

# Edge-Enabled Collaborative Object Detection for Real-Time Multi-Vehicle Perception

Everett Richards  
San Diego State University  
San Diego, California, USA  
Email: ehrichards@sdsu.edu

Bipul Thapa  
University of Delaware  
Newark, Delaware, USA  
Email: bipul@udel.edu

Lena Mashayekhy  
University of Delaware  
Newark, Delaware, USA  
Email: mlена@udel.edu

**Abstract**—Accurate and reliable object detection is critical for ensuring the safety and efficiency of Connected Autonomous Vehicles (CAVs). Traditional on-board perception systems have limited accuracy due to occlusions and blind spots, while cloud-based solutions introduce significant latency, making them unsuitable for real-time processing demands required for autonomous driving in dynamic environments. To address these challenges, we introduce an innovative framework, Edge-Enabled Collaborative Object Detection (ECOD) for CAVs, that leverages edge computing and multi-CAV collaboration for real-time, multi-perspective object detection. Our ECOD framework integrates two key algorithms: Perceptive Aggregation and Collaborative Estimation (PACE) and Variable Object Tally and Evaluation (VOTE). PACE aggregates detection data from multiple CAVs on an edge server to enhance perception in scenarios where individual CAVs have limited visibility. VOTE utilizes a consensus-based voting mechanism to improve the accuracy of object classification by integrating data from multiple CAVs. Both algorithms are designed at the edge to operate in real-time, ensuring low-latency and reliable decision-making for CAVs. We develop a hardware-based controlled testbed consisting of camera-equipped robotic CAVs and an edge server to evaluate the efficacy of our framework. Our experimental results demonstrate the significant benefits of ECOD in terms of improved object classification accuracy, outperforming traditional single-perspective onboard approaches by up to 75%, while ensuring low-latency, edge-driven real-time processing. This research highlights the potential of edge computing to enhance collaborative perception for latency-sensitive autonomous systems.

## I. INTRODUCTION

Autonomous Vehicles (AVs) have seen steady growth in recent years, with clear indications of rapid expansion in the coming decades. According to Litman [1], operational AVs are expected to be commercially available by 2030, and could become affordable and widespread between 2040 and 2060. Additionally, Moody [2] reports that nearly half of those surveyed perceives AVs as “very” or “somewhat” safe. Despite these positive projections, significant safety concerns persist, hindering the broader adoption of AV technology.

A major challenge in ensuring AV safety lies in object detection and situational awareness. Current AV systems usually rely on onboard sensors (e.g., cameras, LiDAR, radar) to detect and classify objects, but they are inherently limited by occlusions, blind spots, and sensor noise, which can cause them to misinterpret their surroundings [3], [4]. These limitations become particularly pronounced in complex and dynamic environments, such as parking lots and intersections,

where poor visibility or unexpected object movements can lead to misclassifications and collisions. Although advancements in computer vision and LiDAR technologies have improved detection capabilities, these systems remain insufficient for ensuring the high level of accuracy and reliability required for consistent safety. The reliance on isolated sensor data from a single AV limits the system’s ability to generate a comprehensive understanding of its surroundings, particularly when visibility is obstructed.

The real-world implications of these shortcomings are demonstrated by the National Highway Traffic Safety Administration (NHTSA). The NHTSA’s ongoing investigation highlights nearly 1,000 accidents involving Tesla’s autopilot features between 2018 and 2023, with over two dozen fatalities [5]. Many of these accidents were due to failures in object classification, such as a notable case where a Tesla vehicle misidentified a truck as a cloud in the sky. The report further reveals that approximately 20% of these accidents occurred with stationary objects, highlighting the limitations of single-vehicle perception and the risks present even in low-speed environments. These failures emphasize the critical need for enhanced object detection and decision-making capabilities in AVs to ensure their safe operation.

A major factor contributing to these limitations is the sensing modality used in many commercial AVs. While advanced perception systems often integrate LiDAR and radar, camera-only sensing remains widespread due to its lower cost and energy efficiency. For instance, Tesla has eliminated LiDAR and radar from its Autopilot and Full Self-Driving platforms, relying exclusively on camera-based inputs [6]. This shift is driven by the declining cost of high-resolution optical sensors and the scalability of vision-based deep learning. However, relying solely on a single vehicle’s camera can result in blind spots and misclassifications, particularly in complex and occluded environments.

