Journal of Big Data

RESEARCH Open Access

Out-of-distribution- and location-aware PointNets for real-time 3D road user detection without a GPU

Alvari Seppänen[1,2*], Eerik Alamikkotervo[1], Risto Ojala[1], Giacomo Dario[1] and Kari Tammi[1,2]

*Correspondence:
alvari.seppanen@aalto.fi

[1] Aalto University, Espoo, Finland
[2] Helsinki Institute of Physics, Helsinki, Finland

Abstract

3D road user detection is an essential task for autonomous vehicles and mobile robots, and it plays a key role, for instance, in obstacle avoidance and route planning tasks. Existing solutions for detection require expensive GPU units to run in real-time. This paper presents a light algorithm that runs in real-time without a GPU. The algorithm combines a classical point cloud proposal generator approach with a modern deep learning technique to achieve a small computational requirement and comparable accuracy to the state-of-the-art. Typical downsides of this approach, such as many out-of-distribution proposals and loss of location information, are examined, and solutions are proposed. We have evaluated the performance of the method with the KITTI dataset and with our own annotated dataset collected with a compact mobile robot platform equipped with a low-resolution LiDAR (16-channel). Our approach reaches a real-time inference on a standard CPU, unlike other solutions in the literature. Furthermore, we achieve superior speed on a GPU, which indicates that our method has a high degree of parallelism. Our method enables low-cost mobile robots to detect road users in real-time.

Keywords: Perception, Deep learning, Object detection, Limited computational resources

Introduction

Detection of road users is an essential task for autonomous vehicles and mobile robots, and it is crucial, for instance, in obstacle avoidance and route planning tasks. The task focuses solely on detecting road users, i.e., cars, pedestrians, and cyclists. This allows the utilization of class-specific biases; for example, objects are expected to be on the ground surface. The task differs from the more general object detection task, which aims to detect any object. Many object detection methods have been developed for detecting road users from point clouds. However, much of the efforts have been directed toward developing methods that achieve high accuracy, and significantly less effort has been directed toward low-latency approaches. The present study addresses this issue with a novel architecture developed with attention to latency. An algorithm that has low latency and computational cost has a large number of benefits. It allows the better

use of high-frequency (> 20 Hz) sensors as the algorithm is also high-frequency. A light algorithm decreases energy consumption, resulting in a longer operation time for mobile robots. It also removes the requirement for a GPU, enabling lighter weight and less expensive hardware, which is crucial in many applications.

To tackle the issue of latency in neural networks, attention should be directed towards the memory (DRAM) reading, as stated by Horowitz et al. [1]. This paper takes a unique data-dependent approach to 3D road user detection to achieve low latency. Unlike most methods in the literature, our approach saves memory by discarding irrelevant parts of the input point cloud. Specifically, a simple algorithm removes coarse parts of the point clouds, such as the ground plane. Then, more complex algorithms, deep neural networks, are used for making the detections. Therefore, as the amount of data decreases, the complexity of the algorithm increases, and the neural network processes only the filtered relevant data. In practice, object proposals are generated with ground removal and clustering, and then a novel out-of-distribution- and location-aware PointNets make the detections from the proposals. We filter out-of-distribution proposals throughout the pipeline to reduce memory and computational footprint. We are also the first to implement an out-of-distribution detection method on classification and bounding box estimation tasks with point cloud object proposals. This approach achieves superior latency and comparable accuracy in 3D road user detection to the state-of-the-art. Huang et al. [2] studied out-of-distribution detection methods directly on 3D object detection, showing the applicability of such methods.

As stated, the literature has many point cloud object detection methods tested for 3D road user detection. However, most of them require a powerful GPU unit to run in real-time, which some mobile robots do not have due to, e.g., power, cost, weight, or size restrictions. To complement this under-explored area, we develop an algorithm that runs in real-time on a standard **CPU**. We summarize the contributions of this paper as follows.

- A novel architecture for 3D road user detection on point cloud data running in real-time on a CPU.
- To the best of our knowledge, we are the first to implement an out-of-distribution detection method on classification and bounding box estimation tasks with point cloud object proposals.
- A novel proposal voxel location encoder, which improves the accuracy of the models by a significant margin.
- A ground segmentation method, which outperforms competitive methods. We present simple convolutional filters for sampling ground points for the plane fit.
- A study on a 3D road user detection task with models trained *only* with the KITTI high-resolution point cloud data and tested with low-resolution and low-perspective point cloud data.

Related work

This section presents frequent 3D object detection approaches that are well-proven and provide a fair comparison to our approach. The approaches are divided into voxel, graph, projection image, point, and bird's eye view methods. VoxelNet [3] partitions the point

cloud into voxel partitions. Points within a voxel partition are encoded into a vector representation characterizing the shape information. 3D convolution is performed on the voxels to predict bounding boxes and classes. SECOND [4] improved the accuracy and reduced computational cost of VoxelNet by preprocessing the point cloud by dropping voxels that include no points.

Authors of [5–8] use methods based on a range image. A range image is a point cloud projection on a spherical surface. The benefit of using projection is to preserve the information of neighboring measurements. However, the authors claim that some information is lost during the projection. The authors use a 2D convolutional neural network to predict bounding boxes and classes.

One approach is to use a bird's eye view image as input for the detection algorithm [9–12]. PointPillars [9] is a popular method using this approach. It constructs pillar-like features from the point cloud with a neural network and forms a bird's eye view image. Then, a 2D convolutional neural network is applied to this image, making predictions. Bird's eye view is a convenient representation of the point cloud because objects are rarely stacked on top of each other in, for example, outdoor driving scenarios, given an optimal vertical field-of-view of the sensor.

More recently, graph neural networks (GNNs) have been implemented into the point cloud object detection task. Shi et al. [13] were the first ones to do so. In their method, point clouds are preprocessed into a graph representation utilizing a grouping method similar to PointNet++ [14]. Then, a GNN architecture predicts the classes and bounding boxes. At the time of publication, they achieved superior results in several public benchmarks, proving that graph-based pre-processing works well in the object detection task from point clouds.

Point-based methods use raw point clouds as an input [15–17]. One benefit is that no pre-processing is needed, e.g., voxelization, range- or bird's eye view-projection. Typically, this results in a lower computational cost overall. However, preserving the geometrical context is challenging since points are processed individually. Ngiam et al. [18] proposed that targeting computation to certain regions benefits computational cost and generalizability. They generated proposals with the furthest and random point sampling and utilized a neural network architecture for estimating bounding boxes.

