Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3646272, 13 pages
https://doi.org/10.1155/2022/3646272

Research Article

Pillar-Based Cooperative Perception from Point Clouds for 6G-Enabled Cooperative Autonomous Vehicles

Jian Wang[1], **Xinyu Guo**[1], **Hongduo Wang**[1], **Pin Jiang**[2], **Tengyun Chen**[2] and **Zemin Sun**[1]

[1]*College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2]*Wireless X Labs, Huawei Technologies, Shenzhen 518000, China*

Correspondence should be addressed to Jian Wang; wangjian591@jlu.edu.cn

Received 8 April 2022; Accepted 7 July 2022; Published 25 July 2022

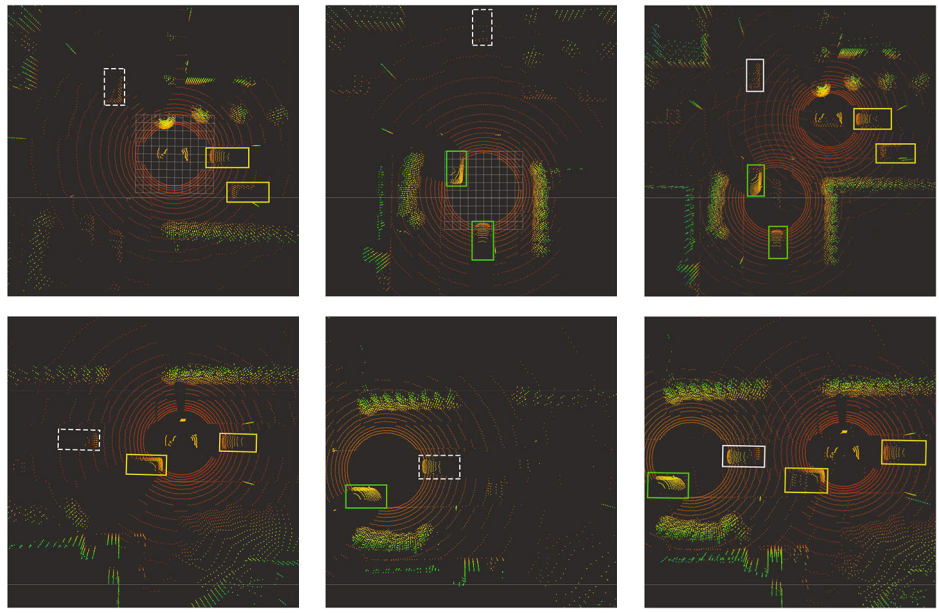
Academic Editor: Ying Chen

3D object detection is a significant aspect of the perception module in autonomous driving; however, with current technology, data sharing between vehicles and cloud servers for cooperative 3D object detection under the strict latency requirement is limited by the communication bandwidth. The sixth-generation (6G) networks have accelerated the transmission rate of the sensor data significantly with extreme low-latency and high-speed data transmission. However, which sensor data format and when to transmit it are still challenging. To address these issues, this study proposes a cooperative perception framework combined with a pillar-based encoder and Octomap-based compression at edges for connected autonomous vehicles to reduce the amount of missing detection in blind spots and further distances. This approach satisfies the constraints on the accuracy of the task perception and provides drivers or autonomous vehicles with sufficient reaction time by applying fixed encoders to learn a representation of point clouds (LiDAR sensor data). Extensive experiment results show that the proposed approach outperforms the previous cooperative perception schemes running at 30 Hz, and the accuracy of the object bounding box results in further distances (greater than 12 m). Furthermore, this approach achieves a lower total delay for the procession of the fusion data and the transmission of the cooperative perception message. To the best of our knowledge, this study is the first to introduce a pillar-based encoder and Octomap-based compression framework for cooperative perception between vehicles and edges in connected autonomous driving.

1. Introduction

Object perception is indispensable in sensing the surrounding environment and improving driving safety in autonomous vehicles (AVs). However, the perception capability of onboard sensors on AVs could be influenced by external factors such as obstacles and weather conditions [1]. To address this issue, European Telecommunications Standards Institute (ETSI) proposes the generation rules of Cooperative Perception Messages (CPMs) [2], which specify that CAVs can generate CPMs for cooperative perception when vehicles receive the object results from the object detection algorithm using their sensor data. As a result, CPMs improve the perception capability of connected autonomous vehicles (CAVs) by expanding the effective sensing range and reducing the missed detections [3].

Nevertheless, CAVs still cannot perceive the missed vehicles, which are not detected by local sensor data, because the content of CPMs only includes the object results. Figure 1 is the comparison between CPMs and fusion sensor data detection. There are two cases to explain the undetected object in which each vehicle can only perceive a sparse point cloud. The above pictures are the situation at an intersection, and the below pictures are for bidirected roads. Figure 1(a) denotes the object detection result (yellow bounding boxes) from vehicle *A*. Also, Figure 1(b) represents the result (green bounding boxes) from vehicle *B*. Finally, the fusion result shows that the undetected vehicle (white boxes) can be detected in Figure 1(c).



(a) (b) (c)

Figure 1: Illustration of the detection results of CPMs and fusion sensor data detection.

Cooper [4] is the basic idea of cooperative perception to transmit CPMs with LiDAR data from multiple CAVs and point cloud fusion for detecting 3D objects. However, transmitting raw sensor data (such as point cloud) is challenging even though the sixth-generation (6G) communication technology. Significantly, the data size of 1 frame of LiDAR sensor with 64 beams can be about 3-4 MB, and the sample rate of typical LiDAR is 30 Hz. Therefore, the network capacity is at least 960 Mb/s. Besides, the raw point cloud is a sparse representation due to the measuring method by the laser scanner. As a result, the transmission of raw sensor data is inefficient and wastes limited communication resources.

To overcome the above challenges, this work proposes a cooperative perception scheme integrating pillar-based encoder sensor data and Octomap-based compression. To the best of our knowledge, this pillar-based fusion for cooperative perception at edges is the first study that tackles the problem of missing object detection. Furthermore, the proposed solution reduces the communication delay and improves the perception capabilities of CAVs. The main contributions of this study can be summarized as follows. First, this study proposes a new data transmission type to exchange cooperative perception messages between CAVs and edges for exploring the potential of "intermediate outputs" in 3D object detection algorithms. Second, this detection scheme performance is achieved while running at 30 Hz, and the precision of distant vehicles (more than 20 m) is improved compared with the feature map fusion solution [5]. Therefore, autonomous vehicles have sufficient reaction time to handle traffic emergencies. Finally, the scheme running in the edges does not require a specific onboard object detection algorithm. In other words, CAVs do not rely on object results from the edges. The proposed solution can be seen as enhancing the perception capability of CAVs by 6G communication.

The remainder of this work is organized as follows. Section 2 presents the preliminaries and background. Section 3 introduces the proposed object perception scheme on CAVs, including the phase to transmit the specific cooperative perception messages and receive the semantic and precise object results. Then, this study designs a pillar-based fusion algorithm on edges to detect cooperative perception results. Finally, we propose an integrating CAV-based and edge-based scheme for cooperative perception. Simulation results and conclusions are given in Section 4 and Section 5.