To mitigate the limitations of single-vehicle camera-only sensing, researchers have emphasized the importance of collaborative perception among multiple Connected Autonomous Vehicles (CAVs), where vehicles share sensory data to improve detection accuracy [7]. This multi-CAV cooperation can significantly enhance situational awareness by fusing data from different vantage points, therefore reducing occlusion-related errors, and improving overall safety. However, existing collab-

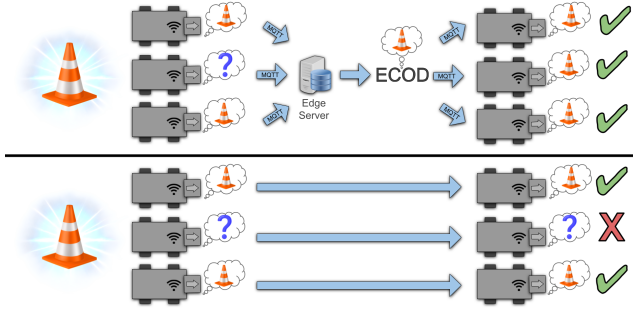


FIG. 1. COMPARISON OF COLLABORATIVE (TOP) AND INDIVIDUAL (BOTTOM) OBJECT DETECTION

orative frameworks predominantly rely on cloud computing, which introduces high latency and bandwidth constraints. While cloud-based systems are suitable for applications with moderate latency tolerance, they are inadequate for the real-time requirements of autonomous driving, where even milliseconds of delay can compromise safety [8].

To support the low-latency requirements of CAVs, a transition to edge computing is essential. Edge computing offers a distributed computing framework that brings computation closer to the data source at the network edge, reducing latency and bandwidth costs, and enhancing privacy [9], [10]. Recent studies have explored edge-assisted cooperative perception, but many rely on raw or feature-level data fusion, which is computationally intensive and bandwidth-heavy. Furthermore, existing methods lack real-world validation, as they are often evaluated in simulated environments rather than on physical testbeds.

Building on the advantages of edge computing, we propose Edge-Enabled Collaborative Object Detection (ECOD), a novel framework that leverages edge computing and multi-CAV collaboration for real-time, multi-perspective object detection (illustrated in Figure 1). ECOD includes two distinct algorithms: Perceptive Aggregation and Collaborative Estimation (PACE) and Variable Object Tally and Evaluation (VOTE). PACE aggregates object detection data from multiple CAVs at the edge server to enhance perception by providing a comprehensive view of the environment, especially in situations where individual CAVs have limited visibility. VOTE enhances the accuracy of object classification through a consensus-based voting mechanism that integrates data from multiple CAVs while accounting for CAV reputation and object visibility constraints. By integrating these algorithms, the ECOD framework enables more precise situational awareness while addressing the limitations of single-CAV perception.

We evaluate the effectiveness of the ECOD framework by deploying a hardware-based testbed consisting of four small camera-equipped robotic CAVs and an edge server. We consider two usecases: intersection mapping and parking lot tracking. We consider various traffic scenarios to assess ECOD’s collaborative performance. The results demonstrate a significant improvement in decision accuracy compared to single-CAV onboard systems, with ECOD outperforming

traditional onboard approaches by up to 75%. ECOD reduces latency by leveraging edge-based processing, ensuring real-time collaborative decision-making. In addition, it enhances scalability, making it suitable for multi-CAV networks in real-world autonomous systems. By integrating edge computing with multi-CAV collaboration, ECOD enables low-latency, real-time perception, contributing to the broader deployment of edge-assisted autonomous driving technologies.

The rest of the paper is organized as follows. In Section II, we review existing work in this domain. In Section III, we outline our framework and describe both PACE and VOTE. In Section IV, we present the experimental setup and evaluate the results. Section V summarizes our findings and outlines potential future research directions.