Methods

A general schematic of the proposed architecture is presented in Fig. 1. The basic principle is to generate simple, unclassified proposals and utilize classifiers that differentiate between the in-distribution (ID) and out-of-distribution (OOD) proposals. This is implemented using novel energy-based OOD PointNets. The first PointNet predicts the class probability vectors and energy scores, which are used for discarding the first batch of OODs. This is done in the first ID pass-through module. Then, 3D bounding boxes are predicted for the remaining proposals with an alternative PointNet, which also predicts energy scores. Further, the second ID pass-through module filters out the remaining OODs based on the bounding box energy scores. In addition, a novel proposal voxel location encoder (PVLE) is utilized to preserve location information otherwise lost in the proposal normalization process. PVLE attempts to increase the accuracy of the neural networks without adding a significant amount of computation. The increased

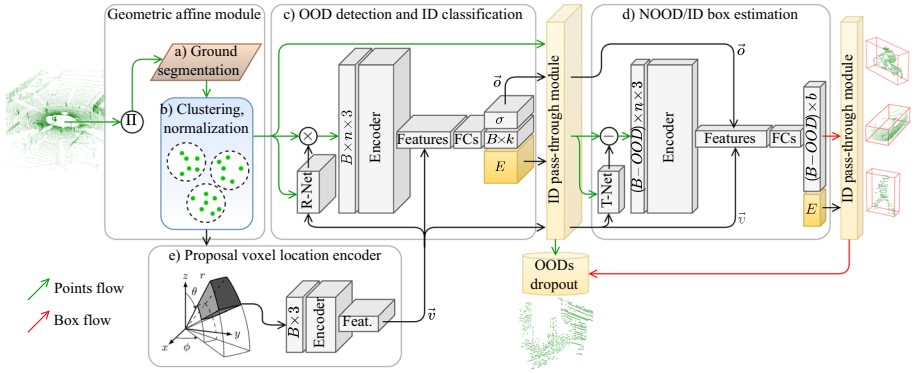


Fig. 1 The proposed architecture. The input point cloud is organized by mapping $\Pi : \mathbb{R}^{n\times 3} \mapsto \mathbb{R}^{s_h \times s_w \times 3}$. Then, the ground segmentation, coupled with a clustering algorithm, generates simple proposals fed into the classifier neural network. Then, the first ID pass-through module discards coarse OOD proposals, which enables low computational requirements for the box estimation network. Similarly, the second ID pass-through module discards boxes that are OOD. In parallel, the PVLE encodes the locations of the proposals and feeds them into the classifier and the box estimator. The final output of the pipeline is 3D bounding boxes and class probabilities for the objects of interest

accuracy will prove that proposal location information has value in the proposal classification and bounding box estimation tasks. The architecture aims to achieve low computational requirements by discarding information hierarchically while preserving helpful information regarding the road user detection task.

Ordered point cloud representation

Point coordinates from a typical LiDAR sensor $\mathbf{p} = (x, y, z)$ are mapped $\Pi : \mathbb{R}^{n\times 3} \mapsto \mathbb{R}^{s_h \times s_w \times 3}$ to spherical coordinates, and finally to image coordinates, as defined by

$$\begin{bmatrix} p_u \\ p_v \end{bmatrix} = \begin{bmatrix} 1/2(1 - \tan^{-1}(yx^{-1})\pi^{-1})s_w \\ (1 - (\sin^{-1}(z||\mathbf{p}||^{-1}) + f_{vup})f_v^{-1})s_h \end{bmatrix} \tag{1}$$

where (s_h, s_w) are the height and width of the desired projection image representation, f_v is the total vertical field-of-view of the sensor, and f_{vup} is the vertical field-of-view spanning upwards from the horizontal origin plane. The resulting list of image coordinates is used to construct a (x, y, z)-channel image, which is the input for the next stage of the architecture.

Ground segmentation

As an additional contribution, we introduce a novel ground segmentation method, which is ultra-fast and more accurate than most of the methods in the literature. Our ground segmentation combines a novel point sampling with the well-proven RANSAC plane fitting method. Figure 2 shows a graphical presentation of the method.

It is essential to segment the ground at the beginning of the pipeline for the following reasons: (a) reduce the number of points to process, (b) remove points that are invalid and not considered in later stages of the pipeline and (c) improve the performance of the clustering algorithm, as some clusters would be fused through ground

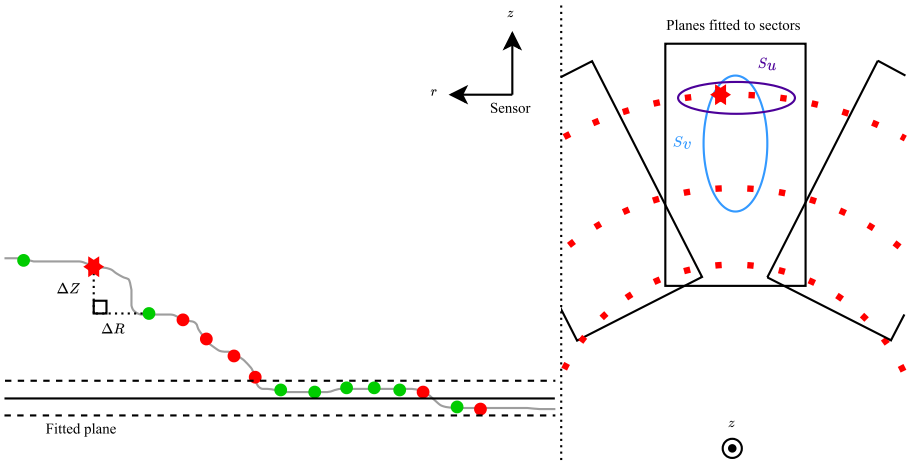


Fig. 2 An illustration of the ground segmentation method. **On the left**, points that passed the filter are indicated by green, and points between the dashed lines are segmented ground points. **On the right**, points surrounded by ellipses are considered by $\mathbf{S}_v$ and $\mathbf{S}_u$ filters when computing the point marked with a star. A plane is fitted to each sector of points. Note that $\mathbf{R} = \sqrt{\mathbf{X}^2 + \mathbf{Y}^2}$, i.e., the distance to the *z*-axis

points. We sample the ordered point cloud for potential ground plane points with two convolutional Sobel [19] inspired filters. These kernel filters are formulated as follows.

$$\mathbf{S}_v = \begin{bmatrix} 2 & 1 \\ -2 & -1 \end{bmatrix} \quad \mathbf{S}_u = \begin{bmatrix} 1 & 2 & -2 & -1 \end{bmatrix}. \tag{2}$$

The filters are discrete differentiation operators that operate on the ordered point cloud tensor. In convolution, $\mathbf{S}_v$ and $\mathbf{S}_u$ yield approximations of vertical and horizontal derivatives on the ordered point cloud, respectively. The first term of $\mathbf{S}_v$ and the second term of $\mathbf{S}_u$ are the centers of the filters. The filters incorporate information of multiple neighboring points with discretized Gaussian function, where points closer to the center have a higher effect on the result of the computation. This is done because simply computing the derivatives with a subtraction between two neighboring points is too noisy in a typical LiDAR measurement. In a typical LiDAR sensor, the horizontal resolution is significantly higher than the vertical resolution. Therefore, the shape of $\mathbf{S}_u$ is 1x4, which means that it does not consider points on neighboring rows, as they are significantly more distant compared to points on neighboring columns (Fig. 2). This way, the filter can capture the approximation of the local derivative more accurately. The convolutions are computed with a range channel $\sqrt{\mathbf{X}^2 + \mathbf{Y}^2} = \mathbf{R} \in \mathbb{R}^{s_h \times s_w}$ and a height channel $\mathbf{Z} \in \mathbb{R}^{s_h \times s_w}$.

