Mechanical Systems and Signal Processing 204 (2023) 110723

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

VINet: Lightweight, scalable, and heterogeneous cooperative perception for 3D object detection

Zhengwei Bai [a,*], Guoyuan Wu [a], Matthew J. Barth [a], Yongkang Liu [b], Emrah Akin Sisbot [b], Kentaro Oguchi [b]

[a] University of California, Riverside, Riverside, CA, United States of America
[b] Toyota North America InfoTech Labs, Mountain View, CA, United States of America

ARTICLE INFO

Communicated by J.E. Mottershead

Keywords:
Cooperative perception
3D object detection
Deep fusion
Artificial intelligence
System-level cost analysis

ABSTRACT

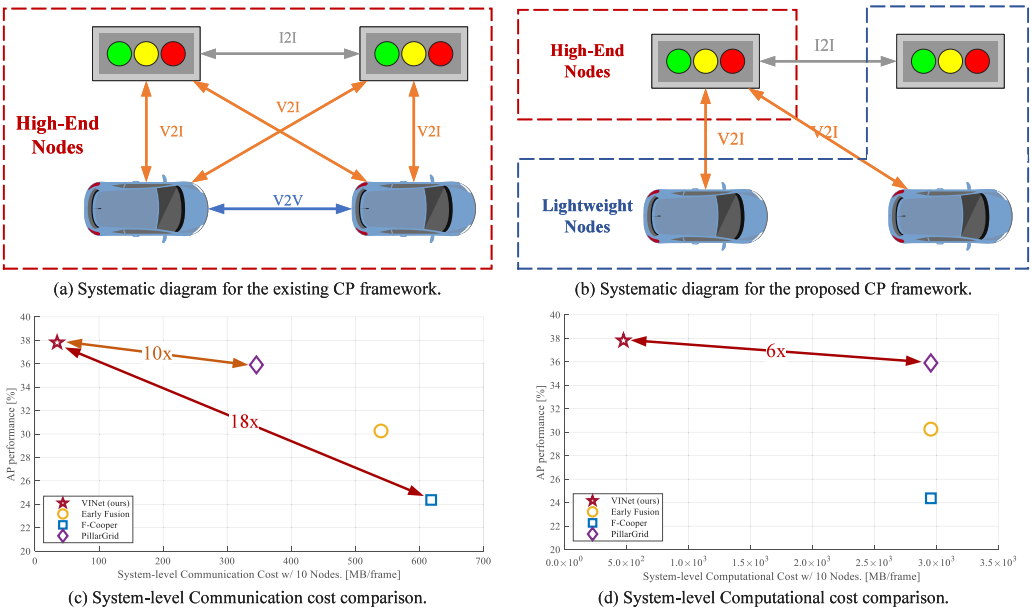
Utilizing the latest advances in Artificial Intelligence (AI), the computer vision community is now witnessing an unprecedented evolution in all kinds of perception tasks, particularly in object detection. Based on multiple spatially separated perception nodes, Cooperative Perception (CP) has emerged to significantly advance the perception of automated driving. However, current cooperative object detection methods mainly focus on ego-vehicle efficiency without considering the practical issues of system-wide costs. In this paper, we introduce VINet, a unified deep learning-based CP network for scalable, lightweight, and heterogeneous cooperative 3D object detection. VINet is the first CP method designed from the standpoint of large-scale system-level implementation and can be divided into three main phases: (1) Global Pre-Processing and Lightweight Feature Extraction which prepare the data into global style and extract features for cooperation in a lightweight manner; (2) Two-Stream Fusion which fuses the features from scalable and heterogeneous perception nodes; and (3) Central Feature Backbone and 3D Detection Head which further process the fused features and generate cooperative detection results. An open-source data experimental platform is designed and developed for CP dataset acquisition and model evaluation. The experimental analysis shows that VINet can reduce 84% system-level computational cost and 94% system-level communication cost while improving the 3D detection accuracy.

1. Introduction

Throughout the world, transportation demand has increased in terms of the movement of people and goods on a daily basis. It is important to note, however, that a rapidly increasing number of vehicles has resulted in several major problems to the current transportation system, including safety [1], mobility [2], and environmental sustainability [3]. In order to improve system-wide performance, Cooperative Driving Automation (CDA) has emerged, leveraging recent advances in advanced sensing, wireless connectivity, and artificial intelligence. CDA enables connected and automated vehicles (CAVs) to communicate with one another, with the roadway infrastructure, and/or with pedestrians and cyclists equipped with mobile devices. In the past few years, CDA has gained increasing attention and is now considered a transformative solution to these challenges [4]. CDA applications are fundamentally dependent on object perception data from the surrounding environments, which is similar to the visual function of

* Corresponding author.
E-mail addresses: zbai012@ucr.edu (Z. Bai), gywu@cert.ucr.edu (G. Wu), barth@ece.ucr.edu (M.J. Barth), yongkang.liu@toyota.com (Y. Liu), akin.sisbot@toyota.com (E.A. Sisbot), kentaro.oguchi@toyota.com (K. Oguchi).

https://doi.org/10.1016/j.ymssp.2023.110723
Received 21 March 2023; Received in revised form 14 June 2023; Accepted 24 August 2023
Available online 18 September 2023



(a) Systematic diagram for the existing CP framework.

(b) Systematic diagram for the proposed CP framework.

(c) System-level Communication cost comparison.

(d) System-level Computational cost comparison.

Fig. 1. The proposed framework for scalable, lightweight, and heterogeneous CP: (a) A free communication and computation assumption for most of the existing CP systems; (b) The proposed CP system framework. For a certain area, a central node is equipped with powerful computing resources while other nodes are only equipped with lightweight computing resources. Lightweight nodes will generate features for cooperation while the central node will generate the cooperative perception results and then broadcast to the lightweight nodes; (c) The proposed approach achieves 10× to 18× smaller communication cost compared with the baselines; and (d) The proposed approach achieves 6× smaller computational cost than the baselines.

automated agents [5]. As the system input, perception data can support a variety of CDA applications, such as Collision Warning [6], Eco-Approach and Departure (EAD) [7], and Cooperative Adaptive Cruise Control (CACC) [8].

In the past few decades, sensing technology has made it possible for transportation systems to retrieve high-fidelity traffic data from various types of sensors. The ability of situation awareness varies among different sensors both onboard vehicles or at the roadside. Cameras can provide detailed vision data for classifying various kinds of traffic objects, including vehicles, pedestrians, and cyclists [9]. On the other hand, high-fidelity 3D point cloud data can be retrieved by LiDAR sensors to grasp the precise 3D location of the traffic objects [10]. The robust performance of radar sensors in variable environmental conditions has made them an integral part of safety-critical automotive applications [11].

Perceiving the surrounding environment based on intelligent entities themselves has been one of the main methodologies to promote the development of Autonomous Driving Technology (ADT) [12]. The difference in the quality of perception would cause a significant impact on the subsequent output for the ADT applications [13]. Automated vehicles are now equipped with more onboard sensors and powerful mobile computing that can process massive amounts of sensor data [14,15]. Not only for onboard sensors, but recent researchers are now also conducting some studies that use infrastructure sensors that have the capability to perceive the traffic conditions at the object level [16,17]. These different sensing entities can be regarded as spatially separated perception nodes (PNs) in a transportation system. Even when empowered with advanced perception methods, both Vehicle-based PNs (V-PNs) and Infrastructure-based PNs (I-PNs) have their specific and common limitations to perceiving the environment on their own. Specifically, V-PNs are inevitably limited by the occlusion of other road objects, while I-PNs are limited by the field of view (FOV) and blind zones. From a common point of view, neither V-PNs nor I-PNs themselves can perceive objects that are physically occluded or out of their sensing range. Thus to get past the bottleneck of single-PN perception, cooperation perception has emerged and is attracting increasing attention from researchers.

From an intuitive perspective, collaborating spatially separated PNs, i.e., Cooperative Perception (CP), naturally becomes a transformative solution to improving the perception breadth and accuracy. Therefore, numerous studies have been conducted to generate the perception information in a cooperative manner [18–22]. In terms of the stage of sensor fusion, CP can be divided into three main categories: (1) early fusion in which raw sensor data is shared and combined [23], (2) late fusion in which perception results (i.e., bounding boxes) from each PNs are fused [18], and (3) deep fusion in which feature data from each PNs are shared and fused [20]. Different fusion schemes have their own merits and drawbacks. Early fusion only needs the calibration for aligning multi-source data into a unified coordinate system but requires a large communication bandwidth for transmitting data and is more sensitive to noise and delay [22] and is difficult to fuse multi-modal sensor data. Late fusion mainly focuses on how to merge the perception results generated from multiple perception pipelines, which is straightforward but suffers from limited accuracy [18]. Deep fusion is capable of generating high-performance perception results with low-communication requirements (compared to early fusion), but the development is still in its infancy and only a few studies have been conducted [19,21].

Furthermore, due to the heterogeneity of V-PNs and I-PNs, only a couple of studies have focused on the vehicle-infrastructure CP [20,22]. A real-world traffic scenario is always a dynamically changing system, which requires the CP methods to be able to cope with scalable PNs. Meanwhile, as shown in Fig. 1(a), most of the existing CP methods assume free communication and unlimited computing power for every node, which would be extremely costly to enable such a CP system in real-world conditions. It is still an open-challenging task to design a cooperative perception system that considers these real-world implementation aspects mentioned above.

Hence, in this paper, we propose a novel framework for cooperative perception as shown in Fig. 1(b). The main ideology is to redesign the fundamental topology among the cooperating nodes from a fully-connected graph to a centralized structure. Following this framework, a cooperative perception model named *VINet* (***V**ehicle-**I**nfrastructure **Net**work*), is proposed to enable scalable and lightweight cooperation of heterogeneous nodes. VINet mainly consists of five components: (1) Global Pre-Processing (GPP) which transforms the raw sensor data based on the global referencing coordinate; (2)Lightweight Feature Extraction (LFE) which generates deep features from raw sensor data with low-computational requirements; (3) Two-Stream Fusion (TSF) that fuses the features generated from scalable and heterogeneous PNs; (4) Central Feature Backbone (CFB) that extract hidden feature from the fused data; and (5) 3D Detection Head (3DH) that generates rotated 3D bounding boxes. The main contributions of this paper can be summarized as follows:

- We introduce the first unified CP framework to offer object detection with low system-level communication and computational cost;
- We propose VINet, the first deep learning-based scalable, lightweight, and heterogeneous CP method;
- We propose TSF, a novel deep-fusion model for heterogeneous feature data from scalable PNs; and
- We design and develop an open-source cooperative perception platform for supporting CP training and evaluation in scalable and heterogeneous environments.

