IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 4, 2020, accepted July 6, 2020, date of publication July 17, 2020, date of current version July 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009914

SAnE: Smart Annotation and Evaluation Tools for Point Cloud Data

HASAN ASY'ARI ARIEF[1], (Member, IEEE), MANSUR ARIEF[2], (Graduate Student Member, IEEE), GUILIN ZHANG[2], ZUXIN LIU[2], (Member, IEEE), MANOJ BHAT[2], ULF GEIR INDAHL[1], HÅVARD TVEITE[1], (Member, IEEE), AND DING ZHAO[2], (Member, IEEE)

[1]Faculty of Science and Technology, Norwegian University of Life Sciences, 1432 Ås, Norway
[2]Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Corresponding author: Hasan Asy'ari Arief (hasanari@nmbu.no)

This work was supported in part by gifts from Bosch Corporate Research.

ABSTRACT Addressing the need for high-quality, time efficient, and easy to use annotation tools, we propose SAnE, a semiautomatic annotation tool for labeling point cloud data. The contributions of this paper are threefold: (1) we propose a denoising pointwise segmentation strategy enabling a fast implementation of one-click annotation, (2) we expand the motion model technique with our guided-tracking algorithm, and (3) we provide an interactive, yet robust, open-source point cloud annotation tool, targeting both skilled and crowdsourcing annotators. Using the KITTI dataset, we show that the SAnE speeds up the annotation process by a factor of 4 while achieving Intersection over Union (IoU) agreements of 84%. Furthermore, in experiments using crowdsourcing services, SAnE achieves more than 20% higher IoU accuracy compared to the existing annotation tool and its baseline, while reducing the annotation time by a factor of 3. This result shows the potential of SAnE, for providing fast and accurate annotation labels for large-scale datasets with a significantly reduced price. SAnE is open-sourced at https://github.com/hasanari/sane.

INDEX TERMS Annotation tool, crowdsourcing annotation, frame tracking, point cloud data.

I. INTRODUCTION

The growing popularity of high-frequency point cloud data, scanning real-world driving scenes, fuels up a new research stream on 3D perception systems. This is enriching the perception systems discussion previously centered around image analysis (from cameras) to the realm of point cloud analysis, which includes point cloud classification, segmentation, and object detection [1], [2]. Several large driving scene datasets, containing point cloud data, have recently been published by self-driving tech companies, such as ArgoVerse, Waymo and Lyft [3], highlighting the trend of collecting and using Light Detection and Ranging (LiDAR) point cloud data in the self-driving technologies that are being developed and deployed in the real world.

Developing robust self-driving technologies requires more than just data acquisition. Data annotation, i.e. labeling objects in point cloud scenes, is also necessary to enable the learning process. The annotation process is usually tedious and resource-consuming, and the results can be inaccurate if done manually [4]. Furthermore, as the complexity of annotating 3D point clouds increases, human annotators become more prone to making mistakes. The annotation errors for 3D objects have been found to be significantly higher than for 2D instances. The erroneous labels of the KITTI 3D object detection dataset (such as objects with missing labels or objects having incorrect bounding box locations) [5] is an example of the practical challenge of providing high-quality ground truth annotations.

To tackle these challenges, researchers have proposed both fully automated point cloud annotation techniques (e.g. Frustum Pointnet [6], F-Convnet [7], and AVOD [8]) and semiautomatic annotation tools (e.g. PolygonRNN [9], PolygonRNN++ [10], 3D-bat [11], and Latte [4]). 3D-bat and Latte are specifically proposed for annotating 3D point cloud data. The 3D-bat application proposes a 3D annotation toolbox that is equipped with features that focus on usability and annotation efficiency, but is lacking with respect to automatic functionalities. Latte proposes a tool for 3D point cloud annotation with one-click annotation (based on the

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

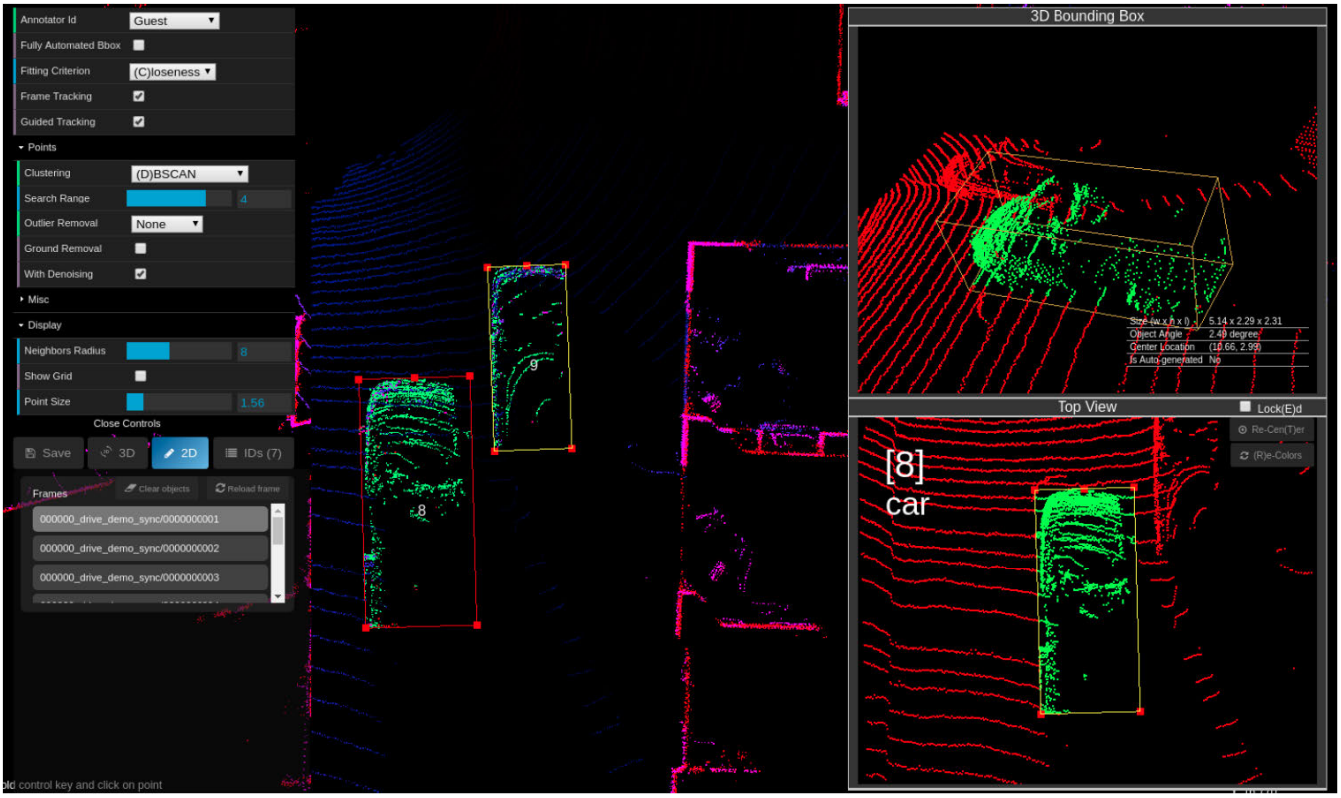


FIGURE 1. The interface of SAnE, a semiautomatic annotation tool based on a one-click annotation scheme empowered by a denoising pointwise segmentation approach and a robust guided-tracking algorithm.