## II. RELATED WORK

In response to growing interest in intelligent and autonomous vehicles, Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Network (V2N), and Vehicle-to-Everything (V2X) communication protocols have been studied extensively in the past few years [18]. These protocols are crucial for enabling real-time situational awareness and improving decision-making in autonomous systems, which are often hindered by the limitations of single-vehicle, single-perspective object classification techniques. In this section, we review key research papers that have made substantial and relevant contributions in collaborative perception, edge computing for CAVs, and networking approaches that form the foundation for our work.

*Networking and V2X-Based Approaches.* Several studies have explored networking solutions for CAVs. Yee et al. [11] explored collaborative perception using a single vehicle with two cameras to mimic multiple viewpoints to provide a framework for V2X object detection. However, this study does not achieve true multi-vehicle perception, as its setup is limited to a single-vehicle testbed with two mounted cameras (multi-sensor fusion) rather than V2V data exchange. Li et al. [13] introduced a collaborative paradigm that leverages LiDAR data from autonomous vehicles to generate real-time 3D maps of multi-story parking garages, but focus on static environments. D’Ortona et al. [14] expanded upon existing inter-vehicle communication solutions by proposing the use of the MQTT protocol for communication between vehicles and vulnerable road users (such as pedestrians and cyclists) using Bluetooth Low Energy (BLE). Their approach assumes that all road users are equipped with BLE-enabled devices, limiting its scalability. Other related work involving MQTT in V2X systems includes approaches by Shin and Jeon [19], Affia and Matulevičius [20], and Hadded et al. [21]. However, these studies each explore the applications of MQTT in solving specific problems in autonomous vehicles (distributed software updates, traffic light perception, and the impact of cybersecurity threats on MQTT in autonomous vehicle systems,

TABLE I. COMPARISON WITH EXISTING RESEARCH

	Studies							
	[11]	[12]	[13]	[14]	[15]	[16]	[17]	ECOD
Collaborative Perception			✓		✓	✓	✓	✓
Edge Computing		✓			✓	✓	✓	✓
MQTT Protocol				✓				✓
V2N Interaction		✓	✓	✓	✓	✓	✓	✓
Physical Testbed	✓	✓	✓	✓				✓

respectively). Moreover, these approaches did not leverage the benefits of edge computing.

*Collaborative Perception in CAVs.* The need for robust collaborative perception spans a variety of domains, from battlefield IoT systems [22] to connected autonomous vehicles. Prior research has explored multi-agent perception using various levels of data fusion. Luo et al. [15] proposed EdgeCooper, a LiDAR-based cooperative perception framework, where raw LiDAR data from multiple vehicles is transmitted to an edge server for fusion (raw-level fusion). However, this approach is bandwidth-intensive and requires extensive edge processing, making it less practical. Similarly, Liu et al. [16] presented EdgeSharing, which constructs a 3D feature map to facilitate collaborative localization and object sharing in urban environments, while Song et al. [17] focuses on sensor noise estimation and fusion in vehicular communication networks. These methods employ feature-level fusion, reduce bandwidth usage compared to raw fusion but still require increases computational overhead and demands significant network resources.

Zhang et al. [23] developed EMP, utilizing LiDAR-based sensing with an object-level fusion, which reduces data transmission requirements by exchanging only detected objects. While object-level fusion is more efficient, EMP primarily relies on LiDAR-based sensing, which may not generalize well to diverse sensor modalities such as cameras and radar. Additionally, EMP lacks mechanisms to handle inconsistencies in multi-vehicle detections. Wang et al. [24] proposed V2VNet, which employs intermediate feature fusion and 3D convolution to aggregate LiDAR-based data from nearby vehicles. Lin et al. [25] introduced V2VFormer and Xu et al. [26] developed V2X-ViT, both utilizing transformer-based architectures to capture spatial dependencies and integrate multi-vehicle perspectives. While these approaches achieve high accuracy in simulated urban environments using platforms such as SUMO, CARLA, and NS3, they often require high bandwidth for feature transmission and lack physical testbed validation. In contrast, ECOD performs lightweight object-level fusion on real hardware, enabling low-latency inference with practical feasibility. Our work complements these state-of-the-art methods by providing a modular, testbed-validated edge-based perception framework suitable for deployment in constrained and heterogeneous environments.