The remainder of this paper is organized as follows: related work is reviewed in Section 2 to give a quick glance at the basic CP background. VINet is introduced in Section 3 with details of each consisting component. Section 4 illustrates experimental details about the training and evaluation process, followed by Section 6 that concludes the paper and highlights future work.

2. Related work

2.1. General 3D object detection from point clouds

Before 2015, one of the most popular mechanisms to handle point cloud data (PCD) from LiDAR sensors is the bottom-up pipeline based on traditional methods, such as "*Clustering [24]→Classification [25]→Tracking [26]*". Due to their explainability and free from data labeling, traditional bottom-up methodologies are still popular in current infrastructure-based LiDAR perception tasks [17,27].

With the great success achieved by convolutional neural networks (CNNs) in image-based object perception, PCD quickly became the upcoming target for CNNs. Point-wise manipulation is considered straightforward for extracting features from PCD for object detection [28]. Endowed with the natural fit, point-based methods provide dominant performance in detection accuracy, however, it scarifies computational efficiencies [29].

Since PCD consists of 3-dimensional sparse data, creatively cut the whole 3D point cloud into 3D voxels grids. A specific voting scheme, named *Vote3D*, was designed in 2015. In 2018, VoxelNet [30] was proposed, which introduced a learnable voxel encoder to generate hidden features of voxels. This voxelization mechanism has been widely used in various works, such as *SECOND* [31], *PointPillar* [32], *Voxel RCNN*

Projecting PCD into a 2D bird's-eye view (BEV) feature map has quickly become a popular methodology. Inspired by YOLO, [33] proposed *ComplexYolo* which projected PCD into three manually defined feature channels, and then the BEV feature map was fed into a 2D backbone for generating detection results. Since the BEV scheme provides a straightforward way for solving 3D data in 2D manners, lots of BEV-based methods have emerged such as *PIXOR* [34], *SCANet* [35], *BEVFusion* [36], etc.

2.2. Roadside LiDAR-based object detection

In recent years, roadside LiDAR sensors have received increasing attention from researchers about object perception in transportation [37]. Using roadside LiDAR, Zhao et al. proposed a detection and tracking approach for pedestrians and vehicles [17]. As one of the early studies utilizing roadside LiDAR for perception, a classical detection and tracking pipeline for PCD was designed. It mainly consists of (1) *Background Filtering*: To remove the laser points reflected from road surfaces or buildings by applying a statistics-based background filtering method [38]; (2) *Clustering*: To generate clusters for the laser points by implementing a Density-based spatial clustering of applications (DBSCAN) method (3) *Classification*: To generate different labels for different traffic objects, such as vehicles and pedestrians, based on neural networks [39]; and (4) *Tracking*: To identify the same object in continuous data frames by applying a discrete Kalman filter [26].

Based on the aforementioned work, Cui et al. designed an automatic vehicle tracking system by considering vehicle detection and lane identification [40]. In addition, a real-world operational system was developed, consisting of a roadside LiDAR, an edge computer, a *Dedicated Short-Range Communication* (*DSRC*) *Roadside Unit* (*RSU*), a Wi-Fi router, and a DSRC *On-board Unit* (*OBU*), and a GUI. Following a similar workflow, Zhang et al. proposed a vehicle tracking and speed estimation approach based on a roadside LiDAR [41]. Vehicle detection results were generated by the "*Background Filtering-Clustering-Classification*" process. Then, a

centroid-based tracking flow was implemented to obtain initial vehicle transformations, and the unscented Kalman Filter [42] and joint probabilistic data association filter [43] were adopted in the tracking flow. Finally, vehicle tracking was refined through a BEV LiDAR-image matching process to improve the accuracy of estimated vehicle speeds. Following the bottom-up pipeline mentioned above, numerous roadside LiDAR-based methods have been proposed from various points of view [27,44–47].

On the other hand, using learning-based models to cope with LiDAR data is another primary stream. Bai et al. [13] proposed a deep-learning-based real-time vehicle detection and reconstruction system from roadside LiDAR data. Specifically, the CARLA simulator [48] was implemented for collecting the training dataset, and ComplexYOLO model [33] was applied and retrained for object detection. Finally, a co-simulation platform was designed and developed to provide vehicle detection and object-level reconstruction, which aimed to empower subsequent cooperative driving automation (CDA) applications with readily retrieved authentic detection data. In their following work for real-world implementation, Bai et al. [16] proposed a deep-learning-based 3D object detection, tracking, and reconstruction system for real-world implementation. The field operational system consisted of three main parts: (1) 3D object detection by adopting PointPillar [32] for inference from roadside PCD; (2) 3D multi-object tracking by improving DeepSORT [49] to support 3D tracking, and (3) 3D reconstruction by geodetic transformation and real-time onboard *Graphic User Interface* (*GUI*) display.

By combining traditional and deep learning algorithms Gong et al. [50] proposed a roadside LiDAR-based real-time detection approach. Several techniques were designed to guarantee real-time performance, including the application of Octree with region-of-interest (ROI) selection, and the development of an improved Euclidean clustering algorithm with an adaptive search radius. The roadside system was equipped with *NVIDIA Jetson AGX Xavier*, achieving the inference time of 110 ms per frame.

2.3. Multi-node cooperative object detection

Although one single LiDAR can provide panoramic FOV around the ego-vehicle, physical occlusion may easily block the lines of sight and cause the ego-vehicle to lose some crucial perception information, which significantly affects its decision-making or control process. Additionally, a spatially separated LiDAR perception system can expand the perceptive range for intelligent vehicles or smart infrastructure.

One of the straightforward inspirations of the multi-LiDAR perception system is sharing the raw PCD via V2V communications [23]. However, limited wireless communication bandwidth may significantly limit real-time performance. Feature data generated from CNN requires much less bandwidth and is more robust to sensor noise, thus becoming a popular solution to multi-LiDAR fusion [19,21]. Marvasti et al. [51] used two sharing-parameter CNNs to extract the feature map for PCD retrieved from two-vehicle nodes. Feature maps were then aligned based on the relative position and fused by element-wise summation. By applying an attention mechanism, Xu et al. [52] proposed a V2V-based cooperative object detection method. A similar CNN process [32] was designed for extracting feature maps for V2V sharing. In real-world scenarios, researchers have initiated the investigation of V2V-based cooperative perception including the advanced multi-modal sensor processing [53], accurate vehicles pose estimation [54], and systematic sensor fusion technology [55], which proved the conception and boosted to deploy the cooperative perception into large-scale applications to address the uncertainties of the sensor measurement, observability issues, occlusion and long-range object detection and tracking problems.

Recently, researchers started focusing on cooperation between heterogeneous perception nodes such as vehicles and infrastructure. For handling the data heterogeneity from the roadside and onboard sensors, Bai et al. [20] proposed a decoupled multi-stream CNN framework for generating feature maps accordingly. Relative position information was applied to PCD alignment, and the shared feature maps were then fused based on grid-wise *maxout* operation. Additionally, Xu et al. [22] proposed a Transformer-based fusion method for cooperative perception among heterogeneous nodes.

3. Methodology

3.1. Overview of VINet

The main purpose of VINet is to reduce the system-wide computational cost by dividing the whole perception pipeline into two phases: (1) the lightweight feature extraction, and (2) the feature fusing and processing. Additionally, perception nodes (PNs) are categorized into two types: (1) *Central Nodes* which are equipped with powerful computing devices, named the Central Computing Unit (CCU), and (2) *Slave Nodes* which are equipped with Light-weight Computing Unit (LCU). In this paper, illustrated in Fig. 2, one intersection PN (I-PN) is assigned as Central Node (I-PN-C) while other vehicle-PNs (V-PNs) and I-PN-S (I-PN-Slave) are assigned as Slave Nodes. Therefore, by assigning a lightweight feature extraction pipeline to Slave Nodes, only LCUs are needed for V-PNs and the I-PN-S. Thus, the whole cooperative perception (CP) system will only need one CCU and several LCUs to make CP available for every node with connectivity.

The network overview of VINet is shown in Fig. 2. To reduce inter-node communication costs, Global Meta Data is designed and transmitted to all the PNs. From the system perspective, unlike previous CP methods which require inter-node local-coordinate transformation [21,22] with a communication cost of $\mathbf{O}(N^2)$, VINet only needs global transformation once for each node which has the communication cost of $\mathbf{O}(N)$ (N represents the number of PNs).

Based on Fig. 2, the workflow of VINet can be summarized as follows:

- Global Pre-Processing (GPP): Based on LCU, raw PCD are transformed to the global coordinate and then geo-fenced within the predefined CP range.