DBSCAN algorithm [12]) and frame tracking (based on the Kalman Filter [13]), reducing the complexity of the annotation task and attaining improved efficiency and an accuracy performance of 87.5% [4]. This seems promising for effective annotation, but for the application to be practical, it has to be easy to use for common users, such as crowdsourcing workers. We have experimented with Latte in such a settings, collecting annotations from workers using Amazon Mechanical Turk, and obtained an annotation accuracy of only 59.32% in terms of Intersection over Union (IoU), a drop of 28.18% in IoU from the performance reported in [4]. This result aligns with findings in [14], suggesting that when an annotation tool is used by crowdsourcing workers, the results tend to be less accurate, and in some cases, not even usable.

Therefore, in this paper, we propose the Smart Annotation and Evaluation (SAnE) tool for cost-effective point cloud annotation, inspired by the Latte interactive tool, implementing a 3D point cloud deep learning model and a guided tracking algorithm to boost performance. SAnE enables both expert annotators and crowdsourcing workers to annotate the point cloud data accurately and efficiently by implementing:

1) **Denoising pointwise segmentation**, a novel nearly noise-free semantic segmentation strategy, enabling a robust one-click annotation technique. In addition, the denoising technique eliminates the need for a workable ground removal algorithm, that is a requirement in Latte [4].
2) **Guided tracking**, based on a motion model that provides baseline tracking through all the frames and refined using heuristic approaches (greedy search and a backtracking algorithm). Only minimal adjustment (if any) is needed for the human annotator to track sequential point cloud scenes.
3) **Improved annotation flow**, enhanced with both AI-based functionalities (one-click annotation, guided-tracking, and fully automated bounding box proposals) and User Interface (UI) based improvements, such as keyboard-only annotations, multi-user environments, user-adjusted parameters, and 3D bounding box estimation.

Our experiments using the KITTI dataset [15] highlight that SAnE can achieve a competitive result compared to Latte under similar experiment settings, an average IoU agreement of 84.27% and a recall value of 86.42%. Furthermore, when tested in a crowdsourcing setting, SAnE achieved much higher performance (79.36% and 80.64%) compared to Latte (59.32% and 58.86%) in terms of IoU agreement and recall value, respectively. Besides, in this crowdsourcing setting, SAnE attains 2.92 times speedup compared to its baseline, implying a significant potential for reducing annotation costs in an already low-cost service.

The rest of our work is organized as follows. In Section II, we review point cloud annotation algorithms. In Section III, we describe the key machinery that we have either

designed or adapted from earlier work when developing the SAnE. Experiment results, ablation studies, and discussions are provided in Section IV. Finally, conclusions are provided in Section V.

II. RELATED WORK

A. POINT CLOUD SEMANTIC SEGMENTATION AND OBJECT DETECTION

LiDAR based 3D object detection is essential for autonomous driving, because point clouds collected from LiDAR contain rich 3D information, including location, dimension, and orientation. However, compared to 2D images, 3D point clouds are irregular and unordered. It is therefore hard to leverage the traditional image analysis techniques to perform general recognition tasks on point clouds, such as semantic segmentation [2], [16]. In early works, people manually transformed irregular point clouds into regular 3D voxel grids [17]. Such a transformation successfully represents irregular 3D data but is constrained by the data density and the shape of the objects. More recent works operate directly on 3D point clouds. PointNet [18] directly consumes point cloud data and provides a unified approach to general 3D recognition tasks. PointCNN [19] is a generalized CNN framework that includes feature learning from point clouds to achieve point cloud segmentation. We leverage and improve this method using our proposed denoising pointwise segmentation method, boosting the accuracy and efficiency of the SAnE.

Some works achieve end-to-end object detection on point clouds. Many works have tried to leverage mature 2D detectors for generating 2D proposals and perform bounding box regression in 3D space, such as the Frustum Pointnet [6]. Inspired by 2D region proposal networks like F-Convnet [7] and AVOD [8], it proposes a novel architecture that contains a feature extractor and subnetworks for 3D proposal generation and regression.

B. ANNOTATION TOOLS FOR POINT CLOUDS

With the development of LiDAR based detection methods and the increasing demand for 3D datasets, some works have contributed annotation tools that aim at improving the efficiency of creating useful datasets. PolygonRNN [9] and PolygonRNN++ [10] propose a semiautomatic approach to polygon region prediction, speeding up the image annotation process by a factor of 7. Annotation tools on 2D image data have been very successful, and 3D annotation tools are also improving. 3D-Bat [11] and Latte [4] provide well-developed point cloud annotation tools integrated with semiautomatic functionalities, deployed as web-based applications. Latte realizes one-click annotation, significantly reducing complex annotation work into a simple click operation. It also proposes frame-to-frame object tracking that further boosts the annotation efficiency for sequential data frames. However, Latte is still using 2D detectors (MaskRCNN [20]) on images, combined with points projected from 3D point clouds, for label prediction. This approach is constrained by camera view and image quality, and tends to mislabel closely located objects. To address this problem, we propose a denoising pointwise segmentation to improve the prediction accuracy and simplify one-click annotation.

An important feature of an annotation tool is that it is easy to use, even with minimum instruction. This is essential for non-experts, like crowdsourcing workers. At the same time, the tool must deliver high-quality results. Balancing between quality and cost is always problematic, especially in crowdsourcing environments [14], [21], [22]. Furthermore, it is important to have high-quality annotation labels to improve the accuracy of heuristic-and-learning based algorithms. To address these problems, we propose SAnE, an easy to use semiautomatic annotation tool, capable of delivering fast and accurate annotation labels, even in a crowdsourcing setting.