$$\mathbf{F}_y = \frac{\Delta \mathbf{Z}}{\Delta \mathbf{R}} = \frac{\mathbf{S}_v * \mathbf{Z}}{\mathbf{S}_v * \mathbf{R}} \quad \mathbf{F}_x = \mathbf{S}_u * \mathbf{R} \tag{3}$$

where $*$ denotes the 2-dimensional convolution operation. Matrices $\mathbf{F}_y$ and $\mathbf{F}_x$ denote an approximation of point-wise normal, as filters produce derivatives. Approximating the point-wise normal with this method requires only a small amount of computation while achieving satisfactory accuracy. We apply a threshold to $\mathbf{F}_y$ and $\mathbf{F}_x$, which gives us a mask of ground point samples:

$$\mathbf{F}_{mask}(r,c) = \begin{cases} 1, \text{ if } |\mathbf{F}_y(r,c)| < y_{th}, \text{ and } |\mathbf{F}_x(r,c)| < x_{th} \\ 0, \text{ otherwise.} \end{cases} \tag{4}$$

Then, the samples $\mathbf{G}_{samples} \in \mathbb{R}^{s_h \times s_w \times 3}$ are computed for the RANSAC algorithm:

$$\mathbf{G}_{samples} = \mathbf{F}_{mask} \odot (\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \tag{5}$$

where $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \in \mathbb{R}^{s_h \times s_w \times 3}$ contain the Cartesian coordinates of each point in the point cloud, and $\odot$ indicates Hadamard product. Only the sampled points are considered in the random sampling of the RANSAC algorithm. Thus, only a handful of iterations are needed, compared to a case where the samples are taken from the entire point cloud, which reduces the computational cost. $RANSAC((\mathbf{X}, \mathbf{Y}, \mathbf{Z})_s, \mathbf{G}_{samples}) = \{a_1, a_2, a_3, a_4\}_s$ gives the parameters of detected planes, as we run this operation on sectors of points (Fig. 2). If the distance between a point and the detected plane in the corresponding sector satisfies a threshold, it is labeled as ground.

$$\mathbf{D} = \sum_{s=0}^{C_s} \frac{(a_1\mathbf{X} + a_2\mathbf{Y} + a_3\mathbf{Z} + a_4)_s}{||\{a_1, a_2, a_3\}_s||} \tag{6}$$

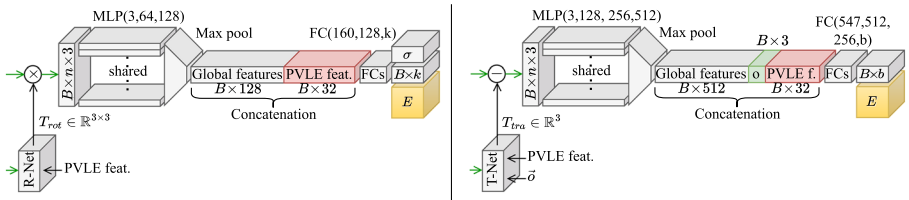
$$\mathbf{G}_{mask}(r,c) = \begin{cases} 1, \text{ if } |\mathbf{D}(r,c)| < p_{th} \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

Angle-based clustering

The clustering algorithm is a key component of the proposal generator. We use the angle-based clustering method [20], mainly because it is fast (full scan from a 64 channel LiDAR in 10 ms with a single core of a 2.2 GHz CPU, reported in [20]) but also because it is sparsity invariant. Moreover, this method is less prone to fusing nearby objects under the same cluster because the clustering is based on *an angle* instead of a distance measurement. This is crucial and affects the performance of our architecture. However, our architecture does not have constraints that would prevent the usage of other standard clustering algorithms such as [21–25]. Still, we found through experimentation that these methods caused our architecture to perform worse. The angle-based clustering algorithm is sparsity invariant because it computes the angle between neighboring points. If this angle satisfies a threshold, the point is assigned under the label of the currently computed cluster. A breath-first search (BFS) is implemented to add points to the current cluster, and a completed BFS indicates the completion of a cluster. The algorithm is fast since it takes advantage of the order of the point cloud, which means that finding the neighboring points is convenient.

Energy-based out-of-distribution detection

The energy-based out-of-distribution detection method performs well in image classification tasks, and it has been well-proven. Its implementation is also convenient because its input is the raw output of a neural network, unlike with methods such as [26–30]. Moreover, it is light computationally. Therefore, we define it as the baseline method on point cloud data.



(a) OOD detection and ID classification.

(b) NOOD/ID bounding box estimation, where $\vec{o}$ denotes a one-hot of the predicted class.

Fig. 3 Proposed classifier PointNet is shown in (**a**) and the bounding box estimator PointNet is shown in (**b**)

The basic idea of an energy-based function is to map each point of the input space to a non-probabilistic scalar called energy $E(\mathbf{x}; f) : \mathbb{R}^L \mapsto \mathbb{R}$ [31]. In our application, the output vectors of the classifier and bounding box estimator networks are mapped into their respective energy scalars that represent the distance to the class distribution. The method presented here is based on Liu et al. [32], where a modified version of the *Helmholtz free energy* from statistical mechanics is used as $E(\mathbf{x}; f)$ [33].

$$E(\mathbf{x}; f) = -T \cdot \log \sum_{i}^{L} e^{f_i(\mathbf{x})/T} \tag{8}$$

where $\mathbf{x}$ denotes the input of a neural network, T temperature scalar, L number of logits, and f a neural network. We utilize Equation (8) in training the classifier and the box estimator networks and during the inference time for separating IDs from OODs.

Network architectures and training objectives

This paper presents a novel implementation of an energy-based OOD detection method for point cloud classifiers and 3D bounding box estimation networks. The proposed neural network architectures (Fig. 3) build on PointNet [34], applying some application-specific modifications to reduce computational cost and increase performance. These modifications include concatenating the proposal voxel position encoder features to leverage the observation angle and distance and simplifying the network in the main encoders and the fully connected layers. The main modification is implementing an energy-based OOD learning objective to mitigate the false positive rate since our proposal generator is simple, resulting in vast OOD instances. Furthermore, we discovered that the respective critical and the upper bound point sets for classifier and box estimation networks differ for the same cluster of points. We exploit this phenomenon by implementing two separate ID pass-through modules for improved OOD detection. The classifier and the bounding box estimator inputs are transformed with R-Net and T-Net networks, respectively. R-Net predicts a rotational matrix $\mathbf{T}_{rot} \in \mathbb{R}^{3\times3}$, which normalizes the rotation angle of the samples to simplify the classification task. Similarly, T-Net predicts a transformation matrix $\mathbf{T}_{tra} \in \mathbb{R}^3$, which normalizes the location of the samples to simplify the bounding box estimation task.

Classifier training objective is to minimize classification cross-entropy and ID/OOD squared hinge loss,

$$\min_{\theta} \ \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{ID}^{train}}(-\log\cdot\sigma(c(\mathbf{x}))) + \lambda\mathcal{L}_{energy} \tag{9}$$

where σ is the softmax output of the classifier c. The energy loss is computed as

$$\begin{aligned}\mathcal{L}_{energy} = \ & \mathbb{E}_{(\mathbf{x}_{ID},y)\sim\mathcal{D}_{ID}^{train}}((E(\mathbf{x}_{ID};c) - m_{ID})^{+})^2 \\ & + \mathbb{E}_{\mathbf{x}_{OOD}\sim\mathcal{D}_{OOD}^{train}}((m_{OOD} - E(\mathbf{x}_{OOD};c))^{+})^2\end{aligned} \tag{10}$$

where $\mathbf{x}_{ID}$ is an ID sample from KITTI training split (points inside a ground truth bounding box), and $\mathbf{x}_{OOD}$ is an OOD sample (clustered points outside the ground truth bounding boxes) also from KITTI training split. Terms m_{ID} and m_{OOD} are the means of ID and OOD energies of the default trained network, respectively. They are used in Equation (10) to push down the energy of IDs, lift the energy of OODs, and expand the energy gap between them. Terms m_{ID} and m_{OOD} are pre-computed and static during the training.