2. Preliminaries and Background

Given state of the art in the field of 3D object detection in autonomous driving, this work starts by briefly reviewing the data choice, including data type, fusion level [6], compression, and reconstruction in general. And then this study focuses on algorithms specific to object detection from LiDAR point clouds. After that, this study describes the outlook and work done in the scheme for cooperative perception.

2.1. Data Choice

2.1.1. Type. Rapid development of 3D object detection has motivated increasing studies to design efficient representations to detect vehicles in point clouds and images. Images can provide rich color properties and detailed texture information. Numerous research deduced the 3D object detection in image [7–11] benefits from nature 2D image-based object detection technology. At first, the 2D bounding boxes are

obtained by processing images in these algorithms. After that, the bounding boxes are converted from 2D to 3D based on different methods, such as template matching [7, 8, 12, 13] and geometric properties [14–16]. However, images lack the depth information and are susceptible to lighting conditions. The accuracy of detection algorithms is bounded by the capability of the depth estimation and light compensation. Point clouds can provide accurate 3D position, speed, and depth information of objects and have advantages of spatial dimension over 2D images [4]. LiDAR is less affected by the environment; it can work even in dark or bad weather and generate real-time, high-resolution point clouds of the surrounding environment. Besides, the point cloud representation (x, y, z, r), indicating each spatial position information and reflectivity, is easy to process. Therefore, this work is dedicated to exchanging data from point clouds.

2.1.2. Fusion Level. This study uses the classification of sensor data fusion defined in [6]: low-level, feature-level, and high-level. Briefly, low-level fusion is raw data from the onboard sensor without processing. Although the low-level fusion method keeps the original spatial information of objects, the transmission of low-level fusion requires ultra-high bandwidth and is hard to apply in the 6G network. On the other hand, feature-level fusion seeks a preprocessing output from raw data before getting the object results. Hence, it can reduce the data size and keep the main features. Finally, high-level fusion means onboard sensors independently process raw data to generate an object list and fuse all the lists. Accordingly, it is less complex to implement in practice, but it cannot detect the missed objects which never detected before fusion. Considering the advantages and disadvantages of different levels fusion, this study pays attention to the feature-level fusion.

2.1.3. Compression. The point cloud provides a highly realistic view of the surrounding environment; it also costs large bandwidth to transmit uncompressed data. Therefore, compression of point clouds is necessary. The function proposed in current research can be summarized into three modes: 1D traversal, 2D projection, and 3D correlations [17]. As for 1D prediction methods, it is to construct a prediction tree to convert the geometry data into a 1D representation. Although this model is relatively simple to achieve, it does not consider the 3D spatiality of point clouds. 2D projection focuses on converting the 3D point cloud into 2D representation by projection or mapping [18]. Because of the 3D correlations of point clouds, the most common way to compress is to convert the point clouds to 3D representation [17]. Therefore, the 3D representation is used in this study. There are various approaches to achieve this purpose, such as octree-based coder [19], hierarchical clustering coder [20], and context-based intracoder [21]. The octree-basedcoder is adopted by this work because the lossless encoding is suitable for fusion.

2.1.4. Reconstruction. The received data cannot be processed straightforwardly because it is generated on different positions and angles. Thus, edges need to reconstruct the exchanging point cloud data into its UTM (Universal Transverse Mercator) coordinate system. In order to merge point clouds, cooperative perception messages need GPS and IMU (Inertial Measurement Unit) as additional information to calculate the offset information (Euler angles, unit quaternions, and rotation vectors [22]). Euler angles represent a rotation with yaw, pitch, and roll angles for $z-$, $y-$ and $x-$ axis that are θ, φ, and ψ, respectively. It can be calculated by using the IMU value difference between the transmitter and receiver. Besides, the quaternions can be represented as follows:

$$q = q_w + q_x i + q_y j + q_z k. \tag{1}$$

Given Euler angles θ, φ, and ψ, the quaternions can be calculated as follows:

$$\begin{aligned}
q_w &= \cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\varphi}{2}\right) + \sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\varphi}{2}\right),\\
q_x &= \cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\varphi}{2}\right) - \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\varphi}{2}\right),\\
q_y &= \cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\varphi}{2}\right) + \sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\varphi}{2}\right),\\
q_z &= \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\varphi}{2}\right) - \cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\varphi}{2}\right).
\end{aligned} \tag{2}$$

Subsequently, the rotation matrix $\mathbf{R}$ can be written as follows:

$$\mathbf{R} = \begin{bmatrix}
1 - 2\left(q_y^2 + q_z^2\right) & 2\left(q_x q_y - q_w q_z\right) & 2\left(q_w q_y + q_x q_z\right)\\
2\left(q_x q_y + q_w q_z\right) & 1 - 2\left(q_x^2 + q_z^2\right) & 2\left(q_y q_z - q_w q_x\right)\\
2\left(q_x q_z - q_w q_y\right) & 2\left(q_w q_x + q_y q_z\right) & 1 - 2\left(q_x^2 + q_y^2\right)
\end{bmatrix}. \tag{3}$$

Finally, the coordinate of point clouds can be calculated from different sensors as follows:

$$\mathbf{P}_{\mathbf{transform}} = \mathbf{R} \cdot \mathbf{P}_{\mathbf{original}}, \tag{4}$$

where P_{original} is the original coordinates from its sensor, and $P_{\text{transform}}$ is the coordinate transformed to the UTM system.

2.2. 3D Object Detection in Point Clouds. A 3D object detection algorithm is divided into three parts: data representation, feature extraction (i.e., backbone), and detection network (detection head). Data representation organizes the point cloud generated by LiDAR into a proper structure to which the convolution operations can be applied. The main approaches are voxel-based, point-based, frustum-based, pillar-based, and 2D projection-based [23]. First, voxel is short for volume pixel, dividing point clouds into 3D grids at a specific resolution in a 3D Cartesian coordinate

system. Many voxel-based methods are proposed in the earlier studies such as VoxelNet [24], SECOND [25], and Voxel-FPN [26]. The advantage is that it can easily migrate neural network operations such as convolution based on 3D grids. The disadvantage is that the efficiency is low due to a large amount of discrete computation and encoder. Besides, point-based methods process the raw data and generate a sparse representation and then aggregate the features of adjacent points, and the feature of each point is extracted [23]. However, this method poses stringent requirements on the hardware compare with the voxel-based method. Pillar-based method organizes point clouds in the x-y plane; since the point cloud is three-dimensional, point clouds are formed into vertical columns, called pillars, such as PointPillars [27]. Due to the exclusion of the z-axis and fixed encoder, the pillar-based method has a significant processing rate improvement compared to other algorithms, and it also achieves top efficiency and performance on recent 3D object detection benchmarks. Therefore, this study chooses the pillar-based method as data representation due to its speed and accuracy. The convolutional neural network (CNN) architectures [28] as feature extraction can be seen as the state of art in images and point clouds. Besides, the detection head used in this work, as the layer to compensate for CNN positioning capabilities, is Single Shot Detector proposed in [29].