III. SAnE ANNOTATION

Creating an open-source, yet high-quality, AI-assisted point cloud annotation tool has been our goal. In this section, we emphasize three key contributions of our work, namely: (1) The denoising pointwise segmentation strategy, enabling accurate one-click annotation, (2) The guided-tracking algorithm, easing the frame-to-frame annotation process, and (3) An interactive yet robust point cloud annotation tool that simplifies the creation of high-quality 3D annotation datasets.

A. DENOISING POINTWISE SEGMENTATION

Deep learning based pointwise segmentation techniques, such as PointNet [18], PointNet++ [23], and PointCNN [19], are based on the cross-entropy loss function and the back-propagation algorithm in their kernel optimization processes. These techniques, even though they tend to provide high accuracy prediction results [23], are prone to provide a noisy segmentation near the object boundaries, see Fig. 2a and b. This is because the particular loss function penalizes all wrong predictions, formulated as

$$L = -\sum_{i}^{C} t_i \log(s_i), \tag{1}$$

where C denotes the number of classes, s_i denotes the confidence-score class i, while t_i denotes the ground truth for class i.

A noisy pointwise segmentation complicates the annotation process, such as the Latte one-click-annotation technique (see [4]). This technique uses the Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [12] to isolate point-clusters and generate a bounding box for the selected cluster. A noisy cluster may result in a wrong bounding box shape and an inaccurate prediction of the box direction, see Fig. 2c and d. The proposed denoising technique aims to provide a noise-free segmentation, enabling one-click annotation. In addition, the technique also does

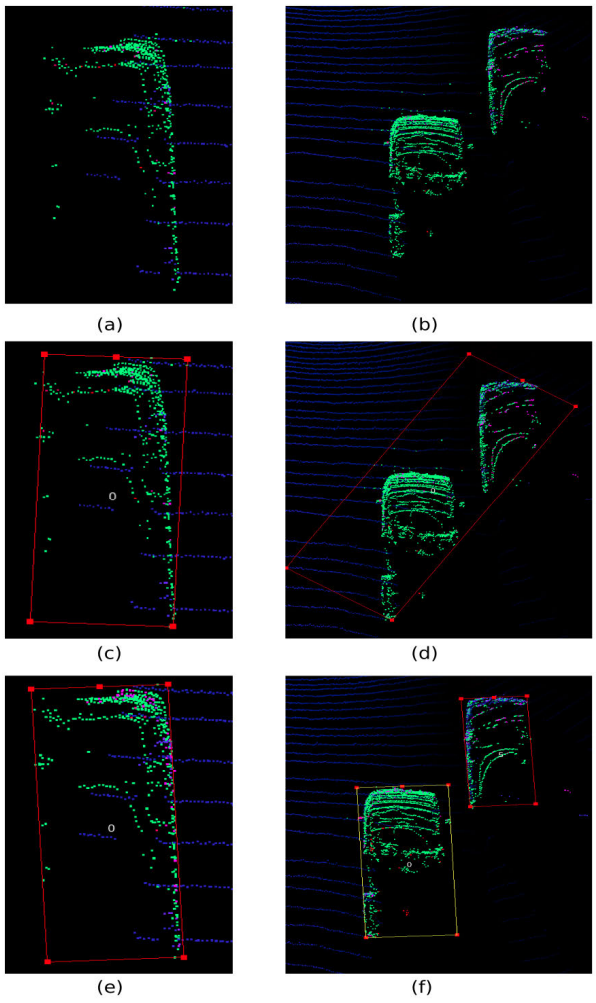


FIGURE 2. The impact of the denoising pointwise segmentation on estimating bounding box proposals using one-click annotation technique: (a-b) noisy boundaries pointwise segmentation, (c-d) bounding box estimation using a standard one-click annotation technique on noisy point cloud segmentation, and (e-f) bounding box estimation using the denoising pointwise segmentation technique.

ground removal that is required for the one-click annotation process.

The main idea of the denoising technique is to force the deep learning model to avoid wrong predictions near object boundaries during kernel optimization (training process) by increased penalization. As shown in Fig. 2e and f, the same one-click annotation technique provides better bounding box proposals for the noise-free point cloud segmentation data. The denoising technique is implemented as a set of penalty values to the prediction results during the loss calculation. Therefore, the technique can be implemented for both the cross-entropy loss function and other loss functions [2], [24].

Given a set of weighted penalties W_p in the point cloud data (P), the denoising penalties are described in Alg. 1. For all objects in a frame, the denoising-weight-penalty calculates all point indices inside the bounding box (Line 3), and recalculates all point indices inside the enlarged ($+nO$) bounding

Algorithm 1 Denoising Weight Penalty

1: W_p, nO, nW, w, zO ▷ Weighted penalties, noise offset, noise weight, normal weight, and distance offset to the ground.
2: **for** *obj* **in** *allObjects* **do**
3: $I_{in} \leftarrow obj.pointIndicesInsideBox()$ ▷ pointIndicesInsideBox() gathers indices of points inside the obj.
4: $obj.dimensions \leftarrow obj.dimensions + nO$ ▷ obj.dimensions contains the dimensions of obj, in terms of width, length, and height.
5: $I_{out} \leftarrow obj.pointIndicesInsideBox()$
6: $W_p[I_{out}] \leftarrow nW$
7: $zMin \leftarrow min(P[I_{in}, Z_{AXIS}]) + zO$
8: $W_p[I_{in} and (W_p[:, Z_{AXIS}] > zMin)] \leftarrow w$
9: **end for**

box (Line 5). Lines 6-8 assigns the noise penalty (nW) for all boundary locations and ground object areas, forcing the loss function to give higher penalties around those areas.

B. GUIDED TRACKING ALGORITHM

Annotating sequential frames of point cloud data can be time-consuming, but it can be speeded up using a frame-to-frame tracking algorithm. For example, the Kalman filtering approach [13] is adopted by Latte [4] to track the bounding box center of an object, and provides a speed-up by a factor of 4.74 compared to manually creating bounding boxes for each new frame. A tracking algorithm does not only speed-up the annotation process but also gives better annotation agreement and accuracy of the tracked bounding boxes [4].

We extend the motion model technique by using the guided-tracking algorithm. The objective of this algorithm is to reduce the effort to refine and/or reannotate the tracked objects. The idea is that some initial bounding box location can be regressed to fit the closest point cloud cluster. The hypothesis is that each cluster belongs to a different object, therefore regressing the bounding box to fit the closest cluster will help refine the bounding box location given by the motion model.