Bounding box estimator training objective combines a bounding box prediction and an energy-based OOD detection objective. Center, size, and heading $(x_c, y_c, z_c, l, w, h, \theta)$ of a bounding box are parameterized to a combination of classes and residuals: $\mathbf{c} \in \mathbb{R}^3, \mathbf{s} \in \mathbb{R}^{N_S}, \mathbf{s}_r \in \mathbb{R}^{N_S\times 3}, \mathbf{h} \in \mathbb{R}^{N_H}, \mathbf{h}_r \in \mathbb{R}^{N_H}$. The goal is to minimize the following function:

$$\begin{aligned}\min_{\theta} \ & \mathbb{E}_{(\mathbf{x},y_b)\sim\mathcal{D}_{ID}^{train}}(\mathcal{L}_{c1-reg} + \mathcal{L}_{c2-reg} + \mathcal{L}_{h-cls} + \mathcal{L}_{h-reg} \\ & + \mathcal{L}_{s-cls} + \mathcal{L}_{s-reg} + \zeta\,\mathcal{L}_{corner}) + \lambda\mathcal{L}_{box-energy}\end{aligned} \tag{11}$$

Our contribution is the definition of the term $\mathcal{L}_{box-energy}$, which penalizes the network depending on the energy output. $\mathcal{L}_{c1-reg}$ and $\mathcal{L}_{c2-reg}$ are for T-Net and center prediction, respectively, $\mathcal{L}_{h-cls}$ and $\mathcal{L}_{h-reg}$ are for heading, and $\mathcal{L}_{s-cls}$ and $\mathcal{L}_{s-reg}$ are for size. Classification and regression tasks have cross entropy and Huber [35] losses, respectively. In addition, a cornel loss term is used. The corner loss helps to minimize both the class and regression losses as it penalizes the network based on the distance between the corners of the predicted and ground truth bounding boxes [36]. It is computed as:

$$\mathcal{L}_{corner} = \sum_{i=1}^{N_S}\sum_{j=1}^{N_H}\delta_{ij}\min\left\{\sum_{h=1}^{8}||P_h^{ij} - P_h^{*}||, \sum_{i=1}^{8}||P_h^{ij} - P_h^{**}||\right\}. \tag{12}$$

where P_h^{**} denotes a corner of the flipped label box relative to the original label box P_h^{*}.

Calculating the energy score is straightforward with the classifier since logits $c(\mathbf{x})$ are just the class probabilities. However, the output of the box estimation network $b(\mathbf{x}) \in \mathbb{R}^{3+4\cdot N_S+2\cdot N_H}$ is a structure of heading and size class probabilities, and residual heading, size, and center residuals for each object class k (car, pedestrian, and cyclist). N_H and N_S denote the number of heading and size classes, respectively. Consequently, we must find an optimal way of using this special vector. We found experimentally that the heading $\mathbf{h} \in \mathbb{R}^{N_H}$ and the size $\mathbf{s} \in \mathbb{R}^{N_S}$ class vectors provide the best measure for ID/OOD separation, we ignore all residual elements because they did not have significant separation in the energy distributions. By ignoring all residual elements, a total of $K \cdot 2$ individual logit vectors remain, where K indicates the number of classes.

We start by examining the energy distributions of the default trained box estimator. Initial energy gaps are found in all logit vectors compared to their respective near-OOD

(NOOD) pairs, which makes it easier to train for larger energy gaps. This allows us to define a loss function from $K \cdot 2$ individual elements that will encourage the model to learn larger energy gaps on each vector pair. A near-OOD is an OOD sample that has passed the classifier pass-through module. It is good to note that the box estimator has a more challenging task than the classifier because it tries to separate NOODs from IDs, unlike the classifier that separates OODs from IDs. The box estimator is often uncertain about the heading angle of the ID samples in π intervals. Thus, the value in the $N_H/2$ offset of the maximum heading logit is changed to $-\infty$, which removes the contribution of that logit to the energy score since $\lim_{b_i(\mathbf{x})\to-\infty} e^{b_i(\mathbf{x})} = 0$:

$$\mathbf{h}_{\text{argmax } \mathbf{h} \pm N_H/2}(\mathbf{x}) = -\infty. \tag{13}$$

We define the loss as a weighted sum of squared hinge loss of each ID/NOOD heading and size pair.

$$\begin{aligned}\mathcal{L}_{box-energy} = \sum_{k=0}^{K}\sum_{g=0}^{G}(\mathbf{w}_k(&\mathbb{E}_{(\mathbf{x}_{ID},y_b)\sim\mathcal{D}_{ID}^{train}}((E(\mathbf{x}_{ID};b)-\mathbf{m}_{ID})^{+})^2 \\ &+\mathbb{E}_{\mathbf{x}_{NOOD}\sim\mathcal{D}_{NOOD}^{train}}((\mathbf{m}_{NOOD}-E(\mathbf{x}_{NOOD};b))^{+})^2))_{kg}\end{aligned} \tag{14}$$

where $\mathbf{w}_k = 1/\sqrt{\mathbf{N}_k}$ where $\mathbf{N}_k$ denotes the number of samples in an ID class k, $G = 2$ indicates the number of vectors per class.

We discovered that the critical and upper bound point sets for a sample $\mathbf{x}$ significantly differ in the classification and box estimation tasks. The classifier and the box estimator learn different sets of features of $\mathbf{x}$. This would explain why the box estimator network has different energy distributions than the classifier network.

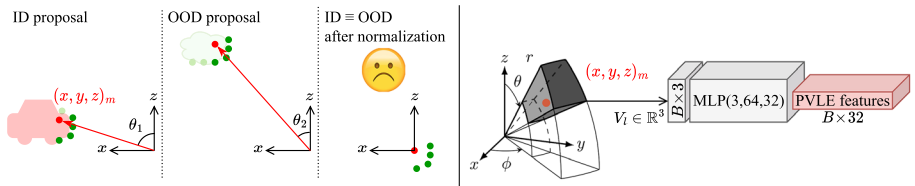
ID pass-through modules

During inference time, energy score for each sample $\mathbf{x}$ is computed from logits $c(\mathbf{x})$, $b(\mathbf{x})_{\mathbf{h}}$, and $b(\mathbf{x})_{\mathbf{s}}$ of the networks using equation (8). Since the classifier and bounding box estimator are optimized to their respective tasks, we implemented two separate ID pass-through modules to have more effective ID/OOD separation. The ID/OOD detection is computed with thresholds γ_c and $\gamma_b(k,g)$. The pass-through modules for the classifier and the bounding box estimator are defined, respectively, as.

$$p_1(\mathbf{x};\gamma_c,c) = \begin{cases}\text{in}, & \text{if } E(\mathbf{x};c) < \gamma_c \\ \text{out}, & \text{otherwise.}\end{cases} \tag{15}$$

$$p_2(\mathbf{x};\gamma_b(k,g),b) = \begin{cases}\text{in}, & \text{if } E(b(\mathbf{x})_{\mathbf{h}}) < \gamma_b(k,h) \\ & \text{and } E(b(\mathbf{x})_{\mathbf{s}}) < \gamma_b(k,s) \\ \text{out}, & \text{otherwise.}\end{cases} \tag{16}$$

where a sample $\mathbf{x}$ is an ID if classifier and box estimation energies are lower than γ_c and γ_b, respectively; otherwise, it is an OOD.



