neighbor points are coming from the same object. In depth cluster algorithm, this condition is defined by using a magical angle β shown in Fig. 2. Authors [6] argue if the β is larger than an angle threshold θ then point A and point B are coming from the same object. With this condition, the depth cluster algorithm is summarized in Alg. 2. In [6], they choose $\theta = 10^o$ as the threshold value that is also used as a fixed parameter in this survey.

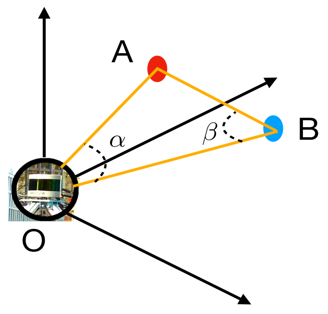

Figure 2: For two neighbor points A and B emitted from LiDAR O, whether the angle β between line BA and line BO is larger or smaller than a threshold θ is the condition to determine if point A and B coming from the same object or not.

Algorithm 2 Depth Cluster

```
Input  : Range image $\mathbf{R}$
Output: Label image $\mathbf{L}$
$\mathbf{L} \leftarrow zeros(R_{rows} \times R_{cols})$,    $label \leftarrow 1$;

for $r = 1, ..., R_{rows}$ do
    for $c = 1, ..., R_{cols}$ do
        if $L(r, c) == 0$ then
            LabelComponentBFS$(r, c, label)$;
            $label = label + 1$;
        end
    end
end
return L
Procedure LabelComponentBFS$(r, c, label)$
    $queue.push(\{r, c\})$;
    while queue is not empty do
        $\{r, c\} \leftarrow queue.top()$;
        $L(r, c) \leftarrow label$;
        for $\{r_n, c_n\} \in Neighborhood\{r, c\}$ do
            $d_1 \leftarrow max(R(r, c), R(r_n, c_n))$;
            $d_2 \leftarrow min(R(r, c), R(r_n, c_n))$;
            if $arctan\frac{d_2 sin\alpha}{d_1 - d_2 cos\alpha} > \theta$ then
                $queue.push(\{r_n, c_n\})$;
            end
        end
        $queue.pop()$;
    end
end
```

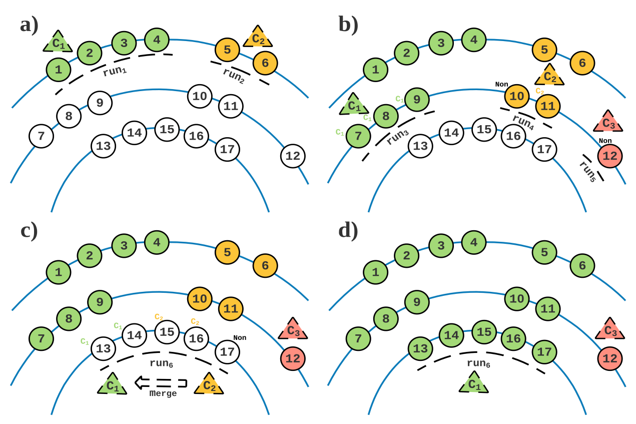

Figure 3: This figure illustrates how Scan-line Run cluster the point cloud. **a)** The first scan-line is segmented as two runs, denoted as run_1 and run_2 with labels C_1 and C_2 respectively. **b)** The second scan-line is firstly segmented as run_3, run_4 and run_5, then merged with existing clusters labeled as C_1 and C_2 from the previous line. The run_5 does not fit merging condition with previous clusters, thus receives a new cluster label C_3. **c)** When two clusters meet in the third line, they will be merged with the smaller cluster label. **d)** The result after merging two clusters.

3.4. Scan-line Run Cluster

Scan-line Run (SLR) cluster [43] is a row-wise fast scan algorithm based on organized point cloud or range image. This method is a counterpart of the image-based two-pass connected component labeling (CCL) algorithm on the LiDAR range image. In SLR, all points emitted from the same horizontal angle are recognized as one scan-line. In a single scan-line, all nearby points closer than a threshold Th_{run} are grouped together, called a run.