The guided-tracking algorithm comes with three modules, namely: (1) A greedy search, regressing the bounding boxes to their closest corresponding clusters, (2) Backtracking, preventing overlapping between multiple bounding boxes, and (3) Tracking refinement, optimizing the final bounding box location based on the closest point cluster.

1) GREEDY SEARCH

The greedy search algorithm, presented in Alg. 2, works by moving the predicted bounding box around its initial location. It uses the bounding box center location and the point cloud data, denoted as b and P, respectively. A dictionary (s), containing bounding box movements, is also used. For each new location of b, the number of points inside the bounding box is

counted (*numPoints*) and the distances between points inside the box and their closest edges are calculated and summarized (*avgDist*). The location with maximum *numPoints* and minimum *avgDist* is assumed to be the best possible location for the current iteration. The search is implemented using a recursive function, and an iteration that doesn't change the value of *numPoints* ends the search process.

Algorithm 2 Greedy Search Algorithm

procedure GreedySearch(s, b, P)
  $b_u \leftarrow b$ ▷ b contains bounding box information. $b.c$ denotes the bounding box center location along the x and y axis.
  $b.c.x \leftarrow b.c.x - s_p$
  $b.c.y \leftarrow b.c.y - s_p$
  $stride \leftarrow s_p * 2/s_{num}$ ▷ s_{num} denotes the number of strides.
  for $x = 0; x \leq s_{num}; x{+}{+}$ **do**
    for $y = 0; y \leq s_{num}; y{+}{+}$ **do**
      $x_t \leftarrow b_u.c.x \leftarrow b.c.x + x * stride$
      $y_t \leftarrow b_u.c.y \leftarrow b.c.y + y * stride$
      $Ps_{in} \leftarrow b_u.contains(P)$
      $s_{dict}[x_t, y_t][0] \leftarrow (len(Ps_{in}))$
      $s_{dict}[x_t, y_t][1] \leftarrow (b_u)$
      $s_{dict}[x_t, y_t][2] \leftarrow Dist(Ps_{in}, b_u.edges)$
      $s_{dict}[x_t, y_t][2] \leftarrow Avg(s_{dict}[x_t, y_t][2])$
    end for
  end for
  $maxIndices \leftarrow argMaxes(s_{dict}[\ldots][0])$ ▷ argMaxes() gathers the indices of the maximum values from the dictionary.
  if $len(maxIndices) > 1$ **then**
    $minIdx \leftarrow argMin(s_{dict}[maxIndices][2])$ ▷ argMin() gathers the index of the minimum value from the dictionary.
    $numPoints, b_{now} \leftarrow s_{dict}[minIdx][0, 1]$
  else
    $numPoints, b_{now} \leftarrow s_{dict}[maxIndices][0, 1]$
  end if
  if $numPoints > s_{numPoints}$ **then** ▷ $s_{numPoints}$ is a global variable, denoting the highest *numPoints* from previous iterations.
    $s_{numPoints} \leftarrow numPoints$
    $s_{dict} \leftarrow \{\}$
    Return GreedySearch(s_{dict}, b_{now}, P)
  end if
  Return s, b, P ▷ $b.c[x, y]$ is the optimal bounding box location.
end procedure

Alg. 2 relies on the padding size (s_p) to move the bounding box to the best possible location with the highest *numPoints*. A higher s_p means a higher possibility for the bounding box to overlap with other bounding boxes, see Fig. 3a. Therefore, a backtracking algorithm is included to alleviate this problem.

2) BACKTRACKING

The backtracking algorithm works by re-tracking (move-back) the overlapped bounding box locations (*Bs*) to the best possible locations where the boxes do not overlap anymore. The first step is that for each bounding box, the distance between the initial and the updated bounding box location from Alg. 2 is calculated. Then, overlapping boxes are re-tracked based on those distances. The overlapping boxes with the longest distances are moved until the particular boxes are not overlapping anymore, see Alg. 3 for a more complete explanation.

Algorithm 3 Backtracking Algorithm

procedure BackTracking(*Bs*)
  $isOverlapExist \leftarrow true$
  while $isOverlapExist$ **do**
    $isOverlapExist \leftarrow false$
    for $i = 0; i < len(Bs); i{+}{+}$ **do**
      for $j = 0; j < len(Bs); j{+}{+}$ **do**
        $idx \leftarrow -1$
        if $i \neq j$ **then**
          while $idx \neq 0 \wedge Bs[i].Overlap(Bs[j])$ ▷ **do**
            Overlap() check if box[i] is overlapping with box[j].
            $isOverlapExist \leftarrow true$
            $iDist \leftarrow Bs[i].centerDist()$ ▷ centerDist() get the current distance of box[i] from its current location to its initial location before the backtracking occurred.
            $jDist \leftarrow Bs[j].centerDist()$
            if $iDist \geq jDist$ **then**
              $idx \leftarrow Bs[i].updateIdx()$ ▷ updateIdx() update (decrements) the tracking index of Bs[i], then return the updated tracking index. Tracking index for each box is initialized as the number of movements for each bounding box.
            else
              $idx \leftarrow Bs[j].updateIdx()$
            end if
          end while
        end if
      end for
    end for
  end while
end procedure

The backtracking algorithm can separate overlapping objects effectively, see Fig. 3b. However, the proposal for each updated bounding box location might not be optimal. This is because the algorithm does not optimize the *numPoints* and *avgDist* of the moved boxes. Therefore, the