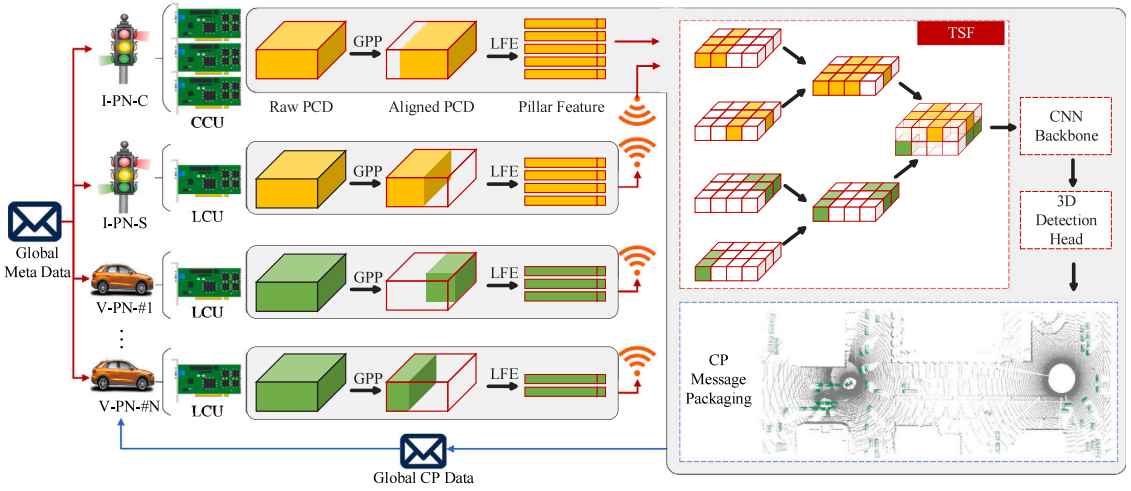


Fig. 2. Systematic diagram of the proposed cooperative perception system – VINet. It consists of five components: Global Pre-Processing (GPP), Lightweight Feature Extraction (LFE), Two-Stream Fusion (TSF), Central Feature Backbone (CFB), and 3D Detection Head (3DH).

- Lightweight Feature Extraction (LFE): based on LCU, globally aligned PCD are fed into a pillar feature extraction network to generate hidden features and transmit them to the Central Node.
- Two-Stream Fusion (TSF): based on CCU, pillar features from heterogeneous nodes are fused into a unified feature map.
- Central Feature Backbone (CFB): based on CCU, a multi-scale CNN backbone generate a feature map from fused feature data.
- 3D Detection Head (3DH): based on CCU, detected bounding boxes are generated with class types for further broadcasting to all the connected nodes.

The following sections will introduce the aforementioned modules and details about the loss function applied in VINet.

3.2. Global pre-processing

3.2.1. Global transformation

For global coordinate referencing, we design sensor calibration matrices that include the global 3D location and rotation information of the sensor, which is named as *Sensor Location and Pose* (SLaP) data. Raw PCD is originally recorded and organized in the 3D Cartesian coordinate with the center of the sensor of each node. Currently, one popular way of cooperative data transformation is to transform the PCD to the ego-vehicle's coordinate [22], otherwise, there will exist spatial matching issues [19].

Since the PCD is collected based on a 3D Cartesian coordinate centered with the SLaP of the sensor, cooperative transformation is designed to unify the PCD from different sensors. The raw PCD can be described by:

$$P = \{[x, y, z, i]^T \mid [x, y, z]^T \in \mathbf{R}^3, i \in [0.0, 1.0]\} \tag{1}$$

The SLaP information is defined by the 3D location and rotation of the sensor:

$$\text{SLaP} = [X, Y, Z, P, \Theta, R] \tag{2}$$

where X, Y, Z, P, Θ, and R represent the 3D location along x-axis, y-axis, z-axis; and the pitch, yaw, and roll angles of the sensors in the global coordinate, respectively.

To avoid square-level inter-node communication which requires unreasonable system-wide communication costs, GPP aims to transform the PCD from PNs' coordinates to a unified static global coordinate. Two main advantages of GPP compared with traditional inter-node transformation can be identified: (1) only linearly increasing communication cost is required for the whole CP system, and (2) a static global coordinate can have much less positioning error compared with the ego-vehicle coordinate.

3.2.2. Cooperative geo-fencing

After the global transformation, a geo-fencing process is applied to PCD. In this paper, the detected region Ω for each of the LiDAR sensors is defined as a 280 m × 90 m area centered at the location of the respective LiDAR. Specifically, $P^{S\to G}$ is geo-fenced by:

$$P_{\Omega}^{S\to G} = \{[x, y, z, i]^T \mid x \in X, y \in Y, z \in Z\} \tag{3}$$

where $P_{\Omega}^{S\to G}$ represents the 3D point cloud data after geo-fencing; and X, Z and Y are defined by the CP range, which is further described in Section 4.2.

3.3. Lightweight feature extraction

Although deep features via the CNN backbone are intuitively suitable for CP tasks [19], it requires that every PNs have the capability of supporting the whole feature extraction pipeline. To distinguish our lightweight feature, the features generated after the whole CNN backbone are named *dense features*. In one of the popular deep fusion-based CP methods, *F-Cooper* [19], dense features need to be calculated for all the involved PNs and then shared with each other. Since a CNN backbone is generally a deep neural network with a large amount of computing requirement [31,56], deploying every PNs with powerful computing devices is a significant load for real-world implementation and commercialization. Additionally, dense features usually have a tremendous data size and thus they require powerful data compression techniques [21] to make the pipeline feasible in terms of real-time performance, which may inevitably cause perception performance drop and/or additional computing costs.

To reduce the computing requirement from the perspective of sharing feature extraction, a lightweight feature extraction (LFE) module is designed in this study. Inspired by [32], LFE aims to generate pillar features without the involvement of the CNN backbone, which can significantly reduce the computational loads on PNs. To cope with the heterogeneity of PCD from vehicles and infrastructures, two decoupled LFE blocks, i.e., vehicle LFE (V-LFE) and infrastructure LFE (I-LFE), are designed, which have the same network structure but different parameters.

For each LFE, the first part is to generate a $D-$dimensional feature vector V_p for all points in pillars:

$$V_p = \{\left[x, y, z, i, x_c, y_c, z_c, x_p, y_p\right]_i\}_{i=1}^{N}, p = 1, \ldots, P \tag{4}$$

where $x_c, y_c, z_c, x_p, y_p, N$, and P represent the distance of each point to the arithmetic center of all points in the $p-$th pillar (the c subscript) and the geometrical center of the pillar (the p subscript), the number of points in the pillar and the number of pillars, respectively.

The second part is to extract the deep features of V_p by the following processes.

$$\mathcal{H}_{inf}^{(j)} = \max_{1 \le i \le N} (Dense_{inf}(\mathcal{F}_{inf}^{(j)})), j = 1, \ldots, n \tag{5}$$

$$\mathcal{H}_{veh}^{(j)} = \max_{1 \le i \le N} (Dense_{veh}(\mathcal{F}_{veh}^{(j)})), j = 1, \ldots, m \tag{6}$$

$$\mathcal{F}_{inf}^{(j)} = \{\{V_p^{(i)}\}_{i=1}^{P_j}\}_{j=1}^{n} \tag{7}$$

$$\mathcal{F}_{veh}^{(j)} = \{\{V_p^{(i)}\}_{i=1}^{P_j}\}_{j=1}^{m} \tag{8}$$

where $\mathcal{H}_{inf}^{(j)}$ and $\mathcal{H}_{veh}^{(j)}$ present the feature output from the LFE module which have the shape of (P_j, C). Two decoupled MLP networks are designed as $Dense_{inf}(\cdot)$ and $Dense_{veh}(\cdot)$ to extract V_p from $D-$dimensional to $C-$dimensional. $\mathcal{F}_{inf}^{(j)}$ and $\mathcal{F}_{veh}^{(j)}$ present the feature data from $j-$th infrastructure/vehicle node.

3.4. Two-stream fusion

In this section, to combine the features from scalable and heterogeneous PNs, a specific feature fusion model, called *TSF*, is proposed. Since the roadside sensors and onboard sensors have different locations and poses, the point cloud distributions vary based on different sensor configurations. Although all those data can be transformed into a unified coordinate, the differences in data distribution still affect the fusion performance [20,22]. Specifically, TSF consists of two main contributors, which are named *VI (**V**ehicle-**I**nfrastructure) Stream* and *VI Fusion*, respectively. The whole fusion process can be formulated as:

$$\mathcal{S}_{inf} = \max_{1 \le j \le n} (Concat(\{\mathcal{M}(\mathcal{H}_{inf}^{(j)})\}_{j=1}^{n})) \tag{9}$$

$$\mathcal{S}_{inf} = \max_{1 \le j \le m} (Concat(\{\mathcal{M}(\mathcal{H}_{veh}^{(j)})\}_{j=1}^{m})) \tag{10}$$

$$\mathcal{H}_{cp} = C(Concat(\mathcal{S}_{inf}, \mathcal{S}_{inf})) \tag{11}$$

where $\mathcal{M}(\cdot)$ is the designed mapping function to generate a 2D-pseudo-feature map from the LFE inputs; $Concat(\cdot)$ is the concatenation function to aggregate feature map H in each stream; $\mathcal{S}_{inf}$ and $\mathcal{S}_{inf}$ are the fused feature from each VI stream with a shape of $(1, C, H, W)$; $\mathcal{H}_{cp}$ represents the output of TSF after a concatenation layer $Concat(\cdot)$ and a convolution layer $C(\cdot)$.

Fig. 3 is provided to further explain the TSF process. To consider the heterogeneity of features from different kinds of PNs, the core ideology of TSF is to create two separate feature streams to decouple the feature extraction and fusion within each class of PNs. For instance, as shown in, two dedicated LFEs are designed for extracting pillar features based on the data from vehicles and infrastructures, respectively. A regrouping process is applied to grouping those pillar features into two groups.

A feature mapping process is implemented for each group of features to generate a 2D spatial feature map for each PN in this group, e.g., feature tensor with the shape of (n, C, H, W) for the infrastructure stream. Based on the previous GPP process, the feature maps from different PNs are spatially aligned with the global coordinate already. Then, a fusion process is designed to fuse features from different nodes while keeping their heterogeneity. At the end of the VI stream, features from each stream will be extracted into a $(1, C, H, W)$ tensor. The fused features for each stream are then fused using VI fusion to form a $(2C, H, W)$ feature tensor. Finally, a CNN fuses the $(2C, H, W)$ tensor into the shape of (C, H, W) to keep a consistent spatial shape of output.