In the beginning, SLR takes in the first line, then groups all nearby points with the Euclidean distance smaller than a threshold Th_{run} together as a run. Each run is assigned a unique label as initial clusters. Next, the SLR moves to the second line repeat the run segmenting and check if the new run in the second line meets the merging condition defined with a new threshold Th_{merge}. Two runs will be merged together if they satisfy the merging condition. The label will also propagate. If a new run in the second line does not fit the merging condition with any previous runs, a new cluster label will be assigned. For the case, if two clusters meet in a new line, SLR will merge them with the smaller cluster id. This procedure will keep moving line by line till all LiDAR scan lines are processed. We visualize the procedure in Fig. 3 by considering the first three lines.

The algorithm is summarized in Alg. 3. The FindNearestNeighbor function is aiming at searching the closest point in the previous scan-line. The original paper [43] provided several ways to query the nearest neighbor with pros and cons. More details about the SLR cluster can be referred to it [43].

Algorithm 3 Scan-line Run

```
Input  : Organized $N$ line Point Cloud $\mathbf{P} = \{P_1, P_2, ..., P_N\}$
Output: Cluster ID of each point $\mathbf{L} = \{L_1, L_2, ..., L_N\}$
Initilization
  $P_i$: $ith$ ordered scan-line with range value of each point
  $L_i$: cluster id of each scan-line, initialized with zero
  $th_{run} \leftarrow 0.5m$ : threshold to group points to a run
  $th_{merge} \leftarrow 1.0m$ : threshold to merge runs to a cluster
  $label_{idx}$ : vector of point index, each label has an independent row

$runsCurrent = \mathbf{FindRuns}(P_1, th_{run})$;
  for $run$ in $runsCurrent$ do
      $label_{idx}.push\_back((i, run))$ ;
      $L_1[run] = label_{idx}.size$;
end
$runsAbove = runsCurrent$ ;
  for $i = 2 : N$ do
      $runsCurrent = \mathbf{FindRuns}(P_i, th_{run})$;
      $\mathbf{UpdateLabelIndex}(runsCurrent, runsAbove)$;
      $runsAbove = runsCurrent$ ;
end
  Assign label from $label_{idx}$ to L;
  return L

  Procedure FindRuns($P_i$, $th_{run}$)
      $run\_all \leftarrow [\,]$, $run\_each \leftarrow [\,]$;
      for $index$ in $P_i.size$ do
          if $is\_empty(run\_each)$ then
              $run\_each.append(index)$; $continue$;
          end
          if $abs(P_i[index] - P_i[index-1]) < th_{run}$ then
              $run\_each.append(index)$; else
                $run\_all.append(run\_each)$; $run\_each \leftarrow [\,]$;
          end
      end
Return $run\_all$
Procedure UpdateLabelIndex($runsCurrent, runsAbove$)
      for $run$ in $currentRuns$ do
          for $index$ in $run$ do
              $index_{nn} = \mathbf{FindNearestNeighbor}(index, runsAbove)$;
              if $abs(P_i[index] - P_{i-1}[index_{nn}]) < th_{merge}$ then
                  $L_i[index] = L_{i-1}[index_{nn}]$;
              end
          end
          if $sum(L_i[run]) == 0$ then
              $label_{idx}.push\_back((i, run))$;
              $L_i[run] = label_{idx}.size$;
          end
          $runLabel = min(L_i[run])$;
            $L_i[run] = runLabel$;
            for $eachLabel$ in $L_i[run]$ do
              if $eachLabel > runLabel$ then
                  $mergedLabel = label_{idx}[eachLabel-1]$;
                    $label_{idx}[eachLabel-1].delete$;
                    $label_{idx}[runLabel-1].insert(mergedLabel)$;
              end
          end
      end
end
```

3.5. Implementations

We implement all aforementioned four methods with C++ and wrap implementations with Pybind11 [18] as python functions. Although those methods are designed for point cloud clustering, none specifically focuses on panoptic segmentation. To make them adapt better to the SemanticKITTI dataset, some minor modifications are introduced.