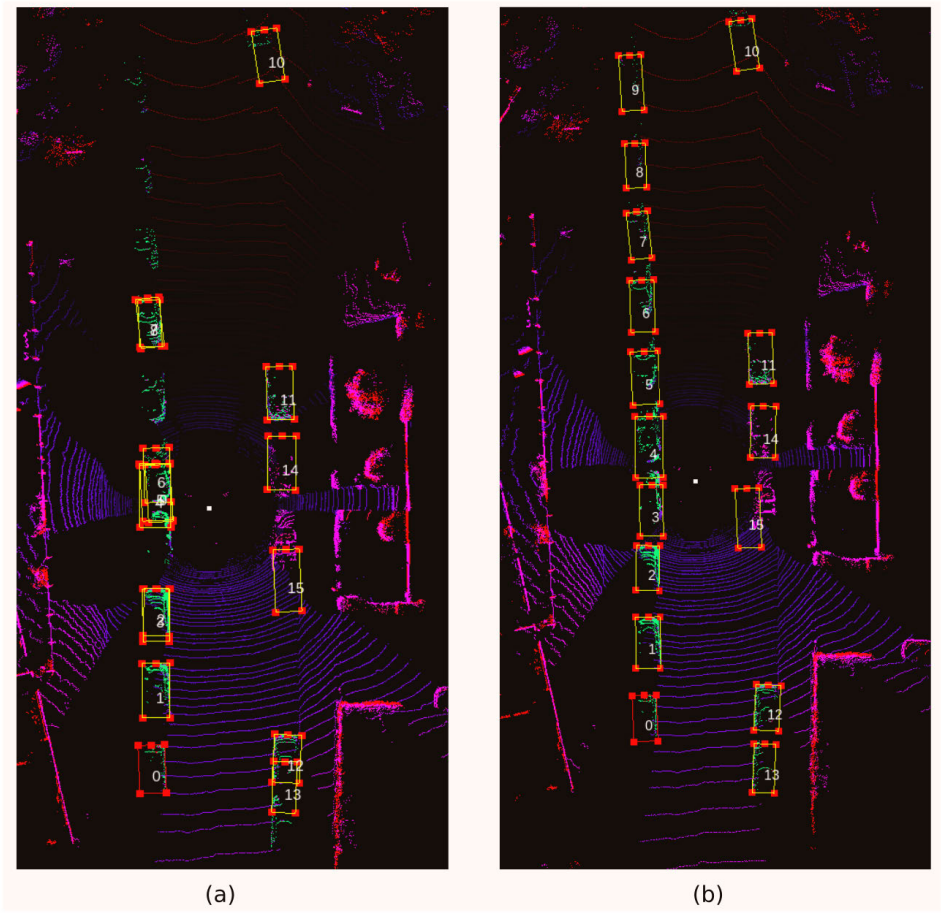


FIGURE 3. The backtracking algorithm for refining the greedy search overlapping problem: (a) before and (b) after the backtracking algorithm.

tracking refinement step is required to optimize the bounding box locations while preventing overlap.

3) TRACKING REFINEMENT

The tracking refinement is a reimplementation of Alg. 2 with much smaller s_p. The intuition is that after the first greedy-search and backtracking processes, the proposed bounding box locations are already close to the optimal solutions. Therefore, only a small change is required to find the best-fitted location.

C. AI-ASSISTED ANNOTATION TOOL

Based on an effective denoising technique and the robust frame-to-frame tracking algorithm, we offer an open-source semiautomatic annotation tool for 3D point cloud data. Several easy to use, yet powerful features are embedded, such as one-click annotation, frame-to-frame tracking, and several other non-AI functionalities.

1) ONE-CLICK ANNOTATION

Our one-click annotation technique is adopted from Latte's implementation [4]. The main improvement is that denoising is used to replace the ground removal algorithm by inducing enhanced penalties around the ground areas. Additionally, a region growing step based on a Nearest Neighbor (NN) search is used as a replacement for Latte's DBSCAN implementation. This is because this DBSCAN implementation is computationally slow (each click requires around 5-10 seconds to process). We hypothesize that the segmented point clusters are separated from each other by some distance, therefore, using an NN search with a predefined search radius gives similar (or better) region proposals compared to the DBSCAN implementation, and works faster for estimating point clusters.

2) FRAME TO FRAME TRACKING.

The frame to frame tracking implemented in the annotation tool follows the description from Subsection III-B including both the motion model technique and guided tracking. The annotators can choose which tracking algorithm they want to use.

Complementing the AI functionalities, SAnE is also equipped with several useful features, including side-view refinement, height estimation, keyboard-only annotation,

object recoloring, and more. The side-view refinement is used to simplify the bounding box refinement in locating, isolating, and magnifying the selected object. The height estimation is used to estimate object height based on the maximum and minimum point inside the bounding box, normalized with the RANSAC algorithm [25]. Moreover, keyboard-only annotation maximizes the use of predefined hotkeys, while object recoloring is used for contrasting the color of points inside and outside a selected bounding box.

IV. EXPERIMENT RESULTS AND FINDINGS

A. EXPERIMENTAL SETUP

We have evaluated our approach on the KITTI tracking dataset [15], and used the training data with their labels for our experiments. The dataset contains 20 scenes and 8 object categories, including car, van, truck, pedestrian, cyclist, sitter, tram, and miscellaneous. 3D Velodyne point cloud data with their colored images along with GPS/IMU data and 3D object tracklet labels are included in the dataset. Due to the limitations of the KITTI labels, i.e inaccurate bounding boxes [4], [5] and only objects in front of the ego vehicle being annotated [15], we also used an expert annotator to provide high-quality Ground Truth labeling (GT). For the rest of this paper, we treat this labeling (GT) as the actual ground truth. It should be noted that the mean IoU agreement between GT and KITTI labels is 72.77%.

For the experiment, we selected eight scenes and used the first 10 frames per scene to do the annotations. We then conducted the experiments by asking four of our skilled workers to annotate objects in those scenes by using the full features of SAnE. We also asked these annotators to annotate the dataset by using a baseline annotation tool, i.e by drawing bounding boxes in the point cloud frame without using any of the proposed SAnE features.

For the crowdsourcing experiments, we used the Amazon Mechanical Turk crowdsourcing services. We used around 70 crowdsourcing workers and each worker annotated (on average) 1.8 scenes. We asked the workers to annotate the point cloud scenes by using the full features of SAnE, the SAnE without any of the proposed features enabled (baseline), and the full features of Latte. We obtained a copy of the version of the Latte annotation tool, that is available in their Github repo, and modified it to suit a multi-user experiment setting. For the ablation study, the crowdsourcing workers annotated the point cloud data by using the SAnE with only one of the proposed features enabled at a time, see Table 2.

In our annotation tool, we used PointCNN [19] as the pointwise segmentation architecture. It was trained by using the density adaptive sampling method [5] and weighted using the proposed denoising technique. The DBSCAN algorithm and region growing method were used for point level clustering, and the proposed guided tracking algorithm with the motion model was implemented for simplifying the frame to frame tracking processes.

Evaluation metrics: Intersection over Union (IoU) for Bird's Eye View (BEV) was used for quantifying bounding box accuracies and we report the IoU averaged over instances annotated by all workers within the same task. The IoU agreement can be seen as the proportional overlap between the intersection areas of the bounding box proposal and its corresponding ground truth data, and the combination (union) of those areas. It should be noted that an object with IoU=0.0 is ignored when calculating the average IoU score. To consider the missing objects, we used the recall values over all existing instances from the GT labels as an additional evaluation metric. Similar to Latte [4], we used a 50% IoU threshold value between an object and a ground truth box to considered it as a true positive. The annotation time per object (in seconds) is also calculated as a surrogate efficiency measure to demonstrate the speed-up of the annotation process for each task, when comparing the baseline annotation process for the skilled annotators (HQ) and the crowdsourcing workers (CS).