*Edge Computing for Autonomous Systems.* Edge computing is increasingly leveraged to support low-latency, real-

time decision-making [27], [28]. Prior work on edge-based resource management and offloading in vehicular systems has addressed challenges such as privacy [29] and computation placement [30]. He et al. [12] proposed an edge-enabled C-V2X (cellular vehicle-to-everything) communication framework, where each vehicle frequently broadcasts motion data, such as speed and heading, to nearby vehicles. However, they assumed that all vehicles in the vicinity are equipped with V2X technology such as a configurable digital BIOS and intelligent transponders, which limits the framework's applicability in current mixed-vehicle environments. Moreover, their framework does not support collaborative object detection, focusing instead on individual vehicle motion awareness. Other edge-based studies, such as EdgeCooper [15] and EdgeSharing [16], utilize edge servers for processing, but primarily rely on feature-heavy or raw sensor fusion, which can introduce latency bottlenecks. ECOD addresses these challenges by performing object-level fusion at the edge, reducing network overhead while maintaining real-time performance. Additionally, existing edge-based frameworks do not explicitly address communication delays or object detection inconsistencies across multiple vehicles. ECOD's VOTE algorithm introduces a reputation-based voting mechanism to mitigate discrepancies in classification, ensuring more robust consensus-driven perception.

Despite progress in collaborative perception and edge computing, existing methods suffer from high bandwidth consumption, lack of real-world validation, and computational inefficiencies. Our research addresses these limitations. Table I summarizes the key differences between ECOD and existing methods, demonstrating its practical feasibility for edge-assisted collaborative perception.

### III. EDGE-ENABLED COLLABORATIVE OBJECT DETECTION (ECOD)

In this section, we introduce the ECOD framework, designed to enable groups of CAVs to collaborate in real-time for multi-perspective object detection using edge computing. ECOD enhances perception accuracy, mitigates sensor occlusions and blind spots, and reduces latency by fusing object detection results from multiple CAVs. Unlike single-CAV perception, which is limited by localized sensor coverage, ECOD integrates collaborative intelligence to improve detection robustness in dynamic environments.

At its core, ECOD provides a pipeline for reliable low-latency data transmission between CAVs and an edge server,

employing two key algorithms for determining verdicts: Perceptive Aggregation and Collaborative Estimation (PACE) and Variable Object Tally and Evaluation (VOTE). These algorithms enable robust and scalable object detection, ensuring accurate classification decisions even in challenging urban environments.

#### A. System Overview

ECOD consists of a two-layer architecture comprising CAVs and an Edge Server. The system operates as follows:

- **CAV Layer:** Each CAV is equipped with onboard sensors (e.g., cameras, LiDAR) that detect objects in its surroundings and generate preliminary object classifications.
- **Edge Layer:** The detected object data is transmitted to a nearby edge server, which aggregates multi-CAV detections, matches objects, and applies collaborative filtering techniques. The edge server processes fused data using the PACE and VOTE algorithms to reach consensus on object classification and positioning, reducing errors caused by individual sensor limitations.

Unlike cloud-based approaches, ECOD ensures low-latency decision-making by processing data at the network edge, enabling real-time object detection for dynamic traffic settings.

The networking module forms the backbone of the ECOD framework, providing the infrastructure for continuous data exchange between CAVs and the edge server. To ensure efficient communication, CAVs connect to the edge server via a dedicated Wi-Fi network hosted by the server itself. This configuration ensures secure and stable high-bandwidth data transmission. To facilitate real-time communication, ECOD utilizes the MQTT protocol, which follows a publisher/subscriber model. This allows multiple CAVs to simultaneously transmit data to the edge server. In addition, the server uses MQTT when returning its classification verdicts to all CAVs in the area. Data is exchanged at the object level, as opposed to feature-level or raw-level, in order to minimize latency and reduce the computational load on the edge server.

#### B. Perceptive Aggregation and Collaborative Estimation (PACE)

The PACE algorithm is designed for multi-CAV cooperative perception, particularly in scenarios where individual CAVs may not have a direct line of sight to the same object, such as in parking lots or urban intersections. By leveraging edge computing, PACE allows CAVs to share detected object information with an edge server, which then compiles these detections into a unified global perception map. This ensures that CAVs can perceive objects beyond their direct line of sight, enhancing situational awareness and navigation efficiency. PACE follows a two-stage process, involving both CAV-level detection and edge-level object matching and mapping.