Euclidean Cluster. We directly use functions in PCL (point cloud library)[32] to implement the Euclidean cluster. Due to the use of the kd-tree data structure, the time complexity of the Euclidean cluster is non-linearly depending on the total number of points. We subsample the whole point cloud, and randomly choose one point from each $10cm \times 10cm \times 10cm$ voxel. The same instance label will be assigned to other points from the same voxel.

Supervoxel Cluster. We discard the RGB color distance as we are working on the LiDAR point cloud.

Depth Cluster. The depth cluster assumes each point on the range image will have neighbors on adjacent pixels. However, the original data provided by KITTI is an unordered 3D point cloud. The range image generated by mapping this point cloud will contain many holes. To overcome this problem, our implementation will keep searching the neighbor within a threshold until finding the neighbor point.

Scan-line Run Cluster. Scan-line Run cluster will meet the same problem as Depth Cluster that many holes exist between LiDAR points. Therefore, we develop a make-up search strategy. The make-up search will brute-force scan every point from the above two lines, instead of the original above one line, to find the nearest points. The make-up search is triggered when searching in above one line failed, thus only slightly increases the time cost.

4. Experiments

In this section, we firstly compare all four methods reviewed in this paper on the validation set of the panoptic segmentation challenge. Then, we report the performance of the best method on the test leaderboard and demonstrate a state-of-the-art result by comparing it with other end-to-end neural network solutions.

4.1. Dataset

Datasets used in previous clustering papers have some common problems. For example, the definition of the cluster is ambiguous, trees and walls are also clustered [6]; the metric is limited such as only consider objects within 15 meters [15]. In this paper, we propose to use the newly released panoptic segmentation challenge on SemanticKITTI as a benchmark. This dataset overcomes all previous problems. The definition of the object is very clear. If the same semantic segmentation model is used to provide labels for those non-object points, metrics designed for panoptic segmentation can exactly be used to evaluate clustering performance.

SemanticKITTI [3] is a point cloud dataset about outdoor autonomous driving. It provides the benchmark of three tasks, point cloud semantic segmentation, point cloud panoptic segmentation [4], and semantic scene completion [2]. The dataset contains a training split with ten LiDAR sequences, a validation split with one sequence, and a test split with eleven sequences. All labels on the test split are unavailable. Users need to submit the prediction results to the leaderboard for final evaluation scores.

Semantic Predictions. The clustering method works as a post-process step only focusing on instance points. All methods should use the same semantic model to make a fair comparison. Intuitively, a semantic model with better performance should give a better performance in terms of the panoptic. Considering this, we choose Cylinder3D [45, 47] as the semantic model to provide semantic segmentation predictions. Cylinder3D is the state-of-the-art semantic model with open-sourced code. In the future, if more powerful semantic models are developed, all panoptic results reported in this paper may become even better.

4.2. Metrics

The metric used in this paper is the same as panoptic segmentation. PQ (panoptic quality) is the major indicator defined to measure both the semantic segmentation for stuff and the instance segmentation for things [19]. An improved $PQ^{\dagger}$ is further defined [30]. More indicators are reported in the leaderboard, please refer to those papers for more details [19, 30]. We give some discussions while comparing with others in Section 4.4. Note, sharing the same semantic segmentation makes it possible to measure cluster algorithms by using those panoptic segmentation indicators. Better metrics to measure the cluster performance may be designed in the future.

4.3. Cluster Performance Comparison

Methods	Settings	PQ
Euclidean Cluster	d_{th}=0.5m	56.9
Supervoxel Cluster	w_c, w_s, w_n= 0.0, 1.0, 0.0	52.8
Supervoxel Cluster	w_c, w_s, w_n= 0.0, 1.0, 0.5	52.7
Depth Cluster	θ= 10^o	55.2
Scan-line Run	th_{run}, th_{merge}= 0.5, 1.0	**57.2**

Table 1: Performance comparison on validation set (sequence 08) of SemanticKITTI panoptic segmentation.