2.3. Current Cooperative Perception Schemes Proposed. The benefits of cooperative perception have been studied in recent years [30–33]. Motivated by its potential, many researches focused on several areas, including communication protocol [34], generation rules of CPMs [1], and algorithm design. In CarSpeak [34], a content-centric communication protocol around the needs of cooperative perception has been proposed. It focuses on sensor information sharing at the medium access control (MAC) layer and is fully integrated with autonomous driving. On the other hand, Thandavarayan et al. [1, 3] has analyzed the performance of cooperative perception messages and offered generation rules to define which information should be included. In algorithm design, Chen et al. [4, 5] and Guo et al. [35] have proposed a series of feature map fusion methods on CAVs by vehicle-to-vehicle (V2V) communication. Aoki et al. [32] proposed the scheme with deep reinforcement learning (DRL) to select the data to transmit. Besides, the infrastructure sensor-supported cooperative perception has been proposed in [36, 37]. However, most of these studies only rely on the sensor data from assist CAVs, which cannot operate independently with local data. 3D object detection as a kind of computation-intensive tasks generated by CAVs is limited by battery and computing capacity. Therefore, some works focus on dynamic task offloading for mobile edge computing [38, 39]. The collaborative task offloading for vehicles and cloud faces challenges by the increase of the scales of task offloading problem and solution space size. These studies are meaningful and significant but beyond the scope of this paper. Distinguished from the previous works, this study focuses on the algorithm on edges to enhance the object detection of missed objects.

3. Pillar-Based Cooperative Perception from Point Clouds

Inspired by the advantages of the fixed encoder such as PointPillars [27] and Octomap compression [40], this study proposes the pillar-based cooperative perception framework from point clouds on CAVs and edges. This study presents two schemes for cooperative perception: Cooperative Perception on Vehicles (CPV) and Cooperative Perception on Edges (CPE). This section presents the detail and architecture framework of two schemes.

3.1. Cooperative Perception on Vehicles. The proposed architecture of CPV, depicted in Figure 2, consists of three components: Octomapcompression, pillarupload, and listfusion. These three components are executed sequentially. Besides, pillarupload and listfusion start to execute when the vehicles receive corresponding messages from edges. It should be noted that the 3D object detection algorithm in this work is PointPillars [27]. However, it does not mean that CPVs only rely on PointPillars. The detection algorithm can be replaced by others. At first, the vehicle process downsampled 3D LiDAR point cloud represented as (x, y, z, r) with 3D position and reflection value r into Octomap [40]. Octomap based on tree structure uses probabilistic occupancy estimates to support lossless compression. Specifically, CAVs convert the range of point clouds into a cube space with side length L. This space will be divided recursively into eight subvoxels until the minimum voxel size is r_o. The leaf node in the Octomap only stores the occupancy probability and one pointer of an inner node with eight pointers to its children or a null pointer (no child node), as shown in Figure 3. The occupancy probability of the node is the maximum occupancy of all the eight subvoxels as follows:

$$O(n) = \max_i o(n_i), i = 1, \cdots, 8. \tag{5}$$

After that, CAVs transmit the Octomap data to edges. Then, the CPVs execute the next component when it has received cuboid regions (how to generate cuboid regions on edges will detail in Section 3.2) from edges. Then, CAVs discretize the points into pillars P, which is no control in the z-dimension based on the cuboid regions. The point p in the pillar represents with nine dimensions as follows:

$$p = \left(x, y, z, r, x_c, y_c, z_c, x_p, y_p\right), \tag{6}$$

where c denotes the center point with distance to the mean of all points, and the p denotes the offset from the center point. Besides, if there are too many points in a pillar, this scheme randomly discards the points. Conversely, the zero points pad into the pillars with less than N points. Once the generation of pillars is finished, CAVs transmit the feature pillars to edges with their locations.

Finally, CAVs receive the object list L_{cloud} from edges. Vehicles execute the listfusion component to obtain final 3D bounding box of objects around itself. The detail of fusion algorithm based on weightbox is shown in Algorithm 1.

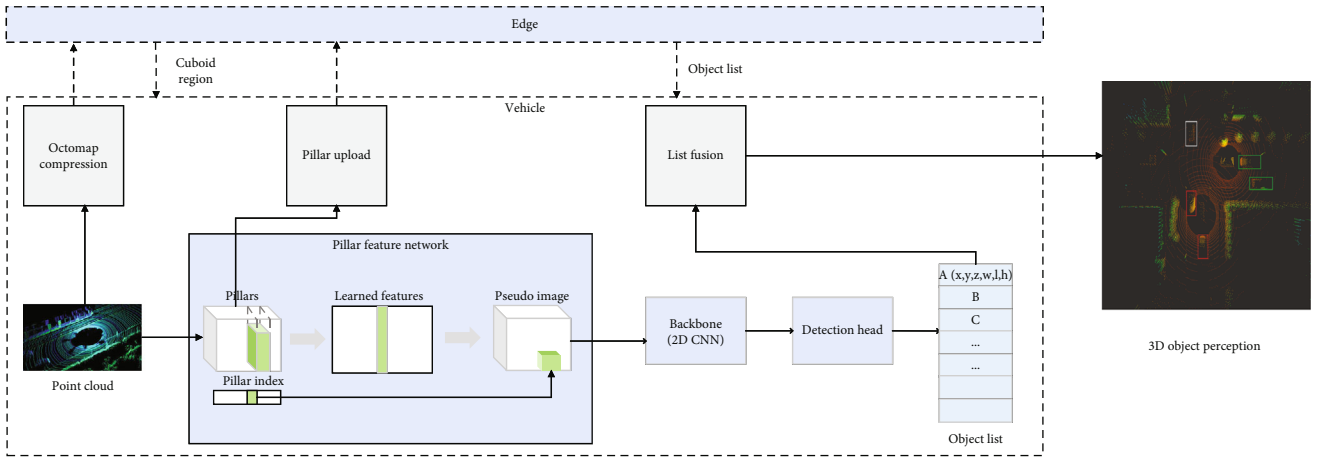


Figure 2: Architecture of the Cooperative Perception on Vehicles (CPV).

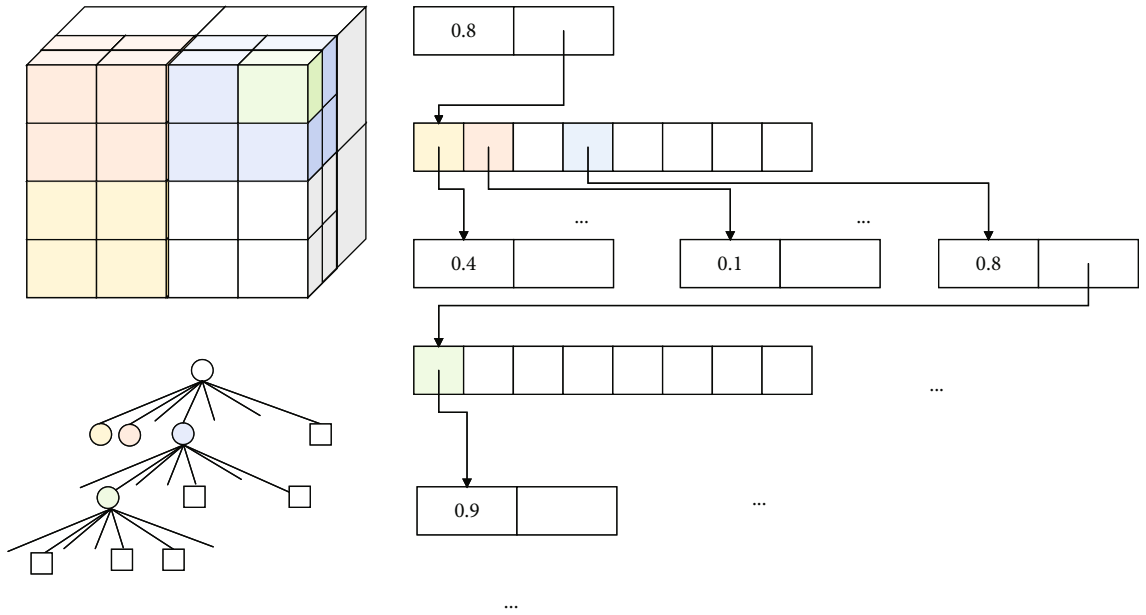


Figure 3: Illustration of an Octomap storing tree, including the model and corresponding tree representation and the structure of a leaf node.