B. EXPERIMENTAL RESULTS

We present the efficiency and accuracy metrics of the annotations obtained by using SAnE and Latte in Table 1. In Table 2, we show the result of our ablation study, comparing the metrics obtained for each proposed feature. Finally, in Table 3 we show the per-scene accuracy and efficiency for each selected point cloud scene in the experiment annotated using different settings.

TABLE 1. The accuracy and efficiency of SAnE and Latte annotation tools.

	Time(s)	Mean IoU(%)	STD IoU(%)	Recall(%)
KITTI	-	72.77	5.67	35.10
SAnE Baseline[1]	29.00	80.50	**9.08**	89.14
SAnE Full-Features[1]	**9.48**	**84.27**	9.44	86.42
SAnE Baseline[2]	42.10	62.02	11.00	67.08
SAnE Full-Features[2]	**14.44**	**79.36**	**9.56**	80.65
Latte Full-Features[2]	32.86	59.32	11.59	58.86

[1]skilled; [2]crowdsourcing workers

TABLE 2. The ablation study of SAnE (including the full features of Latte for comparison) annotated by crowdsourcing workers.

	Time(s)	Mean IoU(%)	STD IoU(%)	Recall(%)
Latte	32.86	59.32	11.59	58.86
Baseline	42.10	62.02	11.00	67.08
One-click annotation[1]	19.64	73.85	10.79	80.68
Motion model	13.43	74.59	10.22	**84.51**
Guided tracking	**11.29**	78.56	**9.11**	82.22
Full-features	14.44	**79.36**	9.56	80.65

[1]using denoising pointwise segmentation

Findings: From Table 1, we first observe that the Latte full-feature efficiency and accuracy performance (in terms of annotation time and IoU) is significantly worse in our experiment (32.86s and 59.32%), compared to what has been reported in [4] (9.51s and 85.5%). One plausible explantation for this is that, in our experiment, we employed crowdsourcing workers who might not have been as familiar with point cloud annotation tools as those in Latte's original experiment. Low efficiency and effectiveness is a common phenomenon reported in the crowdsourcing literature [14]. In our

TABLE 3. The breakdown of accuracy and efficiency of SAnE and Latte annotated by crowdsourcing workers for each point cloud scene.

KITTI	Mean IoU(%)				Time(s)		
SceneID [26]	KITTI	Baseline[1]	Full-Features[1]	Latte	Baseline[1]	Full-Features[1]	Latte
0000	77.12	63.10	**85.01**	67.17	**15.48**	18.95	19.76
0004	67.66	66.17	**79.91**	54.19	35.92	**12.56**	37.65
0006	68.41	53.61	**70.81**	47.84	47.59	**10.76**	32.82
0007	77.35	61.60	**80.34**	61.98	43.58	**19.03**	29.15
0009	75.96	59.07	**85.55**	60.61	60.00	**11.07**	40.21
0011	72.25	53.67	**76.83**	65.82	43.98	**8.57**	25.43
0019	73.48	72.80	**77.40**	70.04	36.86	**19.43**	21.68
0020	69.92	65.21	**78.53**	49.71	49.07	**11.91**	52.95

[1] using SAnE annotation tool.

crowdsourcing experiments, SAnE full-features (14.44s and 79.36%) outperform Latte full-features in both efficiency and accuracy in a quite significant way, while SAnE baseline underperforms in terms of efficiency (42.1s) but is on par in terms of accuracy (62.02%). When SAnE is used by skilled workers, both efficiency and accuracy improve, achieving 9.48s and 84.27% for SAnE full-features.

This significant improvement is due to the improved one-click annotation, object tracking, and annotation flow implemented in SAnE, as can be seen from Table 2. Latte one-click annotation depends on the effectiveness of the ground removal for filtering the point cloud, which can be severely affected by scene structure and complexity (mean IoU 59.32% and STD 11.59%). SAnE denoising pointwise segmentation alleviates this dependence, which results in a generally tighter bounding box, and thus better accuracy performance (mean IoU 73.85% and STD 10.79%). Also, SAnE's guided tracking improves on Latte's Kalman filter (which uses a linear approximation for the motion model), reducing the need for correcting the location of the object bounding box in the subsequent frames, slightly improving both annotation accuracy and efficiency (about 4% mean IoU and 1% STD improvement). Finally, all of this, together with the improved annotation flow and interface, allows SAnE to replace several tedious processes in Latte. These processes include real-time inference for image classification as well as image cropping and bounding box prediction, and an unoptimized DBSCAN algorithm. We replaced them with leaner processes consisting of efficient denoising pointwise segmentation and an optimized DBSCAN on the filtered point cloud. By operating solely on point cloud data, SAnE can quickly perform these processes and show the points belonging to an object in a sidebar to guide the annotator almost instantly, easing the overall annotation flow.

As can be seen in Table 1, relying solely on point cloud data to generate bounding box annotations leads to some problems, especially when using IoU as the accuracy metric. The IoU accuracies for the baseline and the full-features of SAnE, annotated by our skilled workers, are only 80.50% and 84.27%, respectively. And the accuracy drops to 79.36% when the annotation is performed by crowdsourcing workers. The IoU improvement of 17.34% is good and can be considered significant but the IoU accuracies can be considered low for a semiautomatic technique. This is because the IoU is a very sensitive metric, and perfect overlap (IoU=1.0) between two bounding boxes on the point cloud scene is almost impossible, see Fig. 4a. Moreover, objects in the point cloud scene can be represented by only a few points, see Fig. 4b. It means that the annotator subjective preferences play a crucial role for providing the most fitted bounding box.

C. ABLATION STUDY

In Table 2, observe that **one-click annotation** implemented by using denoising pointwise segmentation offers +11.83% improvement in terms of IoU compared to the baseline. It also reduces the annotation time per object by more than half. Furthermore, our proposed **guided tracking** algorithm shows a noticeable improvement compared to the **motion-model** algorithm, from 13.43s to 11.29s and from 74.59% to 78.56% in terms of annotation time and IoU accuracy, respectively.

Utilizing all the proposed features (**full-features**), SAnE achieves the highest accuracy of 79.36% in terms of IoU. The annotation time (per object) is only 14.44s slower when compared to only using the guided-tracking feature. This is because the one-click annotation (included in the full feature setting) requires more time to generate a bounding box proposal (server-side). However, the proposed bounding boxes are tighter and more correct, compared to the manually drawn bounding boxes, as reflected in the accuracy improvement. It should be noted that similar effects are shown for the full-features of Latte when all the automated server-side features are enabled.