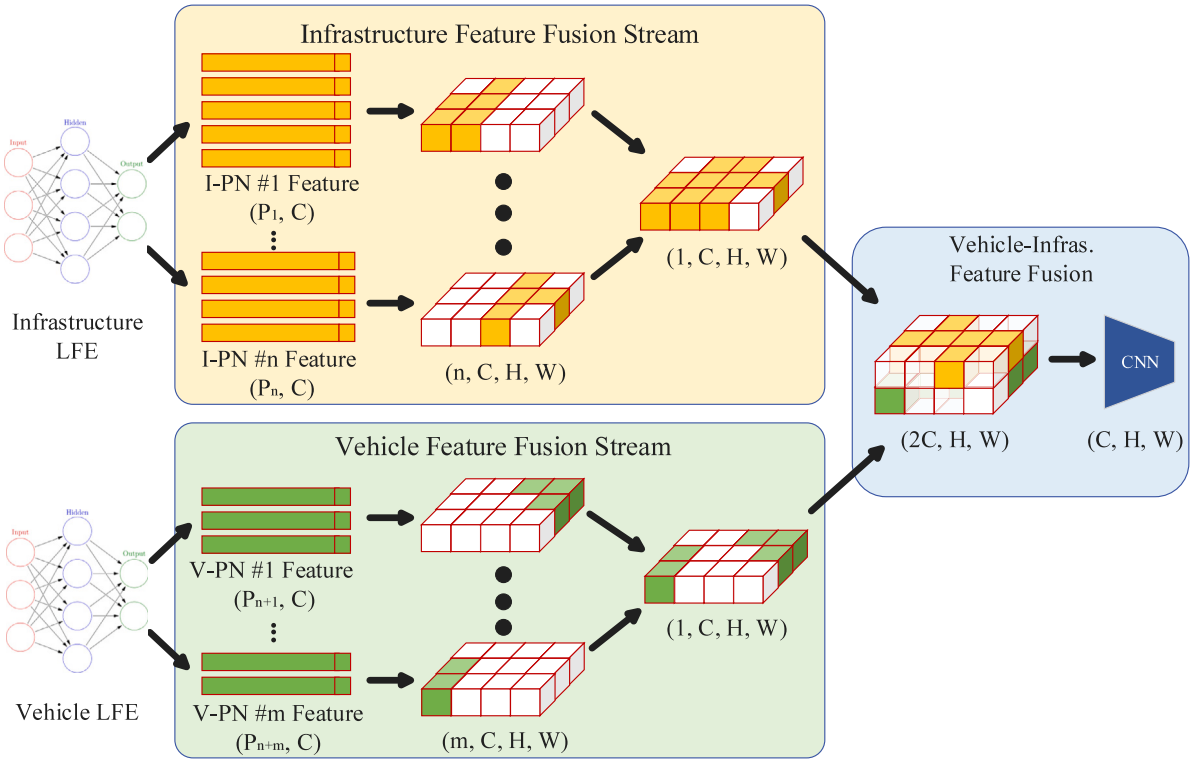


Fig. 3. Illustration of the Two-Stream Fusion process.

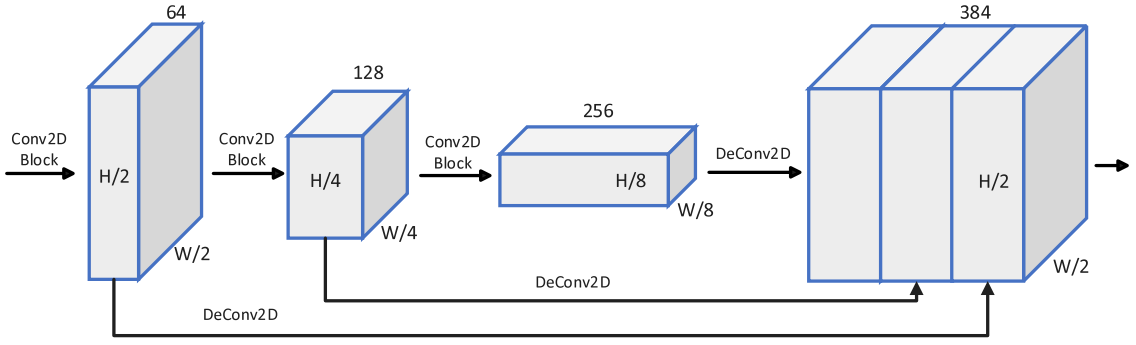


Fig. 4. Illustration of the CFB structure which consists of convolutional and deconvolutional blocks.

3.5. Central feature backbone

Based on the centralized design, VINet only needs one CNN backbone, named Central Feature Backbone (CFB), for the whole CP system for all PNs. To make a fair comparison with others, a classic Region Proposal Network (RPN) [30] is implemented as the CFB in VINet. It is noted that from a theoretical standpoint, since the central PN is equipped with CCU, CFB can be a more powerful backbone to further squeeze the performance of fused features (see Figs. 4 and 7).

Shown in Fig. 4, the CFB consists of three phases: (1) *Downsample CNN*, which consists of several 2D CNN blocks (*Conv2D*) to extract the features in a spatial downsampling manner, (2) *Upsample CNN*, which consists of several deconvolutional blocks (*DeConv2D*) to extract the features in a spatially upsampling manner, and (3) *Multi-scale fusion*, which concatenates the output of each *DeConv2D* to gain multi-scale feature for each spatial location.

Table 1
Parameter configuration and description for the 3D-LiDAR sensors deployed in the CARTI platform.

Parameters	Setting (onboard/roadside)	Description
Channels	64	Number of lasers.
Height	1.74/4.74 m	Height above the ground.
Range	100.0 m	Maximum distance to measure/ray-cast in meters.
Frequency	10.0 Hz	LiDAR rotation frequency.
Upper FOV	+22.5/ + 0	Angle in degrees of the highest laser beam.
Lower FOV	−22.5/ − 22.5	Angle in degrees of the lowest laser beam.
Reflection rate	0.004	Coefficient that measures the LiDAR intensity loss.
Noise stdev	0.01	The standard deviation of the noise model of points.
Dropoff rate	45%	General proportion of points that are randomly dropped.
Dropoff intensity	0.8	The threshold of intensity value for exempting dropoff.
Dropoff zero intensity	40%	The probability value of dropoff for zero-intensity points.

3.6. 3D detection head

To generate 3D rotated bounding boxes, an anchor-based 3D dense head [32] is applied for all the models. Intersection over Union (IoU) is implemented to match the prior bounding boxes with the ground truth. Additionally, both pedestrians and vehicles are considered.

4. Experiments

4.1. Dataset acquisition

Considering the requirement of cooperative perception (CP) research, a platform is needed for customized dataset acquisition. Thus, a CP platform has been designed and developed to support CP model training and validation, such as sensor data collecting and ground truth labeling. We build a CP platform on top of the CARLA simulator [48] for data collection. In addition, to ease the re-usability and accessibility, we organize the new dataset to be compatible with existing open-source training platforms (e.g., OpenMMLab [57]), by following the KITTI's format [58]. Nonetheless, the KITTI's settings and the CARLA have different coordinate systems, so a conversion is used to generate the ground truth labels in the KITTI coordinates.

All the sensors are synchronized based on CARLA and the simulation is running at 10 Hz while the data frame is collected at 2 Hz to increase the variety of the whole dataset (i.e., same strategy as NuScenes [59]). Literally, from our CP platform, unlimited frames of data can be collected with nearly zero cost. In this paper, for speeding up the experimental process, 8695 frames of 3D point clouds are collected, including 4347 frames for training, 1449 frames for evaluation, and 2899 frames for testing. Within each frame, there would be 2 to 7 perception nodes, and a total of 193,263 car objects and 153,077 pedestrian objects are collected. However, it is noted that some of those objects may not be detected by the LiDAR sensors. Therefore, before training and testing, our customized codebase will filter out the objects based on the minimum number of points reflected by the objects, which avoids out-of-sight ground truth labels.

Furthermore, the LiDAR settings for onboard and roadside are different. Detailed specifications of those Lidar sensors are described in Table 1. The main differences lie in the heights and FOVs.

4.2. Experimental setup

For the purpose of evaluating and investigating the CP performance of leveraging multiple PNs, the CP scenario (shown in Fig. 1) is set as a two-adjacent intersection area with 280 m × 80 m, which is much larger than a single-node perception area. Specifically, in Fig. 1, the western infrastructure is defined as the I-PN-C whose coordinate is also set as the reference of the global coordinate. Thus, for GPP, the specific perception field is in the range of $x \in [-53.76\ \text{m}, 181.76\ \text{m}]$ and $y \in [-48.6\ \text{m}, 41\ \text{m}]$. For LFE, both PCD from the roadside LiDARs and onboard LiDARs are transformed into pillars with the voxel size of $[0.23\ \text{m}, 0.23\ \text{m}, 4.0\ \text{m}]$ in this paper, the maximum number of pillars for each node is set as $[25,000, 15,000]$ for training and testing. For CFB, each Conv2D block consists of one Conv2D layer with the kernel of $(3, 2, 1)$, followed by several Conv2D layers with kernels of $(3, 1, 1)$. Specifically, the numbers of Conv2D layers in each block are 4, 6, and 6, respectively. For 3DH, the number of object classes is set to be 2, i.e., car and pedestrian.

4.2.1. Training details

The training and testing platform is equipped with Intel® Core™ i7-10700K CPU@3.80GHz×16 and NVIDIA GPU@GeForce RTX 3090. The training pipeline is designed with 160 epochs with *Batchsize* of 2. During training, a data sample strategy is applied by filtering the ground target by minimum points (MP) reflected by LiDAR. Specifically, MP is set as 5 and 10 for car and pedestrian objects, respectively.

Table 2
AP results on the CARTI dataset using the BEV detection benchmark.

Model	Feature type	Overall AP	Pedestrians AP@0.25			Car AP@0.7		
			MP≥10	MP≥5	MP≥1	MP≥10	MP≥5	MP≥1
Early Fusion	Raw PCD	35.20	7.23	9.39	10.47	61.79	61.57	60.74
F-Cooper	Dense CNN	27.46	4.38	11.63	12.34	46.35	46.34	43.71
PillarGrid	Pillar Feature	39.48	7.49	10.81	12.87	70.42	69.13	66.17
VINet	Pillar Feature	**41.49**	**8.21**	**15.48**	**18.07**	**70.83**	**70.11**	**66.22**

Table 3
AP results on the CARTI dataset using the 3D detection benchmark.