In the algorithm, vehicles first initialize the parameters such as groups that are the same object candidates M and the threshold of IOU (Intersection over Union) Thr. Then, the proposed algorithm removes the points out of range R_{vehicle}. Secondly, this algorithm filters out the candidates of a 3D bounding box for the same object and stores the candidates in a group of the list M. Therefore, M includes multiple groups. Finally, the algorithm outputs the object list L_{final} after processing the list M by weighted average in the same index. This algorithm can effectively reduce the maintenance detection list because these out-of-range bounding boxes have useless value for the vehicle.

3.2. Cooperative Perception on Edges. There are four main components: Octomapfusion, cuboidregionextraction, pillar fusion, and objectdetection. These components are executed sequentially. It should be noted that the 3D object detection algorithm is based on PointPillars [27] in this work. Besides, Octomapfusion and pillarfusion can be executed when the edges have received corresponding basic messages from vehicles.

3.2.1. Octomap Fusion. The CPE executes Octomap fusion component as shown in Figure 4 when the edge receives messages from CAVs, including Octomap and the locations of vehicles. At first, all the Octomaps should transfer into the UTM coordinate system by using data reconstruction mentioned in Section 2.1.4. After that, there are four conditions: (1) unknown, (2) samedepth, (3) coarse, and (4) finer, shown in Figure 5, to merge the leaf node n into final map T_{fusion} from one Octomap T_i due to the hierarchical tree-structure. If the leaf node in T_i is not constructed on the T_{fusion} (the state is unknown), the Octomap adds the node into T_{fusion}, as follows:

Data: Object list from cloud L_{cloud} and object list from vehicle $L_{vehicle}$
Result: Object list L_{final}

List M, $R_{vehicle}$, Thr, α, β initialization
/*Remove the objects which are out of range of the vehicle */
while index in List L_{cloud} **do**
  if $\|L_{cloud}[index] - Location(Vehicle)\| > 2 * R_{vehicle}$ **then**
    remove $L_{cloud}[index]$
/*Storage the objects in same location to M */
while j in List L_{cloud} **do**
  //Thr is the threshold of IOU
  if IOU s$(L_{cloud}[j], L_{vehicle}) \geq Thr$ **then**
    $M[j(L_{vehicle})][:] = L_{cloud}[j]$
  else
    Insert $(L_{cloud}[j])$ to $L_{vehicle}$
    $M[j(L_{vehicle})][:] = L_{cloud}[j]$
/*Calculate the final bounding box and loction of object in List M */
while k in List M **do**
//Weighted Average
$L_{vehicle}[k] = \alpha \cdot L_{vehicle}[k] + \beta \cdot \sum_{i=1}^{sum(M[k])} M[k][i]/sum(M[k])$
$L_{final} = L_{vehicle}$
RETURN L_{final}

Algorithm 1: Object list fusion based on weightbox.

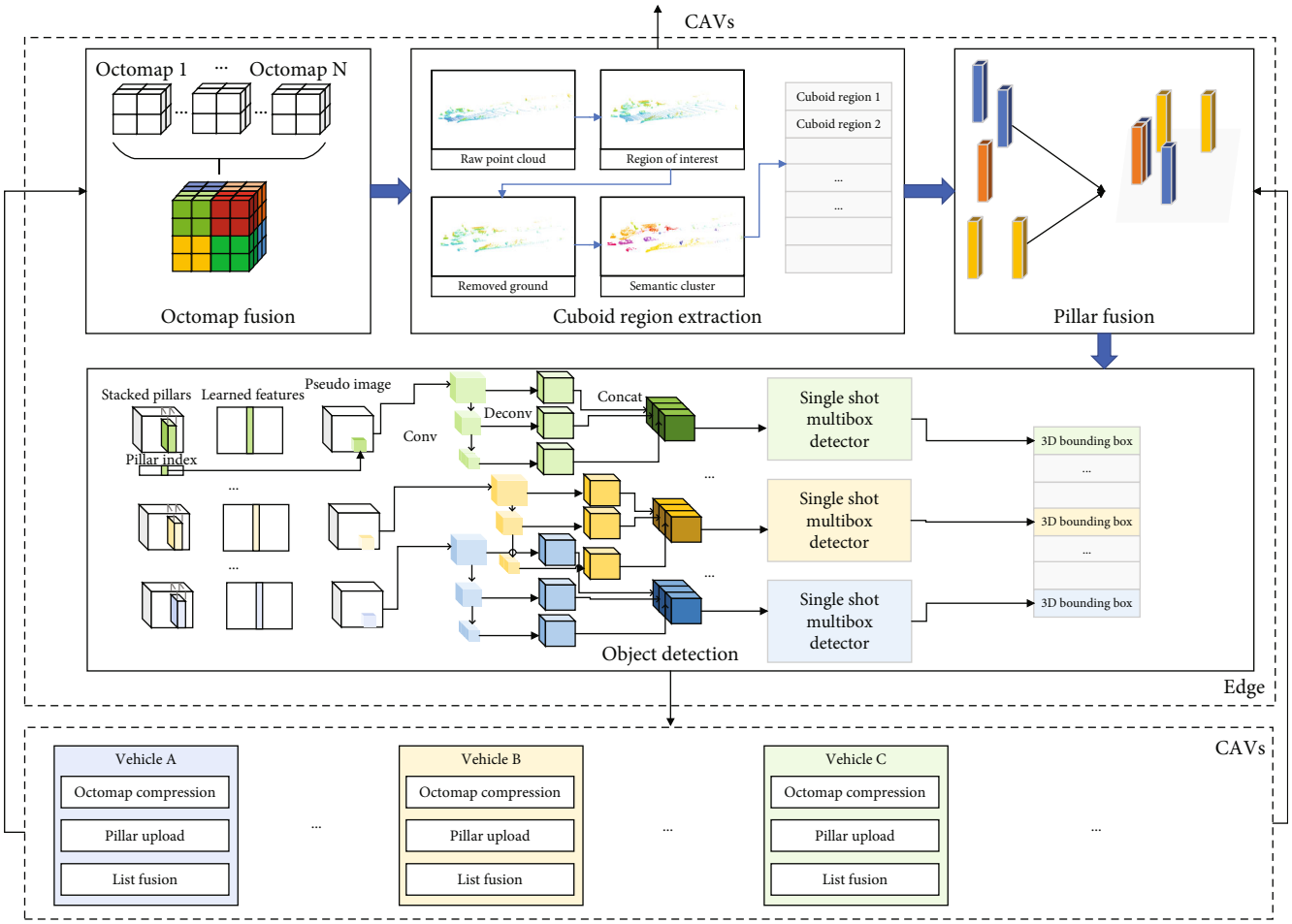


Figure 4: Architecture of the Cooperative Perception on Edges (CPE).