D. SCENE ANALYSIS

In Table 3, the scenes cover different driving environments i.e. parking location, intersection, highway, urban neighborhood, crowded place, and more. From all these different scenes, the full-features of SAnE consistently achieves the highest accuracy, ranging from 70.81% to 85.55%. with a median value of 79.91% in terms of IoU. This is double-digit higher than the Baseline and Latte accuracies. The numbers are also higher than the KITTI label accuracy by more than 5%. It should be noted that the KITTI dataset is widely used for benchmarking the accuracy of state-of-the-art algorithms for object detection and tracking [26].

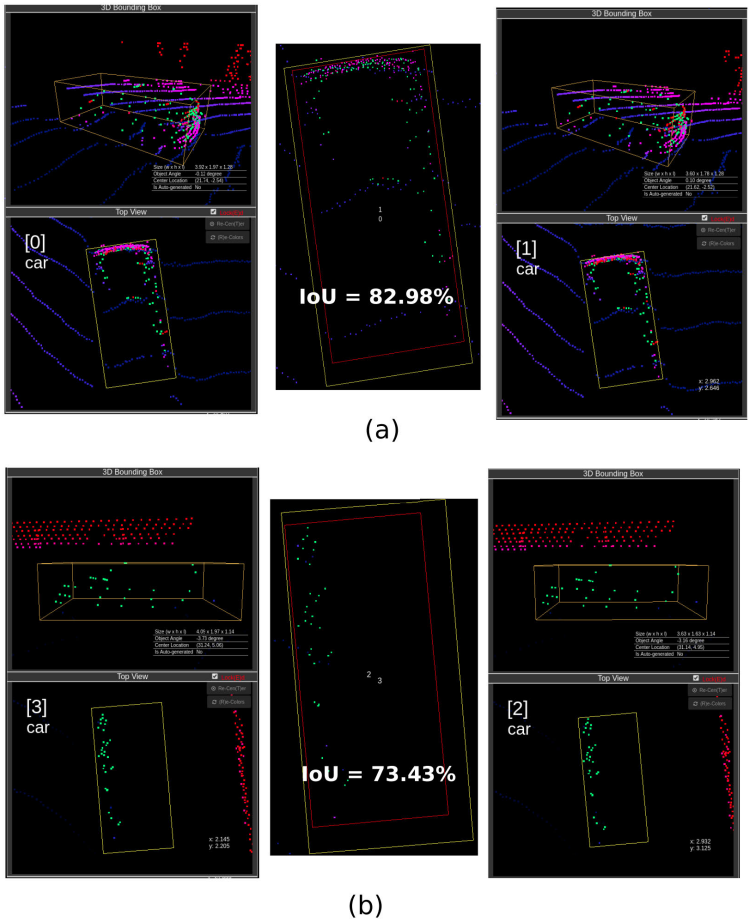


FIGURE 4. Challenges with high accuracy bounding boxes (IoU=1.0) for a point cloud scene.

By looking at the annotation time per object per scene, the full-features of SAnE dominates the results by providing the fastest annotation times for almost all the scenes, more than three times faster compared to the Baseline and Latte annotation times. It shows the applicability of the SAnE to provide fast and accurate annotation labels for point cloud data with a significant cost reduction. This is important because by using the SAnE, computer vision researchers, especially the ones focusing on point cloud learning, can acquire accurate large-scale datasets that are tailored to their needs with a much lower price tag, potentially boosting the development of this research area.

In our experiment using 80 frames from eight scenes of the KITTI dataset, the SAnE shows promising results for generating low cost, high quality point cloud annotations. However, larger scale experiments using multiple datasets with various object categories are still needed to show the applicability of SAnE for annotating different types of objects and environments. Moreover, in this work, the per-class accuracies were not reported because the number of non-car objects, like pedestrian and cyclist, are very limited in the data. Therefore, further research with other datasets, like Waymo Open Dataset [3], will be required to get a more balanced report for those categories.

E. LIMITATIONS

Based on a straight forward weighted penalty, the denoising technique is easy to use and offers powerful guidance for bounding box generation, especially using the one-click annotation algorithm [4]. However, as the penalties emphasize the object boundaries, other areas far from the boundaries suffer, with bad prediction results. This is understandable and even desirable for annotation tools, but for fully automated pointwise segmentation, this approach yields a lower prediction accuracy.

In addition to the segmentation accuracy problem, the one-click annotation is also sensitive to the point-density distribution and object shape representations. The L-shape fitting algorithms [27] are good for generating high quality bounding boxes when the annotated objects are in prefect L-shape forms. However, the L-shape is not the only shape of objects appearing in point cloud scenes. I-shape, U-Shape and even dot-shape are other typical shapes, and the fitting algorithm does not really work on all of these shapes. More work, such

as scene completion [28] and point cloud generation [29], is needed to overcome these problems.

V. CONCLUSION

In this work we have introduced SAnE, a robust semiautomatic annotation tool that simplifies the creation of high accuracy bounding box annotations for point cloud data, targeting both skilled and crowdsourcing annotators. We believe that this tool can help computer vision researchers acquire the tailored high-quality datasets needed to conduct their experiments. The main contributions of our research are threefold. Firstly, we have proposed a denoising pointwise segmentation strategy that can provide nearly noise-free point level classification, enabling one-click annotation. Secondly, we have developed a novel guided tracking algorithm, enhancing the motion model tracking by using a combination of greedy search and backtracking, easing the frame-to-frame annotation processes. Finally, we provide an open-source and easy to use annotation tool that combines AI-based functionalities (such as fully automated bounding box proposals, one-click annotation, and frame-to-frame tracking) and UI-based enhancements, i.e side-view refinement, height estimation, keyboard-only annotation, object recoloring, and more.

Experiments were carried out on the KITTI tracking dataset and the results show that the full features of SAnE, when used by skilled workers, can speed up the annotation process by a factor of 4.44 while achieving higher accuracy than the manual annotation process. Our proposal achieves IoU agreements of 84.27% and a recall value of 86.42%. In our crowdsourcing experiment, the full features of SAnE provided +17.34% and +20.04% improvement in terms of IoU compared to the Baseline and Latte annotation tools, respectively. Furthermore, in the same setting, our proposed annotation tool achieved the fastest annotation time, providing almost 3 times faster annotation than its baseline, potentially providing a significant reduction in annotation costs in an already low-cost environment. Further improvement may be achieved by combining the scene completion and point cloud generation algorithms, alleviating the limitations of point cloud data for representing complete object structure, reducing the influence of human annotation subjectivity.