(a) An example illustrating an issue when the observation angle (θ_i) is discarded in the normalization process. The dark green points illustrate the critical point set.

(b) Proposal voxel location encoder. The input is the voxel coordinate of the proposal arithmetic mean point, marked in red.

Fig. 4 Proposal location normalization is shown in (**a**) and the proposal voxel location encoder is shown in (**b**)

Proposal voxel location encoder

The road user proposals are normalized on the origin before the classifier because it increases performance [34]. However, distance and observation angle information is lost during this process (Fig. 4a). To make use of this information, we propose a proposal voxel location encoder (PVLE) module (Fig. 4b), which aims to improve the point cloud proposal classification and 3D bounding box estimation tasks. The module processes voxel coordinates of the proposals and outputs learned features. The intuition behind this method is that the observation angle and the distance of an ID proposal carry useful information that can be used to improve the detection performance. In practice, the arithmetic mean point of a given proposal is first voxelized and then encoded into a small feature vector, which is concatenated with the global features of the classifier and bounding box networks as well as the R-Net and T-Net. Now, the networks can leverage the observation angle and distance. The design of this encoder (MLP: 3,64,32) is inspired by the first encoder layer of the PointNet (MLP: 3,64). The output feature vector has a length 32 because it is the closest 2^n to the voxel grid resolution 3×10. We utilize similar layer dimensions to the vanilla PointNet because the voxel coordinate input has the same shape as a point input in the vanilla PointNet.

Proposals shift to different voxel locations if the vehicle operates on uneven ground and the sensor pivots. With a spherical coordinate system, the magnitude of the shift is unrelated to the location of the proposal since angle limits define voxel boundaries. Therefore, a spherical coordinate system is more robust in this scenario than Cartesian and cylindrical coordinate systems.

Experiments

Experiments are conducted in three datasets. KITTI [37] and SemanticKITTI [38] are used to measure the accuracy of the 3D object detection and ground segmentation, respectively. Moreover, detection accuracy is also measured on our dataset with annotated road users collected with a compact mobile robot platform (Fig. 5). An in-depth quantitative analysis is carried out to validate our design choices. Lastly, qualitative results and a discussion of the strengths and weaknesses of our methods are presented.

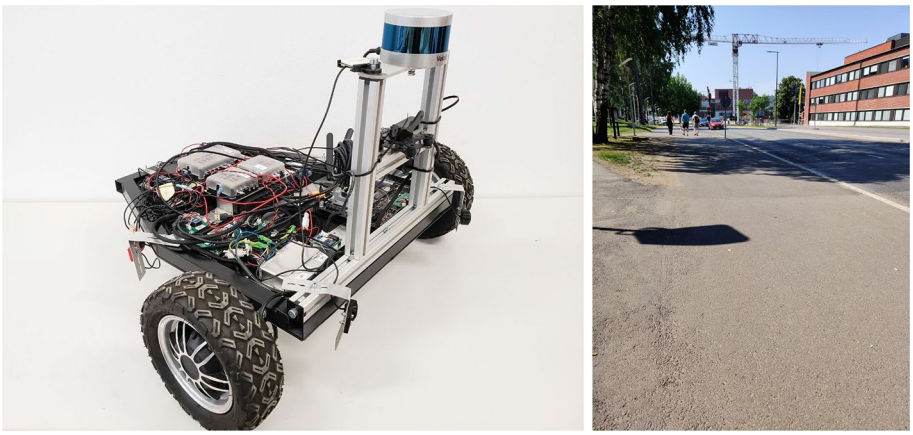


Fig. 5 The mobile robot platform equipped with a 16-channel LiDAR and the environment used for data collection

Table 1 The numbers of 3D bounding box labels in our 16-channel LiDAR point cloud dataset

Sequence	Car	Pedestrian	Van	Truck	Scans	FOV
1	1876	0	136	233	401	360°
2	258	1733	0	0	695	360°

This set is only used for testing

Implementation details

Inference time is benchmarked with a 4.0 GHz CPU. Both classifier and box estimator are trained for 200 epochs with a learning rate of 0.001 and with the Adam optimizer [39]. The resolution of the voxel grid is $\theta = 10°$, and $r = 7.5$ m. Proposals that have a higher number of points than 128 are sampled down to 128.

Data

Set 1. The KITTI dataset [37] is divided into training, validation, and test splits. All models are trained with the training split. The input of our architecture is instance samples; therefore, IDs, OODs, and NOODs are extracted from the training set into a separate set in the following manner. First, points inside ground truth bounding boxes are extracted as ID samples. Second, the remaining points are fed through the proposal generator. The output is saved into an auxiliary OOD dataset. Finally, OODs are fed through a trained classifier, the energy threshold is applied, and the samples that pass are saved into their respective class as NOODs.

Set 2. We also conduct tests with our dataset. It is collected with a compact mobile robot platform with a Velodyne VLP-16 LiDAR from a sidewalk area. There are 2130 and 1733 3D bounding box labels for cars and pedestrians in 1096 scans. The height of the sensor mount is 0.5 m from the ground. We want to emphasize that this dataset is used only for testing, not training. The detailed specifications are listed in Table 1.

Table 2 3D detection on the KITTI dataset

Method	Cars AP (0.7)			Pedestrians AP (0.5)			Cyclists AP (0.5)			Speed FPS	
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	GPU	CPU
VoxelNet [3]	87.99	82.47	77.34	55.35	48.79	46.22	78.44	66.40	59.20	2.3	0.1
SECOND [4]	88.21	81.32	75.23	53.35	49.66	45.79	79.53	64.69	57.32	4.7	1.2
IA-SSD [15]	89.44	81.32	76.36	45.39	41.51	37.77	79.46	62.55	57.05	19.8	4.3
PointPillars [9]	86.64	76.74	74.16	51.46	47.94	43.80	81.86	63.66	60.91	14.5	2.1
BirdNet [40]	24.14	19.22	16.32	42.72	31.25	28.21	41.39	26.92	26.42	9.1	1.7
BirdNet+ [41]	77.15	65.05	60.21	42.36	34.96	33.23	66.27	56.32	54.12	10.5	1.8
AVOD [42]	72.53	64.28	58.11	36.98	30.73	22.52	59.34	42.31	39.79	3.3	0.2
Ours	64.34	56.25	52.01	47.41	41.12	37.61	62.85	47.12	47.26	**76.1**	**15.2**

Bold font indicates the best FPS

GPU: Nvidia GTX 1060, CPU: 4.0 GHz Intel i5-7600K 5th generation

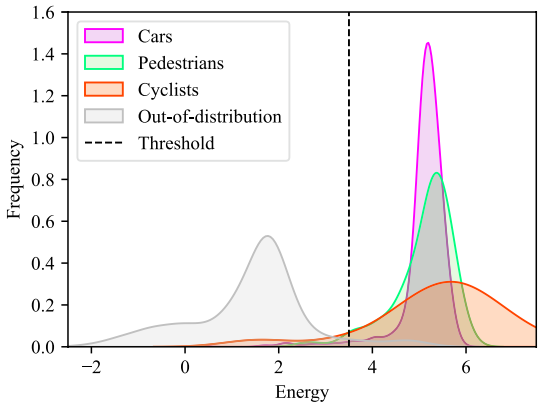


Fig. 6 Energy distributions from the out-of-distribution-optimized classifier display ID/OOD separability of each class

Set 3. The SemanticKITTI [38] is used to measure the accuracy of the ground segmentation methods. It includes point-wise labels for the ground surface in traffic scenarios and thus is ideal for our experiment.