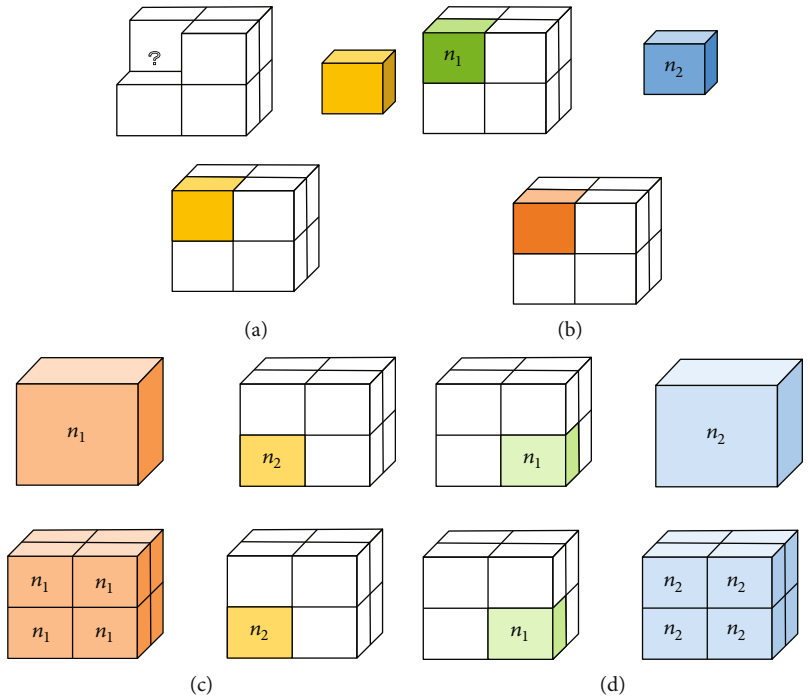


Figure 5: Illustration of an Octomap fusion by different situations, including (a) unknown, (b) same depth, (c) coarse, and (d) finer.

$$L(n_1) = L(n_2), \tag{7}$$

where the logarithmic occupancy probability of n is $L(n)$,and n_1, n_2 denotes the leaf node in T_{fusion} and in T_i, respectively. If $L(n_1)$ and $L(n_2)$ are in samedepth, the probability will be updated, as follows:

$$L(n_1) = L(n_1) + L(n_2). \tag{8}$$

In condition (3), if $L(n_2)$ is represented by coarse resolution node $L(n_1)$, $L(n_1)$ is divided into eight child nodes $L(n_1^i)$ with the same occupied probability and update as follows:

$$\begin{aligned} L\left(n_1^i\right) &= L(n_1), i = 1, \cdots, 8, \\ L\left(n_1^i\right) &= L\left(n_1^i\right) + L(n_2). \end{aligned} \tag{9}$$

Similarly, the update occupied probability is represented in condition (4), as follows:

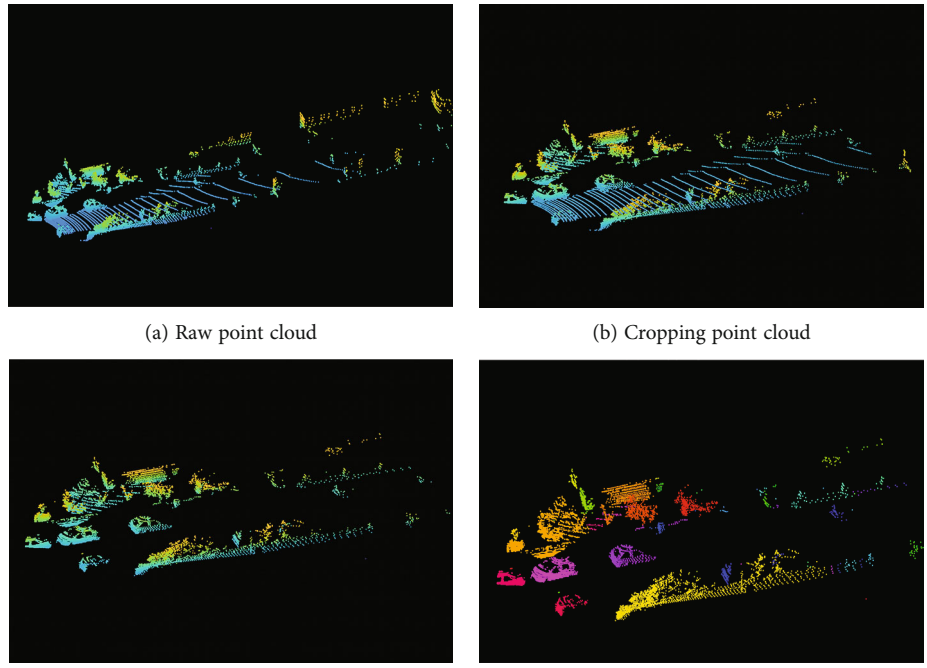
$$\begin{aligned} L\left(n_2^i\right) &= L(n_2), \qquad i = 1, \cdots, 8, \\ L(n_1) &= L(n_1) + L\left(n_2^i\right). \end{aligned} \tag{10}$$

The final Octomap can be transferred into point clouds C_{fusion} after all the T_i merge into T_{fusion}. The detail of the Octomap fusion algorithm on the edge is shown in Algorithm 2.

3.2.2. Cuboid Region Extraction. The cuboid region extraction will be executed after the edge gets point clouds C_{fusion}. Point clouds C_{fusion} are further classified into clusters by Euclidean, and each cluster is converted into a coarse-grained 3D bounding box. At first, point clouds outside the X_{limit}, Y_{limit}, and

Data: Octomap from Vehicles $T_i, i = 1, \cdots, N$
Result: Octomap T_{fusion}
$T_{fusion}, i = 0$ initialization;
while i $\leq N$ **do**
  i=i+1
  T_i reconstruction based on Eq. (4)
  // Octomap Fusion(T_{fusion}, T_i)
  for Leaf node $L(n_2)$ in T_i **do**
    if $L(n_2)$ not in T_{fusion} **then**
      Add node: $L(n_1) = L(n_2)$
    else
      The depth of $L(n_2)$ is d_2
      for Leaf node $L(n_1)$ in T_{fusion} **do**
        The depth of $L(n_1)$ is d_1
        if $d_2 = d_1$ **then**
          $L(n_1) = L(n_1) + L(n_2)$
          else $d_2 - d_1 = 1$ && $L(n_1).child$ is $NULL$ **then**
            Add child node to T_{fusion}: $L(n_1^i) = L(n_1)$;
            $L(n_1^i) = L(n_1^i) + L(n_2)$
          else if $d_1 - d_2 = 1$ && $L(n_2).child$ is $NULL$ **then**
            Add child node to T_i: $L(n_2^i) = L(n_2)$
            $L(n_1) = L(n_1) + L(n_2^i)$
RETURN T_{fusion}

Algorithm 2: Octomap fusion on the edge.