Model	Feature type	Overall AP	Pedestrians AP@0.25			Car AP@0.7		
			MP≥10	MP≥5	MP≥1	MP≥10	MP≥5	MP≥1
Early Fusion	Raw PCD	30.26	6.94	8.56	9.34	52.39	52.27	52.08
F-Cooper	Dense CNN	24.37	4.28	11.42	12.15	40.49	39.91	37.96
PillarGrid	Pillar Feature	35.90	6.58	10.09	12.16	62.28	62.21	62.06
VINet	Pillar Feature	**37.81**	**7.58**	**14.83**	**17.16**	**62.51**	**62.44**	**62.31**

4.2.2. Evaluation matrix

- **True Positive (TP)**: the number of cases predicted as positive by the classifier when they are indeed positive, i.e., a vehicle object is detected as a vehicle.
- **False Positive (FP)** = the number of cases predicted as positive by the classifier when they are indeed negative, i.e., a non-vehicle object is detected as a vehicle.
- **True Negative (TN)** = the number of cases predicted as negative by the classifier when they are indeed negative, i.e., a non-vehicle object is detected as a non-vehicle object.
- **False Negative (FN)** = the number of cases predicted as negative by the classifier when they are indeed positive, i.e., a vehicle is detected as a non-vehicle object.

Precision is the ability of the detector to identify only relevant objects, i.e., vehicles and pedestrians in this paper. It is the proportion of correct positive predictions and is given by

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\#\text{ of all detections}} \tag{12}$$

Recall is a metric that measures the ability of the detector to find all the relevant cases (that is, all the ground truths). It is the proportion of TP detected among all ground truth (i.e., real vehicles) and is defined as

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\#\text{ of all ground truth}} \tag{13}$$

The detection performance is measured with Average Precision (AP) and Average Recall (AR) at Intersection-over-Union (IoU) thresholds of 0.25 for pedestrians and 0.7 for cars, respectively. Furthermore, based on the MP reflected by the ground target, each evaluation class is further divided into three categories: MP≥10, MP≥5, and MP≥1, respectively, to investigate the performance of CP methods on different difficulty levels.

4.2.3. Compared baselines

Three baselines are selected from different perspectives: (1) **Raw PCD fusion** – *EarlyFusion* based on *PointPillar* [32] is considered as it is one of the most straightforward CP methods; (2) **Dense CNN-feature fusion** – *F-Cooper* [19] is considered since it provides a classic fusion scheme by fusing the feature map generated by the CNN backbone; and (3) **Pillar feature fusion** – our last baseline is PillarGrid [20], the first CP method that uses pillar features for fusion. To make a fair comparison, all methods are trained based on the same CNN backbone and 3D detection head with the same training epochs. All the other data sampling and augmentation techniques are set the same for all methods.

4.3. Quantitative evaluation

4.3.1. Object detection

The numerical testing results are illustrated in Tables 2 and 3. Two evaluation benchmarks are involved: the BEV detection benchmark and the 3D detection benchmark. Table 2 demonstrates the performance on the BEV benchmark. VINet dominates all the compared methods. Specifically, VINet can improve the overall mAP by +6.29%, +14.03%, and +2.01% compared with Raw PCD fusion [32], Dense CNN-feature fusion [19], and pillar-feature fusion [20]. Furthermore, for general pedestrian detection (i.e., $MP \geq 1$), VINet significantly improves the mAP by over +5% for all other methods.

For 3D detection results shown in Table 3, VINet can offer the mAP improvements by +7.55%, +13.44%, +1.91% compared with EarlyFusion, F-Cooper, and PillarGrid respectively. Additionally, for car detection, the pillar feature-based methods can bring tremendous performance improvement, such as 10 + % mAP than early fusion and 20 + % mAP than Dense CNN-based fusion.

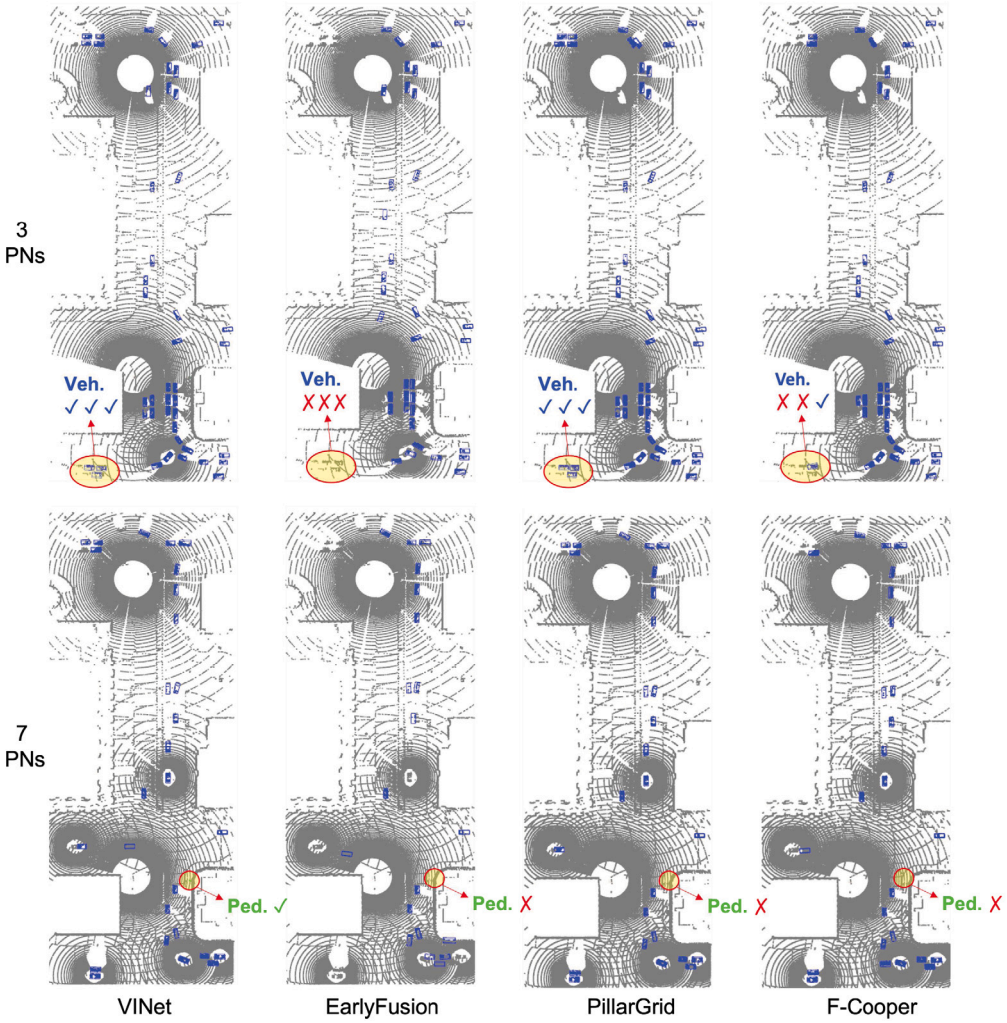


Fig. 5. Visualization results for different methods under different numbers of PNs (top row: 3 PNs, bottom row: 7 PNs).

4.3.2. Running time performance

Table 8 also demonstrates the computing speed of the real-time inference pipeline. Both VINet and EarlyFusion can process over 10 frames per second, while F-Cooper can only get 8 frames. As a LiDAR rotating speed can be set to 10 Hz, VINet can achieve a real-time inference capacity without considering other time-consuming processes, such as transmission delay.

4.4. Qualitative results

Fig. 5 shows the visualization CP results of all the methods in the same data frame under different numbers of PNs.

4.4.1. Object detection

From the perspective of vehicle detection, both VINet and PillarGrid surpass the EarlyFusion and F-Cooper. It is also demonstrated that pillar feature-based cooperation can achieve better performance compared with raw point-based fusion and dense feature-based fusion. In terms of pedestrian detection, for better illustration, we choose a pedestrian located on the lower-left part of each scene. Results show that VINet is the only method that can successfully detect this pedestrian, which demonstrates that TSF has a better capability of extracting and fusing features for small objects (it is noted that there were other pedestrians in the scene which are not highlighted for succinct illustration).

4.4.2. Various number of PNs

Regarding the number of PNs, when it grows from 3 to 7, feature-based methods, e.g., VINet, PillarGrid, and F-Cooper, can maintain their performance, while EarlyFusion has an evident performance drop. One hypothetical reason is when a single object is

Table 4
Component ablation study for: (i) VI Stream and (ii) VI Fusion under 3D benchmark with MP≥1.

VI Stream	VI Fusion	Pedestrian	Vehicle
×	×	9.73	22.15
✓	×	14.33	61.41
×	✓	12.16	62.07
✓	✓	**17.16**	**62.31**

perceived by multiple sensors, feature data is more consistent than raw PCD. For instance, if a car is perceived by two LiDARs (one from the front while another from the back), the raw PCD information reflected by this car from those two LiDARs has a bigger difference than the deep feature extracted from the neural networks. In other words, since the neural nets, especially CNN, act like a filtering process, we can infer that feature data from single objects tend to have a smaller divergence than raw PCD. Thus, feature data is also more suitable for large-scale CP systems than raw data.

4.5. Ablation study

In this section, we provide ablation studies and analyze the key design of VINet. From Table 4, both the VI stream and VI fusion contribute to the detection performance for pedestrians and vehicles. For the deprecated model in which the VI Stream design was removed, the parameters for neural networks used for extracting the sharing data for vehicle and infrastructure nodes are shared with each other. Additionally, to remove the VI Fusion model, we replaced the *concatenated* operation followed by a *convolution* layer with the *maxout* operation to fuse the feature map from the spatial size of $(2, C, H, W)$ to (C, H, W).