Overall performance

The performance on the KITTI dataset is presented in Table 2. For the comparison, we chose state-of-the-art methods both in terms of speed and accuracy [9, 15], and frequent methods that are performing on a similar level to our method [3, 4, 40–42]. They use voxel and bird's eye view modalities, which make a fair comparison as our method utilizes voxels and point-based methods. Moreover, all methods are implemented using PyTorch [43] to have a more fair comparison.

Our method achieves similar AP on pedestrian and cyclist detection to other methods, which is impressive considering the computational cost of our approach. Table 2 compares the performance in terms of FPS and mAP. Our method is the only one that achieves real-time performance on a CPU.

Figure 6 illustrates the separability between ID and OOD samples in the classifier. The plot includes 10^4 samples from IDs and OODs, respectively. In the dataset,

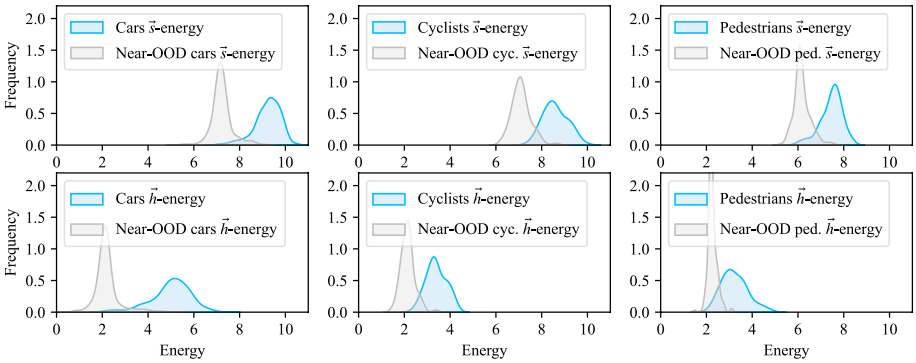


Fig. 7 Bounding box estimator energy distributions with the $\mathcal{L}_{box-energy}$ term in the training objective. The distributions display the NOOD/ID separability of each vector pair

Table 3 3D detection on the 16-channel LiDAR mobile robot dataset

Method	Cars AP (0.5)	Pedestrians AP (0.25)	Speed FPS	
			GPU	CPU
VoxelNet [3]	18.21	17.13	2.3	0.1
SECOND [4]	18.56	18.01	4.8	1.3
IA-SSD [15]	34.24	22.45	20.2	4.4
PointPillars [9]	19.22	18.88	18.1	2.3
Ours	34.25	15.69	**77.2**	**16.1**

Bold font indicates the best FPS

All methods are trained *only* with the KITTI dataset. GPU: Nvidia GTX 1060, CPU: 4.0 GHz Intel i5-7600K 5th generation

the actual partitions of IDs and OODs are approximately 6% and 94%, respectively. The energy distribution of cars is much narrower compared to other classes because the amount of car samples in the training set is more significant compared to other classes. This allows the network to learn the difference between cars and OODs better.

Figure 7 shows the energy distributions in the bounding box estimation network. The plot includes 10^4 samples from IDs and NOODs, respectively. The energy distributions have notable differences depending on the class and the vector type. This is a significant result, given that the classifier falsely detected these NOOD samples as ID samples. Car class has the best ID/NOOD separability. We suspect this is due to a large sample count in the training data compared to pedestrian and cyclist classes.

Performance on the 16-channel LiDAR dataset

Models are trained with the KITTI training set. Performance is tested with our annotated 16-channel LiDAR data. Therefore, this study investigates the performance of models on a low-resolution point cloud trained with a high-resolution point cloud. Furthermore, labels span full 360° in our dataset, unlike in the KITTI dataset, where labels are limited to approximately 90° sector. Our method performs on par with the state-of-the-art while achieving real-time performance on the CPU (Table 3). Our method performs the best in the study that measures the performance difference from the KITTI dataset to the low resolution and low perspective point cloud dataset (Table 4).

Table 4 Decrease in performance on the low-resolution point cloud dataset

Method	Cars ΔAP	Pedestrians ΔAP
VoxelNet [3]	64.39	32.99
SECOND [4]	63.02	31.59
IA-SSD [15]	48.13	19.10
PointPillars [9]	59.96	28.85
Ours	23.28	26.35

ΔAP = (KITTI AP) − (16chn-LiDAR AP) illustrates the performance gap for models trained with the KITTI dataset and tested with the 16-channel LiDAR mobile robot dataset. Smaller is better

Ablation study

Table 5 illustrates the contributions of the proposed modules to the mAP and FPS. Modules improve the mAP and FPS significantly. The first ID pass-through module improves the speed significantly, as it reduces the samples processed by the box estimator. Based on the ablation of the PVLE modules, the location carries valuable information regarding the 3D road user detection task, which satisfies the hypothesis discussed in the methods. The PVLE module for the box estimator slightly improves the car and cyclist classes while worsening it for the pedestrian class. This suggests that the module can be prone to overfit if the total number of samples is small, as it is for the pedestrian class.

Ground segmentation

The performance test results are summarized in Table 6. Our ground segmentation method is compared to frequent and state-of-the-art methods in the literature. It performs well in terms of computational cost, accuracy, and IOU. This is due to our effective sampling method, which reduces the iterations needed in the RANSAC function. Furthermore, fitting multiple planes in sensor azimuth direction yields more accurate segmentation, especially on an uneven ground surface.

Qualitative results and discussion

For qualitative analysis, we have randomly picked detection results. They are visualized in Fig. 8. Videos displaying the detection performance can be found at.[1] The geometrical proposal generator reduces the computational requirement significantly while still achieving mAP comparable to the state-of-the-art. The trade-off suggests that not using learned proposals is justified. The proposal generator has another benefit, too. It allows data streaming, meaning that the point clouds can be processed in smaller sectors to start the processing earlier than in whole scan approaches. This will decrease the latency of the detection significantly. The limitation of the proposal generator is cluster fusion when physical contact of the road users is visible to the sensor. This could be solved using another proposal generator, such as furthest point sampling. However, this is not the accuracy bottleneck of our approach since the performance increases when the IOU threshold is decreased. This is especially apparent with the car class, which has a harsh 0.7 IOU threshold. Although the car class separated better from the OODs than the pedestrian and cyclist classes, the final bounding box predictions were worse with the

[1] https://www.youtube.com/watch?v=CM1c2l8I3ac, https://www.youtube.com/watch?v=Q7VYJcX0UmY.