(a) Raw point cloud (b) Cropping point cloud

(c) Removed ground plane (d) Segment out obstacle

Figure 6: The point clouds of cuboidregionextraction at four different stages: (a) the raw point clouds, (b) the region of interest (RoI) by removing outside points, (c) the points removed ground plane, and (d) the semantic results by using different colors.

Z_{limit} are removed. After that, the ground plane is fitted from the survived points. And the unrecognized obstacle can be clustered by Euclidean because of removing the ground plane. In the end, all the bounding boxes are proposed by clusters. The cuboid region extraction is an adapted version of the traditional and efficient point cloud segmentation algorithm implemented by PCL [41], and its algorithm flowchart is shown in Figure 6.

3.2.3. Pillar Fusion. After vehicles transmit the pillars data, CPE executes *pillarfusion* component on edge as shown in Figure 4. For fixed backbone architecture, this study first feeds the point clouds to pseudoimages. The received pillars based on their locations fuse into different cuboid regions (can also be called as subtasks). In a subtask, there are some pillars in the overlapping area as shown in Figure 4. The approach in this study to fuse these pillars into one subtask can be represented in the following equation:

$$P = \{P_{\text{nol}} \cup P_{\text{ol}} = \{\text{sample}(p_i)\}\}, p_i \in P_A \cap \cdots \cap P_B, \tag{11}$$

where P_{nol} denotes the nonoverlapping areas of subtask in ground plane, P_{ol} randomly samples from all the pillars p_i in the overlapping areas, and P_A, P_B denote the pillars from vehicle A and B.

3.2.4. Object Detection. The fused pillars feed with a linear layer followed by BatchNorm [42], ReLU [43], and max pooling [44] to generate and reshape tensor size (C, W, H). After that, the backbone and detection head are similar to PointPillars [27].

The loss function is based on ground truth boxes and anchors defined by $(x, y, z, w, l, h, \theta)$. The total loss is as follows:

$$\mathscr{L} = \frac{\beta_{\text{cls}}}{N_{\text{cls}}}\mathscr{L}_{\text{cls}} + \frac{\beta_r}{N_r}\left(\mathscr{L}_\theta + \mathscr{L}_{\text{reg}}\right) + \frac{\beta_a}{N_a}\mathscr{L}_a, \tag{12}$$

with $\beta_{\text{cls}} = 1.0$, $\beta_r = 2.0$, and $\beta_a = 0.1$. $\mathscr{L}_{\text{reg}}$ is regression loss for (x, y, z, w, l, h) by Smooth$L1$-loss [45] in the following equation:

$$\mathscr{L}_{\text{reg}} = \sum_{\alpha\in(x,y,z,w,l,h)} L_{1,\text{smooth}}(\Delta\alpha), \tag{13}$$

which

$$\begin{aligned} L_{1,\text{smooth}}(x) &= \begin{cases} \frac{1}{2}x^2 & \text{if } |x| < 1 \\ |x| - \frac{1}{2} & \text{otherwise,} \end{cases} \\ \Delta x &= \frac{x_{gt} - x_a}{\sqrt{\left(w_a{}^2 + l_a{}^2\right)}}, \\ \Delta y &= \frac{y_{gt} - y_a}{\sqrt{\left(w_a{}^2 + l_a{}^2\right)}}, \\ \Delta z &= \frac{z_{gt} - z_a}{h_a}, \\ \Delta w &= \log w_{gt} - \log w_a, \\ \Delta l &= \log l_{gt} - \log l_a, \\ \Delta h &= \log h_{gt} - \log h_a. \end{aligned} \tag{14}$$

The Smooth L1-loss is less sensitive to outliers than other regression loss. Smooth L1-loss can be interpreted as a combination of L1-loss and L2-loss. It behaves as L1-loss when the absolute value of the argument is high, and it behaves like L2-loss when the absolute value of the argument is close to zero. $\mathscr{L}_\theta$ is angle regression loss as follows:

$$\mathscr{L}_\theta = L_c\left(\theta_{gt}^c, \theta_a^c\right) + D\left(\theta_{gt}^r, \theta_a^r\right), \tag{15}$$

where superscript c and r represent angle class and residual, respectively, L_c is orientation classification loss, and D is residual prediction loss.

For the detection classification loss $\mathscr{L}_{cls}$, this study uses the Focal Loss:

$$\mathscr{L}_{cls} = -\alpha(1-p^a)^\gamma \log p^a, \tag{16}$$

with $\alpha = 0.25$ and $\gamma = 2$. Besides, this study uses a softmax loss on the $\mathscr{L}_a$.

4. Simulations

This section presents simulation results to compare the performance with baseline and *F*-Cooper [5] under the same datasets and scenarios. In order to have the same settings, this study carries out experiments with our dataset for cooperative 3D object detection. The dataset is simulated scenarios by the gazebo [46]. All data is collected from vehicles equipped with a 64-beam LiDAR, one GPS, and one IMU. There are the detailed parameters in Table 1. The parameter setting of LiDAR is suitable for most products on the market. Besides, the distance of "Near" and "Far" vehicle is defined by [5], and the other parameters (β_{cls}, β_r, β_a, α, and γ) refer to [27]. In the experiments, the framework proposed, and F-Cooper runs on a computing device with an NVIDIA RTX 3090 GPU.

Table 1: Parameters [5, 27].

Name	Value
Vertical sampling rate	64
Horizontal sampling rate	1024
Angular resolution	0.08
X-axle detection range R_x	[0, 25.6]
Y-axle detection range R_y	[-40, 40]
Z-axle detection range R_z	[-3, 1]
"Near" vehicle	0-12.5 m
"Far" vehicle	≥12.5 m
X-axle pillar size P_x	0.2 m
Y-axle pillar size P_y	0.2 m
Transmission rate	1 Gbps
β_{cls}	1.0
β_r	2.0
β_a	0.1
α	0.25
γ	2
Confidence score threshold	0.5

4.1. Detection Precision Analysis. Table 2 shows the performance of the proposed algorithm against the baseline and *F*-Cooper [5] by comparing the detected vehicles with the confidence score threshold at 0.5. There are three scenarios in the experiments, including multilane roads test, road intersection 1 test, and road intersection 2 test. The difference between these two road intersection tests is the number of vehicles. The former with more vehicles can be seen as traffic jams. Moreover, the latter can be thought of as an uncongested intersection. This study chooses vehicle one as ego vehicle for convenience compared with "Near" and "Far" results.

In a multilane road scenario, vehicle one can have a good "Near" detection accuracy but a weak vision of "Far" vehicles with 22.22% precision. Besides, vehicle 2 can only detect 16.67% of nearby objects and cannot sense any vehicles at a "Far" distance due to the severe occlusion. Next, *F*-Cooper and our proposed algorithm achieve a similar detection precision at "Near" distance compared with baseline. These two approaches can significantly improve detection precision at "Far" vehicles with 42.78% and 37.5%, respectively, compared with 22.22% in baseline test. It means that the drivers can have more reaction slot windows due to the more accurate perception of the further vehicle.

In the road intersections 1 case, the CAVs are not affected by occlusion, which means that detection precision without fusion can perform well at "Near" and "Far" distances, such as 75% and 42.86% detection on vehicle 1, respectively. On the other hand, the fusion methods can perform better with 85.71% by our proposed approach and 80.21% by *F*-Cooper at near distance and 50% and 46.42% at long distance, respectively.