**PACE: CAV Component.** Each CAV operates a local PACE client algorithm (shown in Algorithm 1) to independently detect objects and estimates their properties. The client algorithm runs locally on a V2N-enabled CAV which continuously

---

#### Algorithm 1: PACE CAV Pseudocode

---

**Data:** Camera input (Angle per Pixel  $\gamma$ , field of view), local computer vision model  $\mathcal{M}$ , GPS  $(x_v, y_v, \theta_v)$  for CAV  $v \in V$

**Result:** Published object detections with positions and labels

```

1 mqtt.subscribe("global_detections"); /* CAV
   subscribes to receive final object map from the edge
   */;
2  $\Omega_v \leftarrow \emptyset$ ; /* Object detection list */
3 while true do
4   /* Detect objects using the local model */
5    $\mathcal{O} \leftarrow \mathcal{M}.detect()$ ; /* Get object labels,
   confidence scores, and bounding box params. */
6   /* Compute global position for each detected
   object */
7   foreach object  $\omega \in \mathcal{O}$  do
8      $w_\omega \leftarrow x_\omega^{\max} - x_\omega^{\min}$ ; /* Bounding box width */
9      $s_\omega \leftarrow \omega.size$ ; /* Estimated true object size */
10     $\alpha \leftarrow \gamma \times w_\omega$ ;
11     $d_\omega \leftarrow s_\omega / \tan(\alpha)$ ; /* distance */
12     $x_{center} \leftarrow x_\omega^{\min} + \frac{1}{2}w_\omega$ ;
13    /* Compute relative angle and global
   coordinates */
14     $\theta_\omega \leftarrow \theta_v - (\gamma \times x_{center})$ ;
15     $x_\omega \leftarrow x_v + d_\omega \times \cos(\theta_\omega)$ ;
16     $y_\omega \leftarrow y_v + d_\omega \times \sin(\theta_\omega)$ ;
17    /* Store object position */
18     $\omega.position \leftarrow (x_\omega, y_\omega)$ ;
19     $\Omega_v.append(\omega)$ ;
20  /* Publish detections at regular intervals */
21  if  $t - t_p \geq \tau$  then
22    mqtt.publish("detections",  $\Omega_v$ );
23    /* Clear object list after publishing */
24     $\Omega_v.clear()$ ;
25     $t_p \leftarrow t$ ; /* Update timestamp */

```

---

scans its surroundings using onboard sensors and a computer vision model  $\mathcal{M}$ . To enhance perception beyond its immediate field of view, the CAV subscribes to the global object map (global\_detections) via MQTT to receive the aggregated global perception map from the edge server (line 1). The CAV processes incoming camera data to detect objects using  $\mathcal{M}$ , which assigns each detected object  $\omega$  a classification label  $l_\omega$ , a confidence score  $c_\omega$ , and a bounding box with Cartesian coordinates  $b_\omega = (x_\omega^{\min}, y_\omega^{\min}, x_\omega^{\max}, y_\omega^{\max})$  (line 5). Using this bounding box, the CAV estimates the object's relative position in its own frame of reference by considering the camera's field of view, angle per pixel  $\gamma$ , and the estimated physical size  $s_\omega$  of the object.

Then, the estimated object position is transformed into global coordinates (i.e., the absolute position of the object) using the CAV's own global coordinates and its orientation angle  $\theta_v$  (obtained via GPS or an experimental localization

system), which represents the heading of the CAV in the global frame (lines 8-16). The CAV maintains a dynamic list  $\Omega_v$  of detected objects, including their global positions, classification labels, and confidence values (lines 18-19), and it publishes this data to the edge server every  $\tau$  seconds via MQTT (detections) (lines 21-25). This continuous perception process is repeated in real-time, ensuring that the edge server receives frequent updates from all CAVs, allowing for multi-perspective aggregation.