Table 5 Ablations of different modules and their contribution to the AP and the FPS on the KITTI dataset

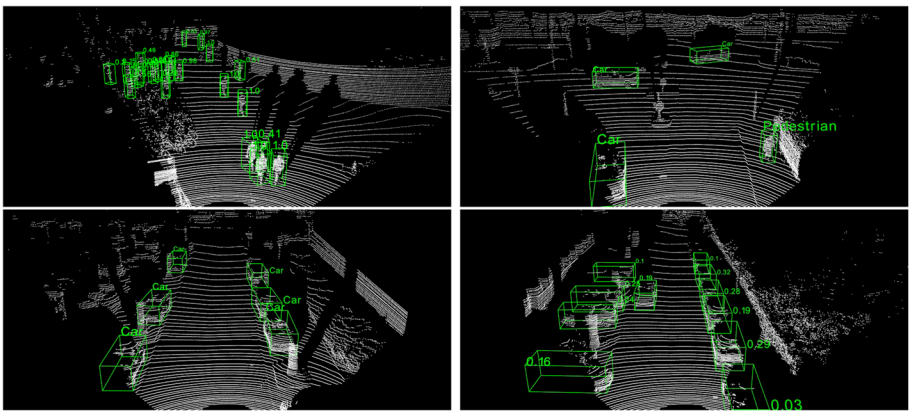
PVLE (cls)	PVLE (box)	IDP (cls)	IDP (box)	Cars AP			Pedestrians AP			Cyclists AP			Speed FPS	
				Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	GPU	CPU
–	–	–	–	44.44	39.27	36.81	34.37	31.32	29.31	39.43	31.31	30.10	64.5	6.5
–	✓	✓	✓	48.12	49.87	47.43	43.83	39.96	34.86	48.68	39.84	39.07	76.1	15.3
✓	–	✓	✓	62.83	55.91	51.68	47.28	42.44	39.20	54.93	43.51	43.06	76.1	15.3
✓	✓	–	✓	58.25	42.43	38.55	36.42	30.36	26.73	45.20	37.45	39.98	64.5	6.4
✓	✓	✓	–	61.43	52.34	49.33	43.31	38.32	32.42	58.32	44.32	42.43	76.1	15.2
✓	✓	✓	✓	64.34	56.25	52.01	47.41	41.12	37.61	62.85	47.12	47.26	76.1	15.2

PVLE: proposal voxel location encoder, IDP: ID pass-through module. When IDP(cls) is –, it is replaced with a typical softmax confidence threshold

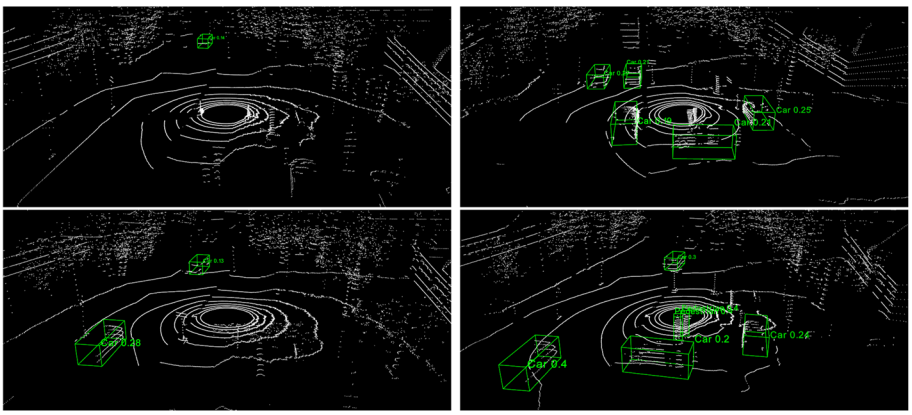
Table 6 A comparison of ground segmentation methods

Method	Dataset	Scans	Time (s)	Precision	Recall	Accuracy	IOU
HD [44]	SemKITTI	23201	0.306	0.47	0.95	–	0.45
LF [44]	SemKITTI	23201	0.658	0.38	0.77	–	0.34
GPF [44]	SemKITTI	23201	31.71	0.67	0.63	–	0.45
GPF-Opti [44]	SemKITTI	23201	0.207	0.66	0.59	–	0.43
GPF-RANSAC [44]	SemKITTI	23201	0.028	0.65	0.88	–	0.74
Hybrid-reg [45]	KITTI	5	0.888	–	–	0.88	–
CNN-method [46]	Custom	252	0.139	0.93	0.99	–	–
CRF-method [47]	SemKITTI	3040	0.147	0.80	–	–	0.78
GndNet [48]	SemKITTI	3040	0.018*	0.84	0.99	–	0.84
Ours	SemKITTI	23201	0.016	0.89	0.93	0.93	0.83

Our results are achieved with a single core of a 4.0 GHz Intel i5-7600K 5th generation CPU. 360° scan is divided evenly into 16 sectors. * – GPU inference



(a) KITTI dataset.



(b) 16-channel LiDAR mobile robot dataset.

Fig. 8 Randomly picked detection results on the KITTI dataset (**a**) and on the 16-channel LiDAR dataset (**b**). We suggest zooming in for better detail

car class. Therefore, the accuracy bottleneck is in the bounding box estimator network caused by incorrectly predicted bounding boxes on correctly classified proposals. Thus, improvements to this network would be a good subject of interest in future research.

How does the OOD training objective affect the accuracy of the ID classification and bounding box estimation? By adding an energy-based OOD training objective, networks learn not only the original task but also the energy-based task. This decreases the performance of the original task slightly. However, many false positives are removed using the energy values, which increase AP more than rare classification errors decrease it.

Is the inductive bias of the proposal voxel location encoder beneficial? The bias of the module is beneficial as the voxel grid resolution is relatively low. This results in more proposals for a single voxel location, which results in a more general representation of the location. Hence, the model is not prone to overfit to voxel location information. This is indicated by the results in Table 5.

Conclusion

This paper presented a novel architecture for the 3D road user detection task. The architecture has an extremely low computational requirement; therefore, it is suitable for applications with limited computational resources. An impressive 15.2 FPS was achieved with a 4.0 GHz CPU-only implementation while having comparable accuracy to the state-of-the-art. Furthermore, our architecture performed the best on the low-resolution LiDAR dataset. The architecture is based on a geometrical proposal generator and out-of-distribution- and location-aware PointNets. To our surprise, the accuracy bottleneck was not the proposal generator but the bounding box estimator. In the future, improvements to the bounding box estimator could be carried out, and other OOD detection methods could be studied in the 3D road user detection task.

Acknowledgements
Not applicable.

Author contributions
AS developed the research idea, implemented the algorithm, designed and executed the experiments, and drafted the manuscript. EA helped with the data collection and labeled the dataset. GD provided fruitful comments on the interpretability of the schematic figures. RO and KT provided general research guidance, managed the research workflow, and revised the initial versions of the manuscript. All authors read and approved the final manuscript.

Funding
The Helsinki Institute of Physics funded this work.