4.5.1. Two stream structure

A noticeable improvement can be observed by checking the first and second rows, which demonstrates the effectiveness of the design of VI Stream. However, if comparing the third and fourth rows, it can be concluded that the VI Stream tends to have a more significant impact on the pedestrian class (AP +5.00%) than on the vehicle class (AP +0.24%). One possible explanation for this would be that by using VI Stream (i.e., applying two decoupled neural networks) the feature extractor can better fit the feature distribution of one specific class (e.g., pedestrian) without the impact of other types of objects (e.g., vehicle). Without the VI Stream, i.e., using one extractor for both classes, due to the significant difference in the spatial sizes between vehicles (e.g., around $2.5\ \mathrm{m} \times 4.5\ \mathrm{m}$) and pedestrians (e.g., around $0.5\ \mathrm{m} \times 0.5\ \mathrm{m}$), the feature from vehicle objects would have stronger dominance than features from pedestrian objects. Thus, by separating the feature extractor for different classes, the feature from pedestrians would be improved more significantly than vehicles.

4.5.2. Feature fusion scheme

Without the TSF structure, the performance will drop drastically for vehicles, i.e., AP −40.16%, and moderately for pedestrians, i.e. AP −7.43%. This indicates that the TSF plays a vital role in feature fusion. A hypothesis is that vehicles are much larger than pedestrians, and therefore their detection can benefit more from the fused features than only from the single feature.

5. System-level cost analysis

In this section, we will comprehensively analyze the computational cost and communication cost from the standpoint of system-level application. Based on the number of nodes that benefited from the cooperative perception system, we categorize cooperative perception into two types:

- Egocentric CP: only one perception node (the ego node) is performing cooperative perception while other nodes only act as the subsidiary to provide feature information to benefit the ego-vehicle. Most of the existing CP work belongs to this category [19,21,52].
- Holistic CP: all perception nodes are performing cooperative perception which means every node will not only share their feature data but also receive the cooperative perception information.

From the perspective of real-world implementation, it may be more beneficial to implement the Holistic CP system in the real world. However, the computing complexity of a Holistic CP system would be extremely high if no specific optimization is considered for the perception structure. We evaluate VINet by comparing it with multiple different baselines as well as under different cooperative perception conditions. Both theoretical and experimental analyses are conducted and shown below.

Table 5
GFLOPs analysis under different cooperative perception conditions.

Model	GFLOPs	VINet(Ours)	EarlyFusion	LateFusion	F-Cooper	PillarGrid
Encoder	No cooperation	19.88	0.55	0.55	0.55	0.55
	Egocentric CP w/ 10 PN	198.80	5.50	5.50	5.50	5.50
	Holistic CP w/ 10 PN	198.80	5.50	5.50	5.50	5.50
Backbone	No cooperation	270.58	289.91	289.91	289.91	289.91
	Egocentric CP w/ 10 PN	270.58	289.91	2,899.10	2,899.10	2,899.10
	Holistic CP w/ 10 PN	270.58	2,899.10	2,899.10	2,899.10	2,899.10
Head	No cooperation	4.83	4.83	4.83	4.83	4.83
	Egocentric CP w/ 10 PN	4.83	4.83	48.30	4.83	4.83
	Holistic CP w/ 10 PN	4.83	48.30	48.30	48.30	48.30
Overall	No cooperation	295.29	295.29	295.29	295.29	295.29
	Egocentric CP w/ 10 PN	474.21	**295.29**	2,952.90	2,909.43	2,909.43
	Holistic CP w/ 10 PN	**474.21**	2,952.90	2,952.90	2,952.90	2,952.90

Table 6
Computational complexity under different cooperative perception conditions.

Model	Complexity	VINet(Ours)	EarlyFusion	LateFusion	F-Cooper	PillarGrid
Encoder	Egocentric Coop. w/ 10 PN	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
	Holistic Coop. w/ 10 PN	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Backbone	Egocentric Coop. w/ 10 PN	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	$\mathcal{O}(K)$
	Holistic Coop. w/ 10 PN	$\mathcal{O}(1)$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	$\mathcal{O}(K)$
Head	Egocentric Coop. w/ 10 PN	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$O(M)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
	Holistic Coop. w/ 10 PN	$\mathcal{O}(1)$	$O(M)$	$O(M)$	$O(M)$	$O(M)$
Overall	Egocentric Coop. w/ 10 PN	$\underline{\mathcal{O}(N)}$	$\underline{\mathcal{O}(N)}$	$\mathcal{O}(N+K+M)$	$\mathcal{O}(N+K)$	$\mathcal{O}(N+K)$
	Holistic Coop. w/ 10 PN	$\underline{\mathcal{O}(N)}$	$\mathcal{O}(N+K+M)$	$\mathcal{O}(N+K+M)$	$\mathcal{O}(N+K+M)$	$\mathcal{O}(N+K+M)$

5.1. Theoretical cost analysis

5.1.1. Computational complexity

To analyze the computational complexity of a model, the total number of computations the model has to perform is an important factor. In this paper, we adopt the Floating Point Operations (FLOPs), which apply to any type of computing operations that involve a floating point value. Generally, the more FLOPs the model has to perform, the more complex the model is. In addition to FLOPs, we also involve Multiply-Accumulate Computations (MACs) to illustrate the computational complexity by showing the algebraic expression with respect to the number of perception nodes (PNs).

To calculate the overall FLOPs of a model, convolution layers and fully connected layers are mainly considered, which can be calculated by the following equations:

$$\mathcal{F}(conv) = 2 \times \mathcal{C} \times \mathcal{S} \times \mathcal{O} \tag{14}$$

$$\mathcal{F}(linear) = 2 \times \mathcal{I} \times \mathcal{O} \tag{15}$$

where $\mathcal{C}$, $\mathcal{S}$, $\mathcal{I}$, and $\mathcal{O}$ represent the number of channels, kernel shape, input shape, and output shape, respectively.

The computational complexity for different models is calculated and shown in Tables 5 and 6. Table 5 shows the GFLOPs (one billion FLOPs) for each model under three different perception conditions: (1) no cooperation, (2) egocentric cooperative perception with 10 perception nodes, and (3) holistic cooperative perception with 10 perception nodes.

Under the Egocentric CP conditions, EarlyFusion achieves the lowest GFLOPs and has an $\mathcal{O}(N)$ complexity. With only one node required for computation, EarlyFusion can be regarded as the lower bound of the computational complexity for Egocentric CP systems. However, as discussed early in Section 5, the Egocentric CP system is too ideal to be implemented in real-world scenarios. Thus, looking into the performance under Holistic CP conditions provides a more valuable assessment for real-world cases. Under the Holistic CP conditions, VINet can reduce the GFLOPs by a large margin. Numerically, VINet can reduce 83.9% computational load compared with other methods.

In addition, Table 6 demonstrates the computational complexity with theoretically concise notations. Our method can achieve constant complexity for the backbone and head and linear complexity for the encoder. On the other hand, most of the compared methods have a linear complexity for the backbone module, i.e. $\mathcal{O}(K)$, which extremely increases the total GFLOPs of the model.

5.1.2. Communication complexity

To theoretically analyze the communication complexity, megabytes (MB) can be used for evaluating the data size that needed to be transmitted via communication. Since metadata transmission requires much less communication bandwidth than feature data

Table 7
Communication complexity under different cooperative perception conditions.

MB	VINet	EarlyFusion	F-Cooper	PillarGrid
Single Transmission	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Egocentric Coop. w/ 10 PNs	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Holistic Coop. w/ 10 PNs	$\underline{\mathcal{O}(N)}$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$

transmission, in this study, the system communication cost is mainly estimated based on feature data transmission. VINet is the first CP method that offers linear communication complexity.

Therefore, the communication complexity can be calculated as:

$$\begin{aligned} \mathcal{M}_{vinet}(N) &= N \times m_{vinet} \\ \mathcal{M}_{early}(N) &= N \times (N-1) \times m_{early} \\ \mathcal{M}_{dense}(N) &= N \times (N-1) \times m_{dense} \end{aligned} \tag{16}$$

where m_{vinet}, m_{early}, and m_{dense} represent the bandwidth cost per channel for VINet, EarlyFusion, and F-Cooper, respectively. Specifically, m_{early} is based on the statement in [60], while m_{vinet} and m_{dense} are calculated based on their feature size. Thus, the communication complexity for different methods can be summarized in Table 7. Getting benefit from the global CP structure, VINet is the only method that can achieve linear complexity for communication under Holistic CP conditions.

5.2. Experimental cost analysis

In this paper, we also conduct real-world experiments to analyze the system-level cost. Since the capacity of GPU memory and communication bandwidth cost are crucial factors for the real-world implementation of the deep network-based method and cooperative perception, respectively, these two factors are analyzed by real-world evaluation and estimation.

5.2.1. GPU cost estimation

In this study, we use GPU memory consumption as a surrogate and propose a polynomial model to estimate the system-wide computational costs with limited computational power, in a mathematical manner.

We can start the analysis from the single-node perception process, which can be divided into three main components in terms of the computing process: (1) the voxelization and pillar encoder, (2) the pillar scatters and CNN backbone and (3) the detection head. We denote the computational costs of these computing processes to be $\mathcal{E}$, $\mathcal{B}$, and $\mathcal{D}$, respectively. Then a polynomial model for single-node perception can be designed as:

$$\mathcal{E} + \mathcal{B} + \mathcal{D} = \mathcal{C}_{single} \tag{17}$$

According to Table 11, the VINet computing process can be divided into three phases: (1) GPP + LFE on LCU, (2) TSF + CFB on CCU, and (3) 3DH on CCU. For CP system with N nodes, the computational cost can be defined as:

$$N \times \mathcal{E} + \mathcal{B} + \mathcal{D} = \mathcal{C}_{vinet} \tag{18}$$

Furthermore, for the dense-feature-based CP methods [19], the fusion happens after the CNN backbone. So their computational cost can be defined as:

$$N \times (\mathcal{E} + \mathcal{B}) + \mathcal{D} = \mathcal{C}_{dense} \tag{19}$$

Both early fusion and late fusion need $\mathcal{C}_{single}$ for each perception node, thus their computational cost can be defined as:

$$N \times (\mathcal{E} + \mathcal{B} + D) = \mathcal{C}_{early/late} \tag{20}$$

For the CARTI dataset in this paper, we have 2 to 7 perception nodes collected randomly. Here we assume an average number of PNs, i.e., $N = 4$ to process the testing results for GPU memory cost. During the inference with *batchsize* of 1, $\tilde{\mathcal{C}}_{early}$, $\tilde{\mathcal{C}}_{vinet}$ and $\tilde{\mathcal{C}}_{dense}$ are measured as 12.52, 5.60, and 10.95, respectively (in GB). Therefore, by solving the aforementioned polynomial equations, we will get:

$$\begin{aligned} \mathcal{E} &= 0.823 \\ \mathcal{B} &= 1.783 \\ D &= 0.520 \end{aligned} \tag{21}$$

Thus, we can get the polynomial representation of the system-wide cost estimation for the models mentioned above:

$$\begin{aligned} \mathcal{C}_{single}(N) &= 3.13 \\ \mathcal{C}_{vinet}(N) &= 0.82 \times N + 2.30 \\ \mathcal{C}_{dense}(N) &= 2.61 \times N + 0.52 \\ \mathcal{C}_{early/late}(N) &= 3.13 \times N \end{aligned} \tag{22}$$

Table 8
GPU memory cost analysis under different cooperative perception conditions.