Table 2: The performance comparison on SemanticKITTI test set (sequence 11 to sequence 21). The red indicates the best, and the blue indicates the runner-up. sem+box means semantic segmentation with 3D object detection. sem+cluster means semantic segmentation with clustering algorithm.

Methods	Type	Venue	**PQ**	**PQ**†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	$mIoU$
KPConv[39]+PointPillars[24]	sem+box	-	44.5	52.5	54.4	80.0	32.7	38.7	81.5	53.1	65.9	79.0	58.8
RangeNet++[28]+PointPillars[24]	sem+box	-	37.1	45.9	47.0	75.9	20.2	25.2	75.2	49.3	62.8	76.5	52.4
KPConv[39]+PV-RCNN [33]	sem+box	-	50.2	57.5	61.4	80.0	43.2	51.4	80.2	55.9	68.7	79.9	62.8
Panoptic RangeNet[27]	panoptic	iros20	38.0	47.0	48.2	76.5	25.6	31.8	76.8	47.1	60.1	76.2	50.9
Panoster[10]	panoptic	ral21	52.7	59.9	64.1	80.7	49.4	58.5	83.3	55.1	68.2	78.8	59.9
PolarNet_seg[46]	panoptic	cvpr21	54.1	60.7	66.0	81.4	53.3	60.6	87.2	54.8	68.1	77.2	59.5
DS-Net[17]	panoptic	cvpr21	55.9	62.5	66.7	82.3	55.1	62.8	87.2	56.5	69.5	78.7	61.6
Cylinder3D[47, 45]+SLR [43]	sem+cluster	-	56.0	62.6	67.4	82.1	51.8	61.0	84.2	59.1	72.1	80.6	67.9

By using the same semantic result, we compare the performance of all four selected cluster algorithms on the SemanticKITTI validation set in Table 1. For the Euclidean cluster, we set d_{th} (distance threshold) with a moderate threshold 0.5 meter. This makes sense as a large d_{th} will group close objects together, and a small d_{th} will be sensitive to faraway objects with few points. For the supervoxel cluster, we mainly investigate the effect of fusing normal information with distance. However, the result in Table 1 reports a slightly worse PQ when considering the extra normal information. Without normal, supervoxel will degenerate to evenly cut of the 3D space, thus has the worst performance among all four methods. For the depth cluster, we keep the same choice of the angle threshold θ as the original paper. For the scan-line run cluster, the values of two distance thresholds are also kept the same with the paper. The scan-line run cluster achieves the best performance on the validation set in all four methods. We visualize some samples of this clustering algorithm on SemanticKITTI validation set in Fig. 4.

4.4. State-of-the-art Comparison on Leaderboard

From Table 1, we show the Scan-line Run (SLR) [43] is the best clustering algorithm among the four methods evaluated on the SemanticKITTI validation set. Then, we evaluate our pipeline on the SemanticKITTI test set by using SLR as the traditional cluster to combine with Cylinder3D [47, 45]. In Table 2, we report the performance and compare this hybrid solution with others. There are two major types of competitors. One type is panoptic segmentation combined with semantic segmentation and 3D object detection, numbers are cited from [17]. This kind of solution needs two large networks with redundant computational cost and information. The second type is the end-to-end panoptic segmentation model. The proposed hybrid solution outperforms all others on the leaderboard in terms of two major indicators PQ and $PQ^\dagger$. For other metrics reported in the leaderboard, our pipeline achieves comparable performance with the current state-of-the-art method DS-Net [17].