Availability of data and materials
The KITTI dataset is publicly available, and the dataset generated during the current study is available from the corresponding author upon reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 27 March 2023 Accepted: 14 December 2023
Published online: 02 January 2024

References

1. Horowitz M. 1.1 computing's energy problem (and what we can do about it). In: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014; pp. 10–14. IEEE.
2. Huang C, Nguyen VD, Abdelzad V, Mannes CG, Rowe L, Therien B, Salay R, Czarnecki K Out-of-distribution detection for lidar-based 3d object detection. In: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), 2022; pp. 4265–4271.
3. Zhou Y, Tuzel O. Voxelnet: end-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018; pp. 4490–4499.
4. Yan Y, Mao Y, Li B. Second: sparsely embedded convolutional detection. Sensors. 2018;18(10):3337.
5. Fan L, Xiong X, Wang F, Wang N, Zhang Z. Rangedet: in defense of range view for lidar-based 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021; pp. 2918–2927.
6. Chai Y, Sun P, Ngiam J, Wang W, Caine B, Vasudevan V, Zhang X, Anguelov D. To the point: Efficient 3d object detection in the range image with graph convolution kernels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021; pp. 16000–16009.
7. Liang Z, Zhang M, Zhang Z, Zhao X, Pu S. Rangercnn: towards fast and accurate 3d object detection with range image representation, 2020; arXiv preprint arXiv:2009.00206.
8. Meyer GP, Laddha A, Kee E, Vallespi-Gonzalez C, Wellington CK. Lasernet: an efficient probabilistic 3d object detector for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019; pp. 12677–12686.
9. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O. Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019; pp. 12697–12705.
10. Yang B, Luo W, Urtasun R. Pixor: real-time 3d object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018; pp. 7652–7660.
11. Zheng W, Tang W, Jiang L, Fu C-W. Se-ssd: self-ensembling single-stage object detector from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021; pp. 14494–14503.
12. Zheng W, Tang W, Chen S, Jiang L, Fu C-W. Cia-ssd: confident iou-aware single-stage object detector from point cloud. arXiv preprint, 2020; arXiv:2012.03015.
13. Shi W, Rajkumar R. Point-gnn: graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020; pp. 1711–1719.
14. Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Proc Syst. 2017;30.
15. Zhang Y, Hu Q, Xu G, Ma Y, Wan J, Guo Y. Not all points are equal: learning highly efficient point-based detectors for 3d lidar point clouds. Accepted to CVPR 2022, arXiv preprint, 2022; arXiv:2203.11139.
16. Qi CR, Litany O, He K, Guibas LJ. Deep Hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019; pp. 9277–9286.
17. Shi S, Wang X, Li H. Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019; pp. 770–779.
18. Ngiam J, Caine B, Han W, Yang B, Chai Y, Sun P, Zhou Y, Yi X, Alsharif O, Nguyen P, et al. Starnet: targeted computation for object detection in point clouds. arXiv preprint , 2019;arXiv:1908.11069.
19. Sobel I. An isotropic 3x3 image gradient operator. Presentation at Stanford A.I. Project 1968; 2014.
20. Bogoslavskyi I, Stachniss C. 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), 2016; pp. 163–169. IEEE.
21. Zhao Y, Zhang X, Huang X. A technical survey and evaluation of traditional point cloud clustering methods for lidar panoptic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021; pp. 2464–2473.
22. Rusu RB. Semantic 3d object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz. 2010;24(4):345–8.
23. Rusu RB, Cousins S. 3d is here: point cloud library (pcl). In: 2011 IEEE International Conference on Robotics and Automation, 2011; pp. 1–4. IEEE.
24. Papon J, Abramov A, Schoeler M, Worgotter F. Voxel cloud connectivity segmentation-supervoxels for point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,2013; pp. 2027–2034.
25. Zermas D, Izzat I, Papanikolopoulos N. 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), 2017; pp. 5067–5073. IEEE.
26. Dong X, Guo J, Li A, Ting W-T, Liu C, Kung H. Neural mean discrepancy for efficient out-of-distribution detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; pp. 19217–19227.
27. Liang S, Li Y, Srikant R. Enhancing the reliability of out-of-distribution image detection in neural networks. arXiv preprint, 2017; arXiv:1706.02690.
28. Hsu Y-C, Shen Y, Jin H, Kira Z. Generalized odin: detecting out-of-distribution image without learning from out-of-distribution data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020; pp. 10951–10960.
29. Ren J, Fort S, Liu J, Roy AG, Padhy S, Lakshminarayanan B. A simple fix to mahalanobis distance for improving near-ood detection. arXiv preprint, 2021; arXiv:2106.09022.
30. Lee K, Lee K, Lee H, Shin J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. Adv Neural Inf Proc Syst 2018; 31.
31. LeCun Y, Chopra S, Hadsell R, Ranzato M, Huang F. A tutorial on energy-based learning. Predicting structured Data. 2006; 1(0).
32. Liu W, Wang X, Owens J, Li Y. Energy-based out-of-distribution detection. Adv Neural Inf Process Syst. 2020;33:21464–75.

33. Hinton GE, Zemel R. Autoencoders, minimum description length and helmholtz free energy. Adv Neural Inf Proc Syst. 1993; 6.
34. Qi CR, Su H, Mo K, Guibas LJ. Pointnet: deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017; pp. 652–660.
35. Huber PJ. Robust estimation of a location parameter. In: Breakthroughs in Statistics, 1992; pp. 492–518. Springer.
36. Qi CR, Liu W, Wu C, Su H, Guibas LJ. Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018; pp. 918–927.
37. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The Kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012; pp. 3354–3361 . IEEE.
38. Behley J, Garbade M, Milioto A, Quenzel J, Behnke S, Gall J, Stachniss C. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: the SemanticKITTI Dataset. Int J Robot Res. 2021;40(8–9):959–67.
39. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
40. Beltrán J, Guindel C, Moreno FM, Cruzado D, Garcia F, De La Escalera A. Birdnet: a 3d object detection framework from lidar information. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018; pp. 3517–3523. IEEE.
41. Barrera A, Beltrán J, Guindel C, Iglesias JA, García F. Birdnet+: two-stage 3d object detection in lidar through a sparsity-invariant bird's eye view. IEEE Access. 2021;9:160299–316.
42. Ku J, Mozifian M, Lee J, Harakeh A, Waslander SL. Joint 3d proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018; pp. 1–8. IEEE.
43. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Proc Syst. 2019; 32.
44. Ouyang Z, Dong X, Cui J, Niu J, Guizani M. Pv-enconet: fast object detection based on colored point cloud. IEEE Trans Intel Transport Syst. 2021.
45. Liu K, Wang W, Tharmarasa R, Wang J, Zuo Y. Ground surface filtering of 3d point clouds based on hybrid regression technique. IEEE Access. 2019;7:23270–84.
46. Velas M, Spanel M, Hradis M, Herout A. Cnn for very fast ground segmentation in velodyne lidar data. In: 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2018; pp. 97–103. IEEE.
47. Rummelhard L, Paigwar A, Nègre A, Laugier C. Ground estimation and point cloud segmentation using spatiotemporal conditional random field. In: 2017 IEEE Intelligent Vehicles Symposium (IV),2017; pp. 1105–1110. IEEE.
48. Paigwar A, Erkent Ö, Sierra-Gonzalez D, Laugier C. Gndnet: fast ground plane estimation and point cloud segmentation for autonomous vehicles. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020; pp. 2150–2156 . IEEE.

Publisher's Note