Road intersection 2 is a particular case because the vehicles in this scenario are far away from each other. Therefore, it mainly focuses on the result of the far distance to show the performance. It can be noted that the detection precision on vehicle 1 is 25%, and on vehicle 2 is 40%. Besides, the *F*-Cooper and method proposed in this study can achieve 36.41% and 48.81%, respectively. This result signifies that our method can improve approximately 10% on fusion precision.

4.2. Data Volume Evaluation. The data size generated from different scenarios is approximately equal. Therefore, this study only shows the data from road intersection 1, which means that vehicles are detected near and far. In Figure 7, the bars depict the amount of data that needs to transmit and processed compared to the raw point cloud and original Pillars.

It can be seen that the raw data gathered directly from the 64-beam LiDAR region of interest (RoI) is about 4 MB. Similarly, the original data volume of pillars ranges from 3.4 MB to 3.7 MB because of lots of null points. After

Table 2: Precision comparison among baseline, *F*-Cooper, and proposed algorithm on a receiver.

Scenario	Baseline (vehicle 1)		Baseline (vehicle 2)		*F*-Cooper [5]		Ours	
	Near	Far	Near	Far	Near	Far	Near	Far
Multilane roads	66.66%	22.22%	16.67%	0.00%	65.51%	42.78%	71.43%	37.50%
Road intersection 1	75.00%	42.86%	66.67%	36.36%	80.21%	46.42%	85.71%	50.00%
Road intersection 2	N/A	25.00%	N/A	40.00%	N/A	36.41%	N/A	48.81%

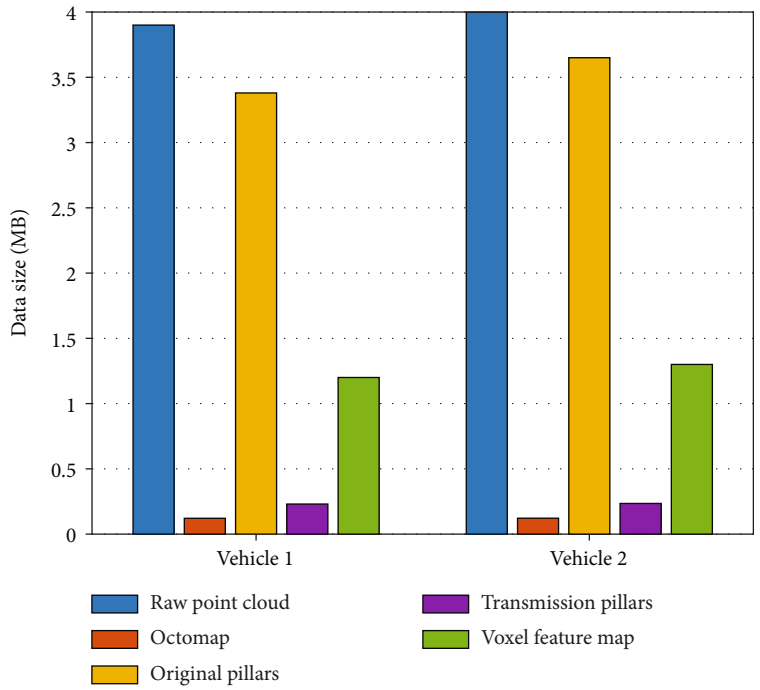


Figure 7: Comparison of data volume using different strategies. The data is collected from Road intersection 1.

processing and compression, the data size of different strategies falls off sharply to less than a quarter of raw data volume. The average voxel feature map mentioned by *F*-Cooper is about 1.25 MB. The data volume satisfies the transmission constraint of 6G communication (about 125 KB per millisecond in this experiment setting). The processing time of the feature map is too long to meet the real-time running before transmission. This study will discuss this in the following subsection. Moreover, point clouds compressed into Octomap significantly reduce data size, approximately 124.6 KB. Besides, the data volume of pillars transmitted by CAVs is about 238.29 KB. Both Octomap and transmission pillars can process within time slots that meet real-time requirements (details will be shown later).

It should be noted that the Octomap compression is necessary for transmission by the difference in data volume between the original pillars and transmission pillars. The dominant factors that impact the detection precision in PointPillars [27] are the width and length of a pillar. The smaller pillar size is able to get a more accurate object detection result. While ensuring the granularity of pillar, Octomap compression can significantly reduce the data size before transmitting pillars.

4.3. Time Consumption Evaluation. As shown in Figure 8, the total time used for different components proposed in the architecture of this study is all less than 15 ms. Besides, the time consumption of this whole framework is approximately 39 ms, which is satisfied with the 30 Hz requirement. It can be noted that Octomapcompression, and pillarupload consume the most transmission time because the data size of Octomap and pillars is much greater than the object list used in the other components. However, the time spent is still less than 2 ms. As a result, the transmission time is almost negligible compared with the processing time. The total time of cuboidregionextraction and objectdetection includes Octomapfusion and pillarfusion, respectively, before running the algorithm in this work. It can be seen that the primary time spent on the process is in Octomapcompression and objectdetection. Therefore, the raw data compression technology and 3D object perception algorithm will be mainly focused on improving cooperative detection in the future.

Besides, this study shows the time-consuming comparison with different strategies mentioned before in Figure 9. Although communicating with edges/vehicles by 6G, the raw data exchanging in Cooper [4] and feature map sharing in *F*-Cooper spend 32.76 ms and 10.24 ms in transmission,

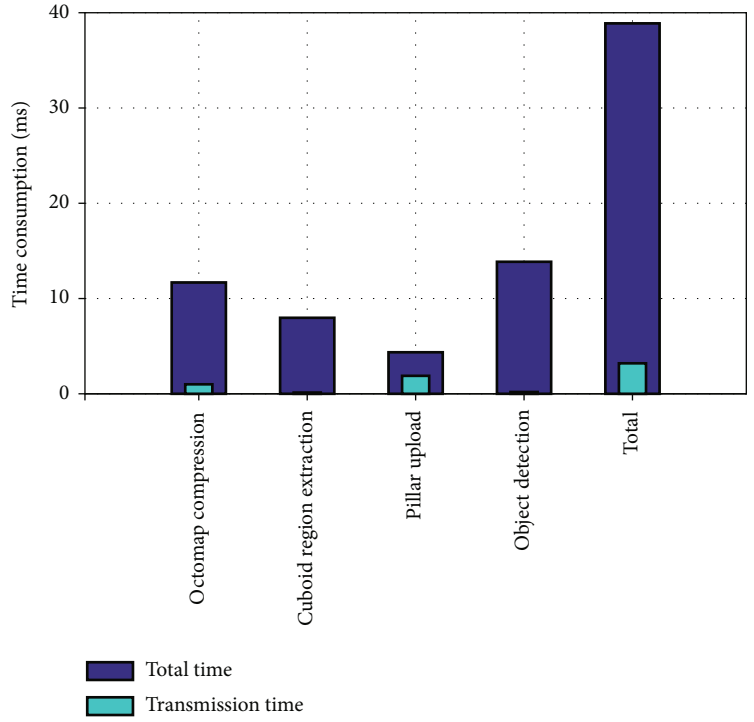


Figure 8: Comparison of time-consuming used in different components.