Relationship with Semantic Segmentation. One interesting finding from the Table 2 is the number comparison with DS-Net [17]. Our hybrid solution performs better for PQ^{St}, RQ^{St} and SQ^{St}, but performs worse for PQ^{Th}, RQ^{Th} and SQ^{Th}. The indicators PQ^{St}, RQ^{St}, and SQ^{St} measure the model performance on non-object points. So we can say the reason our hybrid solution can achieve the state-of-the-art performance is because the traditional cluster algorithm almost keeps the same high performance of the semantic segmentation model as well as provides pretty good instance clustering results. This is a unique advantage of the traditional method. For those end-to-end network solutions, involving the extra instance branch beside the semantic segmentation makes the training of the network become a multi-task learning problem. It is unclear how multi-task learning will affect the network performance on semantic segmentation. In Cylinder3D [45, 47], authors show extending the semantic with extra instance branch decreases the mIoU from 65.9 to 63.5 with 2.4 point drop on the validation set. In our case, adding the traditional cluster only decreases the mIoU from 68.9 to 67.9 with one point drop on the test leaderboard. This advantage of the traditional method with the high-quality instance clustering performance helps our proposed pipeline outperform others for major panoptic indicators PQ and $PQ^\dagger$.

4.5. Time Complexity

The inference time is also an important factor. Let's assume there are N points in total, and the number of pix-