GPU memo. [GB]	Shallow feature(Ours)	Early feature	Dense feature	Late feature
No Cooperation	3.13	3.13	3.13	3.13
Egocentric Coop. Avg.	5.60	3.13	10.85	12.52
Holistic Coop. Avg. Est.	**5.60**	12.52	12.52	12.52
Egocentric Coop. w/10PN Est.	10.50	3.13	26.52	31.30
Holistic Coop. w/10PN Est.	**10.50**	31.30	31.30	31.30

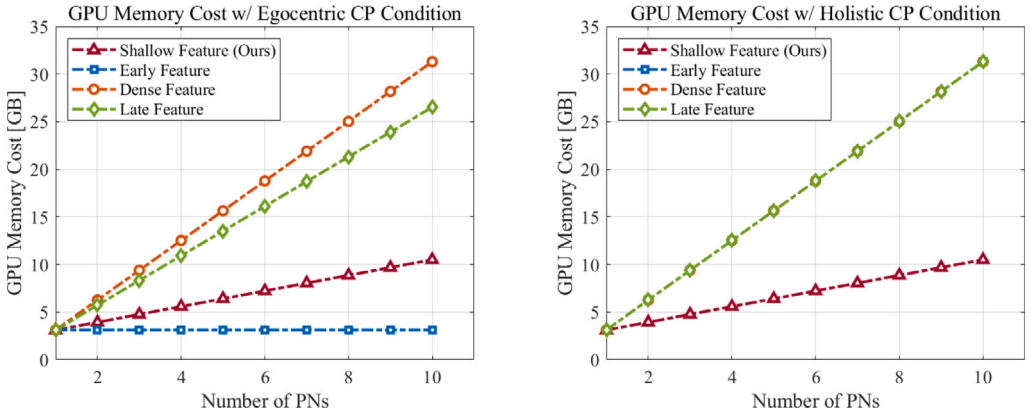


Fig. 6. GPU memory cost under different CP condition.

Table 9
Bandwidth requirement analysis under different cooperative perception conditions.

MB	Shallow feature (Ours)	Early feature	Dense feature	Late feature
Single Transmission	3.84	6.00	200/6.87[a]	0.025
Egocentric CP w/ 10 PN	34.56	54.00	61.83	0.225
Holistic CP w/ 10 PN	34.56	540.00	618.30	2.25

[a] For dense-feature sharing, data compression is generally necessary to make data transmission possible. Here a 32× compression rate is assumed to F-Cooper for communication cost analysis, based on other dense-feature sharing studies, e.g., [22].

We use the inference test with *Batchsize*= 1 which can be regarded as the way they work in the real world. Since the CARTI dataset contains 2 to 7 PNs, we use the average system-wide cost for the CARTI dataset testing results as shown in Table 8. The estimation at 10 PNs is also listed in Table 8.

In Table 8, we use *Shallow Feature* – the lightweight feature from the Encoder – to distinguish our methods with *Dense Feature* – the normally used deep feature from the Backbone [19,22]. Under Egocentric CP conditions, the GPU memory consumption of shallow feature-based methods is 48.4% and 55.3% lower than the dense feature-based and late feature-based methods on average. If 10 PNs are involved in the Egocentric CP system, our method can reduce GPU memory consumption by 60.4% and 66.5% with respect to dense feature-based and late feature-based methods, respectively. Under Holistic CP conditions, our method can reduce 66.5% GPU memory consumption compared with all the other methods.

In addition, Fig. 6 demonstrates the change in GPU Memory Cost with different PNs involved in the CP system. It is shown that shallow features are very efficient in terms of GPU memory consumption. Although the early feature requires the lowest GPU memory under Egocentric CP conditions, its GPU cost is the same as the late feature and dense feature due to its fundamental system structure.

5.2.2. Bandwidth cost estimation

For bandwidth cost, we estimate the data size of the feature needed to be transmitted. Specifically, the size for feature per transmission can be roughly formulated using the following equation:

$$m = \frac{f \times c \times b}{1,000,000} \tag{23}$$

where f, c, and b represent the number of feature grids, the number of channels in each feature grid, and the number of bytes for each data point (4 for *float32* type used in this paper), respectively. Based on the communication complexity (see Eq. (16)), the bandwidth requirement analysis is shown in Table 9.

Shown in Fig. 7, from the perspective of feature size, the shallow feature can save 35.7% data space with respect to early features, and 44.1% data space with respect to a compressed dense feature. Similar bandwidth reduction can be achieved under egocentric

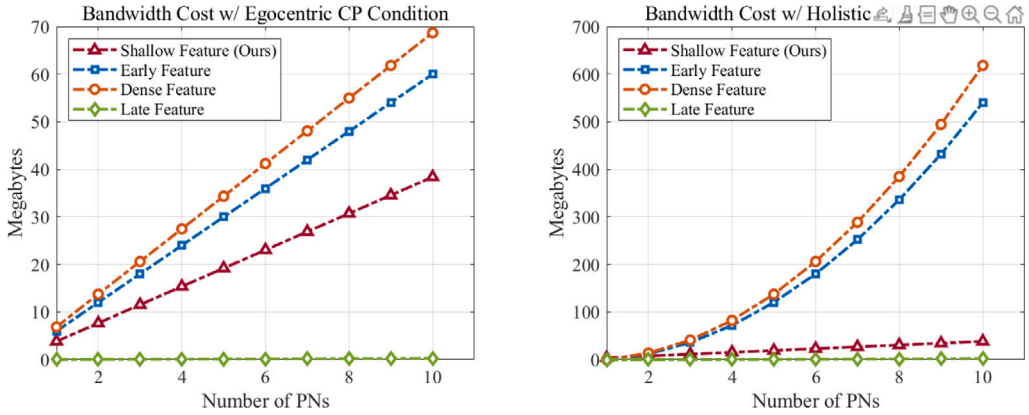


Fig. 7. Bandwidth cost under different CP condition.

CP conditions. Under Holistic CP conditions, our method can reduce 93.6% and 94.4% bandwidth requirements compared with early feature-based and dense feature-based methods (see Table 9).

6. Conclusions

In this paper, we propose VINet, the first cooperative object detection method designed from the standpoint of large-scale system-level implementation. We demonstrate that for lightweight, scalable, and heterogeneous cooperative 3D object detection tasks, VINet can enhance overall mAP by 1.7%–18.9% for car and pedestrian classes, while reducing system-wide computational costs by 84% and communication costs by 94%. Furthermore, VINet also offers a unified framework for cooperative object detection with low system-wide costs. For future work, data compression, impacts of noise levels, and latency will be further investigated to evaluate the performance of VINet under more challenging environments.

6.1. Limitations and future work

The most prominent limitation of the proposed system is the absence of latency and synchronization analyses, which can be easily overcome in a simulation environment but are inevitable in real-world situations. An exciting direction for future work would be to design a more comprehensive CP framework that can handle cooperative perception under more realistic conditions (e.g., the presence of communication latency and asynchronous/heterogeneous issues). Additionally, the proposed centralized system structure enables a significant cost reduction in both computation and communication, but limited V2V communication can be beneficial for the system and thus should be considered rather than purely removed. More practically, reducing the error and noise for self-pose estimation plays a vital role in implementing cooperative perception in real-world conditions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was funded by Toyota Motor North America, InfoTech Labs. The contents of this paper reflect the views of the authors only, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views of Toyota Motor North America.

References

[1] U.S. Department of Transportation, Overview of motor vehicle crashes in 2019, 2020, Available: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/813060.

[2] INRIX, INRIX: Congestion costs each American 97 hours, $1,348 a year, 2018, Available: https://inrix.com/press-releases/scorecard-2018-us/.

[3] US Department of Energy, FOTW #1204: Fuel wasted due to U.S. traffic congestion in 2020 cut in half from 2019 to 2020, 2021, Available: https://www.energy.gov/eere/vehicles/articles/fotw-1204-sept-20-2021-fuel-wasted-due-us-traffic-congestion-2020-cut-half.

[4] D.J. Fagnant, K. Kockelman, Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations, Transp. Res. A 77 (2015) 167–181.

[5] Society of Automotive Engineers (SAE), Taxonomy and definitions for terms related to cooperative driving automation for on-road motor vehicles, 2021, Available: https://www.sae.org/standards/content/j3216_202107.

[6] J. Wu, H. Xu, Y. Zhang, R. Sun, An improved vehicle-pedestrian near-crash identification method with a roadside LiDAR sensor, J. Saf. Res. 73 (2020) 211–224.

[7] Z. Bai, P. Hao, W. Shangguan, B. Cai, M.J. Barth, Hybrid reinforcement learning-based eco-driving strategy for connected and automated vehicles at signalized intersections, IEEE Trans. Intell. Transp. Syst. (2022) 1–14, http://dx.doi.org/10.1109/TITS.2022.3145798.