ACKNOWLEDGMENT

The authors would like to thank Dr. J. E. Kim for valuable discussions and suggestions. They would also like to thank all the anonymous reviewers and their constructive notes and reviews, through which the manuscript was enriched and improved.

REFERENCES

[1] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[2] H. A. Arief, G.-H. Strand, H. Tveite, and U. G. Indahl, "Land cover segmentation of airborne LiDAR data using stochastic atrous network," *Remote Sens.*, vol. 10, no. 6, p. 973, Jun. 2018.

[3] S. Abuelsamid. (Jun. 2019). Argo AI and Waymo Release Automated Driving Data Sets. Forbes Magazine. Accessed: Jan. 20, 2020. [Online]. Available: https://www.forbes.com/sites/samabuelsamid/2019/06/19/argo-ai-and-waymo-release-automated-driving-data-sets/

[4] B. Wang, V. Wu, B. Wu, and K. Keutzer, "LATTE: Accelerating LiDAR point cloud annotation via sensor fusion, one-click annotation, and tracking," Apr. 2019, *arXiv:1904.09085*. [Online]. Available: http://arxiv.org/abs/1904.09085

[5] H. A. Arief, M. Arief, M. Bhat, U. G. Indahl, H. Tveite, and D. Zhao, "Density-adaptive sampling for heterogeneous point cloud object segmentation in autonomous vehicle applications," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2019, pp. 26–33.

[6] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[7] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for Amodal 3D object detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1742–1749.

[8] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[9] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler, "Annotating object instances with a polygon-RNN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5230–5238.

[10] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-RNN++," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 859–868.

[11] W. Zimmer, A. Rangesh, and M. Trivedi, "3D BAT: A semi-automatic, Web-based 3D annotation toolbox for full-surround, multi-modal data streams," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1816–1821.

[12] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 1996, pp. 226–231.

[13] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. Chapel Hill, NC, USA: Univ. of North Carolina at Chapel Hill, Nov. 1995.

[14] S. Vittayakorn and J. Hays, "Quality assessment for crowdsourced object annotations," in *Proc. Brit. Mach. Vis. Conf.*, 2011, pp. 1–11.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[16] H. A. Arief, U. G. Indahl, G.-H. Strand, and H. Tveite, "Addressing overfitting on point cloud classification using atrous XCRF," *ISPRS J. Photogramm. Remote Sens.*, vol. 155, pp. 90–101, Sep. 2019.

[17] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[18] 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.

[19] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2018, pp. 820–830.

[20] K. He, G. Gkioxari, P. DollÃİr, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.

[21] D. Mitry, K. Zutis, B. Dhillon, T. Peto, S. Hayat, K.-T. Khaw, J. E. Morgan, W. Moncur, E. Trucco, and P. J. Foster, "The accuracy and reliability of crowdsource annotations of digital retinal images," *Transl. Vis. Sci. Technol.*, vol. 5, no. 5, p. 6, Sep. 2016.

[22] P.-Y. Hsueh, P. Melville, and V. Sindhwani, "Data quality from crowdsourcing: A study of annotation selection criteria," in *Proc. NAACL HLT Workshop Act. Learn. Natural Lang. Process.*, 2009, pp. 27–35.

[23] 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. (NIPS)*, Dec. 2017, pp. 5099–5108.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[27] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 54–59.
[28] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Niessner, "ScanComplete: Large-scale scene completion and semantic segmentation for 3D scans," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4578–4587.
[29] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A Papier-Mâché approach to learning 3D surface generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 216–224.

HASAN ASY'ARI ARIEF (Member, IEEE) was born in Indonesia, in 1989. He received the Ph.D. degree in applied informatics from the Norwegian University of Life Sciences (NMBU), in March 2020. He was a Visiting Researcher at the Safe AI Lab, Carnegie Mellon University (CMU). His current research interests include the implementation of deep learning technique for spatial- and spatiotemporal data for automatic classification, segmentation, object detection, and localization.

MANSUR ARIEF (Graduate Student Member, IEEE) received the master's degree from the University of Michigan, Ann Arbor. He is currently pursuing the Ph.D. degree with Carnegie Mellon University working with Prof. D. Zhao in the CMU Safe AI Lab. His research interests include stochastic modeling and optimization, rare-event simulation, and applications in intelligent transportation and logistics systems.

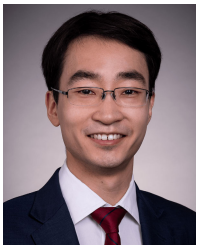
GUILIN ZHANG received the bachelor's degree from The Chinese University of Hong Kong (CUHK). He is currently pursuing the master's degree with the Mechanical Engineering Department, Carnegie Mellon University. His research interests include 3D environment understanding, lidar processing, and self-driving.

ZUXIN LIU (Member, IEEE) received the B.S. degree from Beihang University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree in mechanical engineering with Carnegie Mellon University, Pittsburgh, PA, USA. His research interests are robotics, reinforcement learning, and multi-agent systems.

MANOJ BHAT received the B.Tech. degree from Manipal University, India, in 2018, and the M.S. degree in ME research on robotics from Carnegie Mellon University, USA, in 2020. He was a Research Assistant with the Safe AI Lab, Carnegie Mellon University. His research interests include improving safety and robustness in perception systems, co-operative perception, and simulation for autonomous vehicles.

ULF GEIR INDAHL is currently an Associate Professor of statistics with RealTek/NMBU. He is interested in a broad range of applications as well as the development of new methodology ranging from classical multivariate data analysis to artificial neural nets and deep learning.

HÅVARD TVEITE (Member, IEEE) is currently an Associate Professor of geographical information science with RealTek/NMBU, where he specialises in spatio-temporal geographical data modeling and analysis.



DING ZHAO (Member, IEEE) received the Ph.D. degree from the University of Michigan, Ann Arbor, in 2016. He is currently an Assistant Professor with the Department of Mechanical Engineering, Carnegie Mellon University. His research interests include safely deploy the AI-enabled robots to the real world by developing reliable, verifiable, explainable, and trustworthy learning methods in the face of the uncertain, dynamic, time-varying, multi-agent, and human-involved environment. His works are at the intersection of statistical machine learning, robotics, and optimal design, with applications on autonomous vehicles, smart manufacturing, intelligent transportation, assistant robots, and cybersecurity.