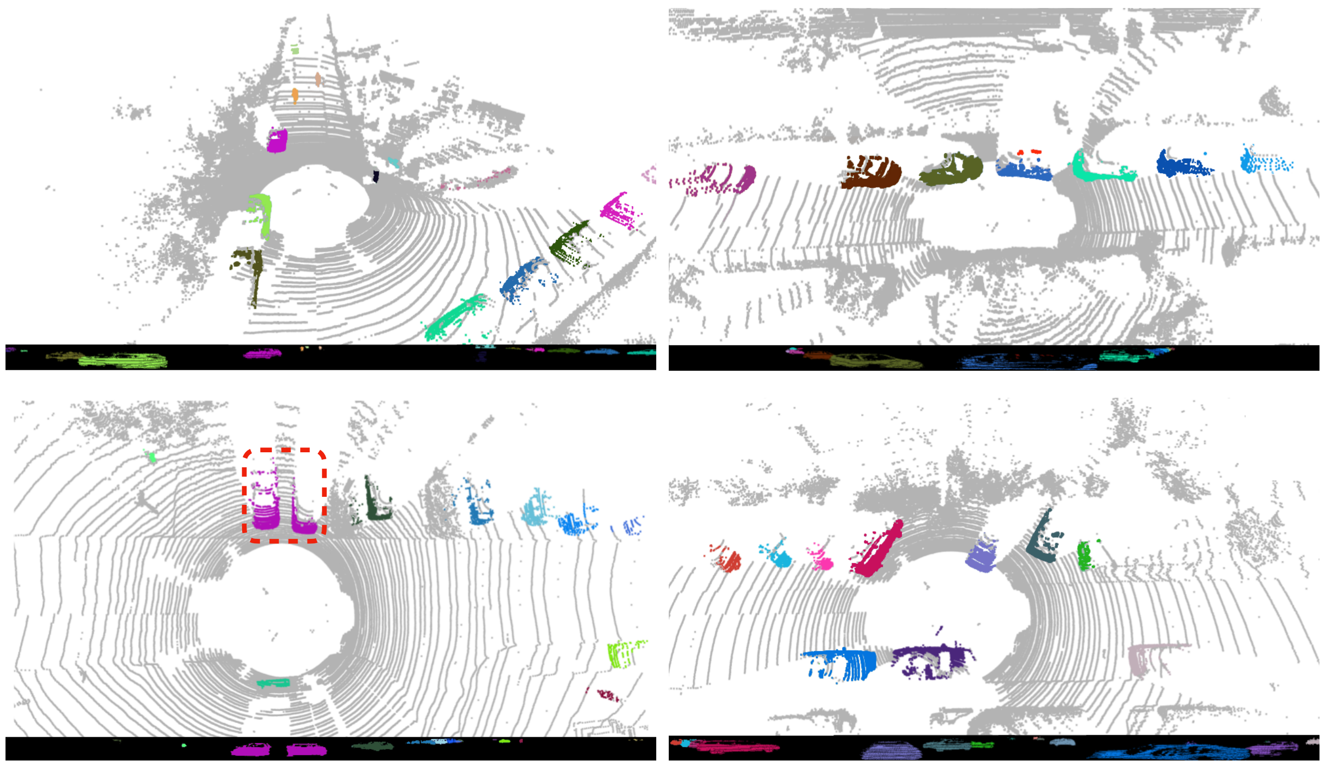

Figure 4: Visualization of the Scan-line Run clustering result on some samples from the SemanticKITTI validation set. One failure mode is two cars stay close to each other as well as park perpendicular to the LiDAR scanner, circled with the red dash box. For each sample pair, the top is the point cloud and the bottom is the range image.

els on the range image is roughly the same as N. Due to the construction of the kd-tree, the Euclidean cluster has $Nlog(N)$ time complexity. We voxelize the space with the cube of size $10cm$. If there are m cubes with points inside them, the time complexity of the Euclidean cluster is $mlog(m)$, and $m \ll N$ for 64 line LiDAR. Both the depth cluster and scan-line run cluster are working on the range image with modified connected-component labeling algorithms. Therefore, those two clustering methods have the linear $O(N)$ complexity. The inference speed of supervoxel is saved by limit the searching region, or seed resolution. In Table 3, we directly report the inference time of the supervoxel implementation in point cloud library (PCL) with 0.5m voxel resolution and 8.0m seed resolution.

Methods	Speed of Single Frame
Euclidean Cluster	16.2ms
Supervoxel Cluster	62.5ms
Depth Cluster	18.1ms
Scan-line Run	29.0ms

Table 3: Inference time of one LiDAR frame. All our implementations run on a single thread of i7-6700K CPU @ 4.00GHz.

We report the average single frame inference time of our implementations in Table 3. Although the inference time is also decided by the coding quality, we present those numbers here to emphasize that traditional cluster algorithms are fast enough to work with the 10Hz LiDAR frame rate. Note, the original depth cluster paper [6] reports 4.7ms to process each LiDAR frame. This time difference with our implementation is partially caused by the existence of many holes on the KITTI range image.

5. Conclusion

This is a technical survey paper about LiDAR point cloud cluster methods and their performance evaluation on the panoptic segmentation benchmark. We want to deliver the message that the traditional clustering algorithm is a useful asset for 3D point cloud tasks. From the view of the LiDAR panoptic segmentation, we propose a new hybrid pipeline with state-of-the-art performance. From the view of the point cloud cluster method, we provide a benchmark to fairly evaluate and compare different clustering algorithms. All the code used in this paper will be released to the public. We believe this work will be useful for both the academic research and engineering designs to better understand the LiDAR point cloud towards solving the real-world problem.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Jürgen Gall, and Cyrill Stachniss. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The semantickitti dataset. *The International Journal of Robotics Research*, page 02783649211006735, 2021.

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[4] Jens Behley, Andres Milioto, and Cyrill Stachniss. A benchmark for lidar-based panoptic segmentation based on kitti. *arXiv preprint arXiv:2003.02371*, 2020.

[5] Yizhak Ben-Shabat, Tamar Avraham, Michael Lindenbaum, and Anath Fischer. Graph based over-segmentation methods for 3d point clouds. *Computer Vision and Image Understanding*, 174:12–23, 2018.

[6] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 163–169. IEEE, 2016.

[7] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019.

[8] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020.

[9] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011.

[10] Stefano Gasperini, Mohammad-Ali Nikouei Mahani, Alvaro Marcos-Ramiro, Nassir Navab, and Federico Tombari. Panoster: End-to-end panoptic segmentation of lidar point clouds. *IEEE Robotics and Automation Letters*, 6(2):3216–3223, 2021.

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[12] Frederik Hasecke, Lukas Hahn, and Anton Kummert. Flic: Fast lidar image clustering. In *ICPRAM*, pages 25–35, 2021.

[13] Lifeng He, Yuyan Chao, and Kenji Suzuki. A run-based two-scan labeling algorithm. *IEEE transactions on image processing*, 17(5):749–756, 2008.