[8] Z. Wang, Y. Bian, S.E. Shladover, G. Wu, S.E. Li, M.J. Barth, A survey on cooperative longitudinal motion control of multiple connected and automated vehicles, IEEE Intell. Transp. Syst. Mag. 12 (1) (2020) 4–24, http://dx.doi.org/10.1109/MITS.2019.2953562.

[9] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, M. Pietikäinen, Deep learning for generic object detection: A survey, Int. J. Comput. Vis. 128 (2) (2020) 261–318.

[10] E. Arnold, O.Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, A. Mouzakitis, A survey on 3D object detection methods for autonomous driving applications, IEEE Trans. Intell. Transp. Syst. 20 (10) (2019) 3782–3795.

[11] A. Manjunath, Y. Liu, B. Henriques, A. Engstle, Radar based object detection and tracking for autonomous driving, in: 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, ICMIM, 2018, pp. 1–4, http://dx.doi.org/10.1109/ICMIM.2018.8443497.

[12] E. Yurtsever, J. Lambert, A. Carballo, K. Takeda, A survey of autonomous driving: Common practices and emerging technologies, IEEE Access 8 (2020) 58443–58469.

[13] Z. Bai, G. Wu, X. Qi, Y. Liu, K. Oguchi, M.J. Barth, Cyber mobility mirror for enabling cooperative driving automation in mixed traffic: A co-simulation platform, IEEE Intell. Transp. Syst. Mag. 15 (2) (2023) 251–265, http://dx.doi.org/10.1109/MITS.2022.3203662.

[14] Waymo, Introducing the 5th-generation waymo driver: Informed by experience, designed for scale, engineered to tackle more environments, 2022, Available: https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html.

[15] E. Lindholm, J. Nickolls, S. Oberman, J. Montrym, NVIDIA Tesla: A unified graphics and computing architecture, IEEE Micro 28 (2) (2008) 39–55.

[16] Z. Bai, S.P. Nayak, X. Zhao, G. Wu, M.J. Barth, X. Qi, Y. Liu, E.A. Sisbot, K. Oguchi, Cyber mobility mirror: A deep learning-based real-world object perception platform using roadside LiDAR, IEEE Trans. Intell. Transp. Syst. (2023) 1–14, http://dx.doi.org/10.1109/TITS.2023.3268281.

[17] J. Zhao, H. Xu, H. Liu, J. Wu, Y. Zheng, D. Wu, Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors, Transp. Res. C 100 (2019) 68–87.

[18] E. Arnold, M. Dianati, R. de Temple, S. Fallah, Cooperative perception for 3D object detection in driving scenarios using infrastructure sensors, IEEE Trans. Intell. Transp. Syst. (2020).

[19] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, S. Fu, F-Cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, SEC '19, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450367332, 2019, pp. 88–100, http://dx.doi.org/10.1145/3318216.3363300.

[20] Z. Bai, G. Wu, M.J. Barth, Y. Liu, E.A. Sisbot, K. Oguchi, PillarGrid: Deep learning-based cooperative perception for 3D object detection from onboard-roadside LiDAR, in: 2022 IEEE 25th International Conference on Intelligent Transportation Systems, ITSC, 2022, pp. 1743–1749, http://dx.doi.org/10.1109/ITSC55140.2022.9921947.

[21] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, R. Urtasun, V2vnet: Vehicle-to-vehicle communication for joint perception and prediction, in: European Conference on Computer Vision, Springer, 2020, pp. 605–621.

[22] R. Xu, H. Xiang, Z. Tu, X. Xia, M.-H. Yang, J. Ma, V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer, 2022, arXiv preprint arXiv:2203.10638.

[23] Q. Chen, S. Tang, Q. Yang, S. Fu, Cooper: Cooperative perception for connected autonomous vehicles based on 3D point clouds, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2019, pp. 514–524.

[24] S.M. Ahmed, C.M. Chew, Density-based clustering for 3D object detection in point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10608–10617.

[25] Z. Zhang, L. Zhang, X. Tong, P.T. Mathiopoulos, B. Guo, X. Huang, Z. Wang, Y. Wang, A multilevel point-cluster-based discriminative feature for ALS point cloud classification, IEEE Trans. Geosci. Remote Sens. 54 (6) (2016) 3309–3321.

[26] G. Bishop, G. Welch, et al., An introduction to the kalman filter, in: Proc of SIGGRAPH, Course, Vol. 8, no. 27599–23175, 2001, p. 41.

[27] Z. Zhang, J. Zheng, H. Xu, X. Wang, X. Fan, R. Chen, Automatic background construction and object detection based on roadside LiDAR, IEEE Trans. Intell. Transp. Syst. 21 (10) (2019) 4086–4097.

[28] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, Adv. Neural Inf. Process. Syst. 30 (2017).

[29] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, H. Li, PV-RCNN: Point-voxel feature set abstraction for 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10529–10538.

[30] Y. Zhou, O. Tuzel, 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.

[31] Y. Yan, Y. Mao, B. Li, Second: Sparsely embedded convolutional detection, Sensors 18 (10) (2018) 3337.

[32] A.H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, O. Beijbom, 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.

[33] M. Simony, S. Milzy, K. Amendey, H.-M. Gross, Complex-yolo: An euler-region-proposal for real-time 3D object detection on point clouds, in: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018.

[34] B. Yang, W. Luo, R. Urtasun, 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.

[35] H. Lu, X. Chen, G. Zhang, Q. Zhou, Y. Ma, Y. Zhao, SCANet: Spatial-channel attention network for 3D object detection, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2019, pp. 1992–1996.

[36] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, S. Han, BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation, 2022, arXiv preprint arXiv:2205.13542.

[37] Z. Bai, G. Wu, X. Qi, Y. Liu, K. Oguchi, M.J. Barth, Infrastructure-based object detection and tracking for cooperative driving automation: A survey, in: 2022 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2022, pp. 1366–1373.

[38] J. Wu, H. Xu, J. Zheng, Automatic background filtering and lane identification with roadside LiDAR data, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2017, pp. 1–6.
[39] J. Li, J.-h. Cheng, J.-y. Shi, F. Huang, Brief introduction of Back Propagation (BP) neural network algorithm and its improvement, in: Advances in Computer Science and Information Engineering, Springer, 2012, pp. 553–558.
[40] Y. Cui, H. Xu, J. Wu, Y. Sun, J. Zhao, Automatic vehicle tracking with roadside LiDAR data for the connected-vehicles system, IEEE Intell. Syst. 34 (3) (2019) 44–51.
[41] J. Zhang, W. Xiao, B. Coifman, J.P. Mills, Vehicle tracking and speed estimation from roadside LiDAR, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 13 (2020) 5597–5608.
[42] S.J. Julier, J.K. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (3) (2004) 401–422.
[43] Y. Bar-Shalom, F. Daum, J. Huang, The probabilistic data association filter, IEEE Control Syst. Mag. 29 (6) (2009) 82–100.
[44] L. Zhang, J. Zheng, R. Sun, Y. Tao, GC-Net: Gridding and clustering for traffic object detection with roadside LiDAR, IEEE Intell. Syst. (2020).
[45] Y. Song, H. Zhang, Y. Liu, J. Liu, H. Zhang, X. Song, Background filtering and object detection with a stationary LiDAR using a layer-based method, IEEE Access 8 (2020) 184426–184436, http://dx.doi.org/10.1109/ACCESS.2020.3029341.
[46] M. Gouda, B. Arantes de Achilles Mello, K. El-Basyouny, Automated object detection, mapping, and assessment of roadside clear zones using LiDAR data, Transp. Res. Rec. 2675 (12) (2021) 432–448.
[47] Z. Zhang, J. Zheng, X. Wang, X. Fan, Background filtering and vehicle detection with roadside LiDAR based on point association, in: 2018 37th Chinese Control Conference, CCC, 2018, pp. 7938–7943, http://dx.doi.org/10.23919/ChiCC.2018.8484040.
[48] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator, in: Conference on Robot Learning, PMLR, 2017, pp. 1–16.
[49] B. Veeramani, J.W. Raymond, P. Chanda, DeepSort: Deep convolutional networks for sorting haploid maize seeds, BMC Bioinform. 19 (9) (2018) 1–9.
[50] Z. Gong, Z. Wang, B. Zhou, W. Liu, P. Liu, Pedestrian detection method based on roadside light detection and ranging, SAE Int. J. Connect. Autom. Veh. 4 (12-04-04-0031) (2021).
[51] E.E. Marvasti, A. Raftari, A.E. Marvasti, Y.P. Fallah, R. Guo, H. Lu, Cooperative LiDAR object detection via feature sharing in deep networks, in: 2020 IEEE 92nd Vehicular Technology Conference, VTC2020-Fall, IEEE, 2020, pp. 1–7.
[52] R. Xu, H. Xiang, X. Xia, X. Han, J. Liu, J. Ma, OPV2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication, 2021, arXiv preprint arXiv:2109.07644.
[53] X. Xia, Z. Meng, X. Han, H. Li, T. Tsukiji, R. Xu, Z. Zheng, J. Ma, An automated driving systems data acquisition and analytics platform, Transp. Res. C 151 (2023) 104120.
[54] X. Xia, E. Hashemi, L. Xiong, A. Khajepour, Autonomous vehicle kinematics and dynamics synthesis for sideslip angle estimation based on consensus Kalman filter, IEEE Trans. Control Syst. Technol. 31 (1) (2022) 179–192.
[55] X. Xia, L. Xiong, Y. Huang, Y. Lu, L. Gao, N. Xu, Z. Yu, Estimation on IMU yaw misalignment by fusing information of automotive onboard sensors, Mech. Syst. Signal Process. 162 (2022) 107993.
[56] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2016) 1137–1149.
[57] Open-mmlab, MMDetection3D: OpenMMLab next-generation platform for general 3D object detection, 2020, https://github.com/open-mmlab/mmdetection3d.
[58] A. Geiger, P. Lenz, R. Urtasun, 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, http://dx.doi.org/10.1109/CVPR.2012.6248074.
[59] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuscenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11621–11631.
[60] J. Cui, H. Qiu, D. Chen, P. Stone, Y. Zhu, COOPERNAUT: End-to-end driving with cooperative perception for networked vehicles, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 17252–17262.