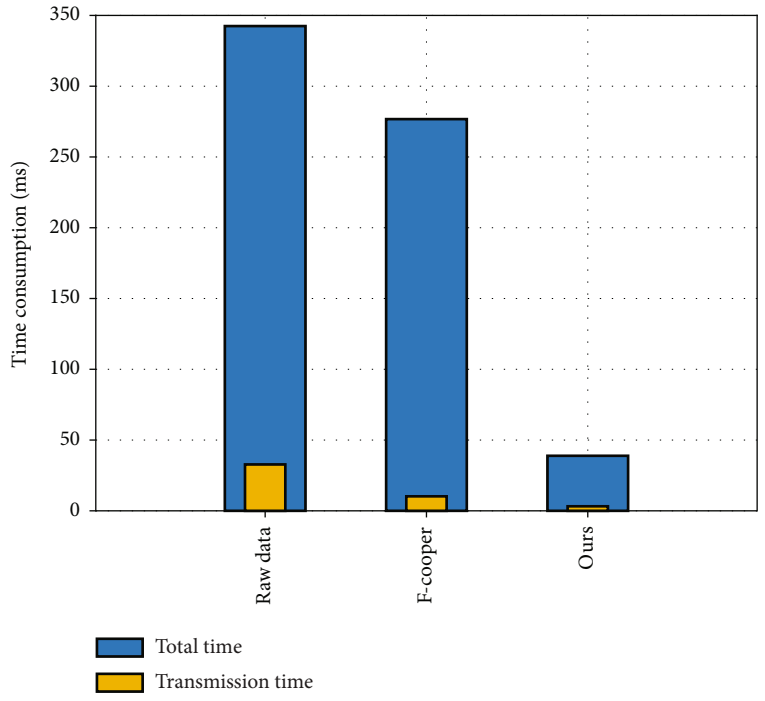


Figure 9: Comparison of time-consuming used by different strategies, including raw data, *F*-Cooper, and the proposal.

respectively. This time consumption does not meet the low latency requirement of object detection. Besides, both Cooper and *F*-Cooper are based on VoxelNet, a 3D CNN backbone. It can be noted that the processing time based on VoxelNet is approximately 237 ms, which is almost 15 times longer than that of the algorithm based on PointPillars.

5. Conclusions

In this study, a novel pillar-based perception framework has been proposed for cooperative autonomous vehicles and edges. The simulation results and analysis show that the proposed algorithm is able to achieve a 30 Hz running and

higher detection accuracy than previous schemes in terms of "Far" vehicles. It can be deployed for both CAVs and roadside edges as the improvement of perception in real-world scenarios.

In the future, the authors will focus on more data exchange types for CAVs to achieve higher detection accuracy under low delay constraints. Besides, we will also pay attention to the object detection algorithm with a higher frame rate, such as 60 Hz, without detection precision loss.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Jilin Province Science and Technology Project (No. 20200501012GX).

References

[1] G. Thandavarayan, M. Sepulcre, and J. Gozalvez, "Generation of cooperative perception messages for connected and automated vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16336–16341, 2020.

[2] *Intelligent transport system (its); vehicular communications; basic set of applications; analysis of the collective-perception service (cps)*, ETSI 103 562 V2.1.1, Tech. Rep, 2019.

[3] G. Thandavarayan, M. Sepulcre, and J. Gozalvez, "Analysis of message generation rules for collective perception in connected and automated driving," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 134–139, 2019.

[4] Q. Chen, S. Tang, Q. Yang, and 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)*, pp. 514–524, 2019.

[5] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and 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*, pp. 88–100, 2019.

[6] J. Shi, W. Wang, X. Wang et al., "Leveraging spatio-temporal evidence and independent vision channel to improve multi-sensor fusion for vehicle environmental perception," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 591–596, 2018.

[7] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2147–2156, 2016.

[8] X. Chen, K. Kundu, Y. Zhu et al., "3d object proposals for accurate object class detection," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[9] S. Song and M. Chandraker, "Joint sfm and detection cues for monocular 3d localization in road scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3734–3742, 2015.

[10] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1903–1911, 2015.

[11] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler, "Detailed 3d representations for object recognition and modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2608–2623, 2013.

[12] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, "Deep manta: a coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2040–2049, 2017.

[13] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2018.

[14] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7074–7082, 2017.

[15] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang, "Gs3d: an efficient 3d object detection framework for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1019–1028, 2019.

[16] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7644–7652, 2019.

[17] C. Cao, M. Preda, and T. Zaharia, "3d point cloud compression: a survey," in *The 24th International Conference on 3D Web Technology*, pp. 1–9, 2019.

[18] H. Houshiar and A. Nüchter, "3d point cloud compression using conventional image compression for efficient data transmission," in *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT)*, pp. 1–8, 2015.

[19] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "Octree-based progressive geometry coding of point clouds," in *PBG@ SIGGRAPH*, pp. 103–110, 2006.

[20] Y. Fan, Y. Huang, and J. Peng, "Point cloud compression based on hierarchical point clustering," in *in 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–7, 2013.

[21] D. C. Garcia and R. L. de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 1807–1811, 2018.

[22] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[23] D. Fernandes, A. Silva, R. Névoa et al., "Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy," *Information Fusion*, vol. 68, pp. 161–191, 2021.

[24] Y. Zhou and 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*, pp. 4490–4499, 2018.

[25] Y. Yan, Y. Mao, and B. Li, "Second: sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[26] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang, "Voxel-fpn: multiscale voxel feature aggregation for 3d object detection from lidar point clouds," *Sensors*, vol. 20, no. 3, p. 704, 2020.

[27] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12697–12705, 2019.

[28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[29] W. Liu, D. Anguelov, D. Erhan et al., "SSD: single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[30] S.-W. Kim, B. Qin, Z. J. Chong et al., "Multivehicle cooperative driving using cooperative perception: design and experimental validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 663–680, 2015.

[31] Y. Wang, G. De Veciana, T. Shimizu, and H. Lu, "Performance and scaling of collaborative sensing and networking for automated driving applications," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, 2018.

[32] S. Aoki, T. Higuchi, and O. Altintas, "Cooperative perception with deep reinforcement learning for connected vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 328–334, 2020.

[33] R. Hussain and S. Zeadally, "Autonomous cars: research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2018.

[34] S. Kumar, L. Shi, N. Ahmed, S. Gil, D. Katabi, and D. Rus, "CarSpeak," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 259–270, 2012.

[35] J. Guo, D. Carrillo, S. Tang et al., "Coff: cooperative spatial feature fusion for 3-d object detection on autonomous vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11078–11087, 2021.

[36] M. Gabb, H. Digel, T. Müller, and R. W. Henn, "Infrastructure-supported perception and track-level fusion using edge computing," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1739–1745, 2019.

[37] E. Arnold, M. Dianati, R. de Temple, and S. Fallah, "Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors," in *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[38] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, vol. 10, 2021.

[39] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning," *International Journal of Communication Systems*, vol. e5154, 2022.

[40] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: a probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[41] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[42] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[43] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2018, http://arxiv.org/abs/1803.08375.

[44] J. Nagi, F. Ducatelle, G. A. Di Caro et al., "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pp. 342–347, 2011.

[45] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[46] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, 2004.