[14] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43, 2017.

[15] David Held, Devin Guillory, Brice Rebsamen, Sebastian Thrun, and Silvio Savarese. A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues. In *Robotics: Science and Systems*, volume 12, 2016.

[16] Noureldin Hendy, Cooper Sloan, Feng Tian, Pengfei Duan, Nick Charchut, Yuesong Xie, Chuang Wang, and James Philbin. Fishing net: Future inference of semantic heatmaps in grids. *arXiv preprint arXiv:2006.09917*, 2020.

[17] Fangzhou Hong, Hui Zhou, Xinge Zhu, Hongsheng Li, and Ziwei Liu. Lidar-based panoptic segmentation via dynamic shifting network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13090–13099, 2021.

[18] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11–seamless operability between c++ 11 and python. *URL: https://github. com/pybind/pybind11*, 2017.

[19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019.

[20] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3d laser data. In *2008 IEEE international conference on robotics and automation*, pages 4043–4048. IEEE, 2008.

[21] Klaas Klasing, Dirk Wollherr, and Martin Buss. Realtime segmentation of range data using continuous nearest neighbors. In *2009 IEEE International Conference on Robotics and Automation*, pages 2431–2436. IEEE, 2009.

[22] Xin Kong, Guangyao Zhai, Baoquan Zhong, and Yong Liu. Pass3d: Precise and accelerated semantic segmentation for 3d point cloud. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3467–3473. IEEE, 2019.

[23] Loic Landrieu and Mohamed Boussaha. Point cloud over-segmentation with graph-structured deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7440–7449, 2019.

[24] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[25] You Li, Clément Le Bihan, Txomin Pourtau, and Thomas Ristorcelli. Insclustering: Instantly clustering lidar range measures for autonomous vehicle. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[26] You Li, Clement Lebihan, Txomin Pourtau, Thomas Ristorcelli, and Javier Ibanez-Guzman. Coarse-to-fine segmenta-

tion on lidar point clouds in spherical coordinate and beyond. *IEEE Transactions on Vehicular Technology*, 2020.

[27] Andres Milioto, Jens Behley, Chris McCool, and Cyrill Stachniss. Lidar panoptic segmentation for autonomous driving. IROS, 2020.

[28] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019.

[29] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2027–2034, 2013.

[30] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019.

[31] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[32] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[33] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[34] Myung-Ok Shin, Gyu-Min Oh, Seong-Woo Kim, and Seung-Woo Seo. Real-time and accurate segmentation of 3-d point clouds based on gaussian process regression. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3363–3377, 2017.

[35] Kshitij Sirohi, Rohit Mohan, Daniel Büscher, Wolfram Burgard, and Abhinav Valada. Efficientlps: Efficient lidar panoptic segmentation. *arXiv preprint arXiv:2102.08009*, 2021.

[36] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017.

[37] Xuebin Sun, Han Ma, Yuxiang Sun, and Ming Liu. A novel point cloud compression algorithm based on clustering. *IEEE Robotics and Automation Letters*, 4(2):2132–2139, 2019.

[38] Xuebin Sun, Yuxiang Sun, Weixun Zuo, Shing Shin Cheng, and Ming Liu. A novel coding scheme for large-scale point cloud sequences based on clustering and registration. *IEEE Transactions on Automation Science and Engineering*, 2021.

[39] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[40] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020.

[41] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Analysis and Applications*, 12(2):117–135, 2009.

[42] Xia Yuan, Yangyukun Mao, and Chunxia Zhao. Unsupervised segmentation of urban 3d point cloud based on lidar-image. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2565–2570. IEEE, 2019.

[43] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073. IEEE, 2017.

[44] Xiangmo Zhao, Pengpeng Sun, Zhigang Xu, Haigen Min, and Hongkai Yu. Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications. *IEEE Sensors Journal*, 20(9):4901–4913, 2020.

[45] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020.

[46] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13194–13203, 2021.

[47] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9939–9948, 2021.