**PACE: Edge Server Component.** The edge server component of PACE (shown in Algorithm 2) acts as a fusion center, consolidating object detections and generating a refined global perception map. It manages interactions with multiple CAVs simultaneously by processing multi-CAV incoming data through the following steps. The server continuously listens for object detections from CAVs by subscribing to the detections topic via MQTT (line 1). As CAVs publish their detected objects, the server receives and updates a list of reported detections from all actively connected CAVs (lines 3-5). Every  $\tau$  seconds, the server processes the latest object detections and consolidates them into a unified detection list  $\Omega_{all}$ , which contains object labels, confidence scores, and estimated positions (lines 8-11). To construct the global map, the edge server associates these detected objects with real locations  $\rho \in \mathcal{R}$  by comparing the reported positions (lines 13-30). Objects reported within a distance threshold  $\delta$  of a given location  $\rho$  are grouped together into  $\mathcal{O}_\rho$  (lines 16-18). This is to reduce sensor-based inaccuracies (positional noise) and aligning observations from multiple viewpoints. For the detected object located near  $\rho$ , the edge server assigns the highest confidence label, the confidence score (greater weight to detections with higher confidence values), and the estimated position based on all contributing CAVs (lines 19-30). The edge server compiles this information into the global object map, *object\_map*, and publishes it via MQTT to the global\_detections topic, ensuring that all CAVs receive the updated global perception data in real-time (lines 31-34).

This approach is particularly useful in complex, multi-level, or highly obstructed environments where direct line-of-sight perception is often limited. By aggregating detections from multiple CAVs, PACE reduces individual sensor uncertainty and accelerates the object labeling process, enabling faster and more accurate global perception updates.

### C. Variable Object Tally and Evaluation (VOTE)

The goal of VOTE is to assess object labels in scenarios where many CAVs classify the same objects simultaneously, potentially assigning conflicting labels. VOTE facilitates a robust voting system to resolve discrepancies in diverse viewpoints and uncertain sensing environments by weighing the confidence scores of different labels based on CAV reputation and visibility parameters to generate a consensus label for each object. VOTE requires an unlabeled list of objects with known locations, which can be predefined (hard-coded) or detected

---

### Algorithm 2: PACE Edge Server Pseudocode

---

**Data:** List of actively-connected CAVs ( $V$ ), global object location map ( $\mathcal{R}$ ), distance threshold ( $\delta$ ), update interval ( $\tau$ )

**Result:** Global object map with assigned labels

```

1 mqtt.subscribe("detections"); /* Edge Server
  subscribes to CAV detections */
2 while true do
3   /* Receive and update list of client-mapped data
     from CAVs */
4   foreach CAV  $v \in V$  do
5      $\Omega_v \leftarrow \text{mqtt.getDetectedObjects}(v)$ ;
6   /* Periodically update the global object map */
7   if  $t - t_p \geq \tau$  then
8      $\Omega_{all} \leftarrow \emptyset$ ; /* A unified detection list */
9     foreach CAV  $v \in V$  do
10      foreach detected object  $\omega \in \Omega_v$  do
11         $\Omega_{all}.append(l_\omega, c_\omega, \omega.position)$ ;
12    /* Match detected objects to real locations */
13    foreach location  $\rho \in \mathcal{R}$  do
14      /* Find all detected objects within distance
        threshold  $\delta$  of  $\rho$  */
15       $\mathcal{O}_\rho \leftarrow \emptyset$ ;
16      foreach  $\omega \in \Omega_{all}$  do
17        if  $\text{distance}(\omega.position, \rho) \leq \delta$  then
18           $\mathcal{O}_\rho.append(\omega)$ ;
19      if  $\mathcal{O}_\rho \neq \emptyset$  then
20        /* Assign highest-confidence label */
21         $l_\rho \leftarrow \arg \max_l \sum_{\omega \in \mathcal{O}_\rho, l_\omega = l} c_\omega$ ;
22        /* Compute the confidence score */
23         $c_\rho \leftarrow \frac{\sum_{\omega \in \mathcal{O}_\rho} c_\omega^2}{\sum_{\omega \in \mathcal{O}_\rho} c_\omega}$ ;
24        /* Compute the average position */
25         $x_\rho \leftarrow \frac{1}{|\mathcal{O}_\rho|} \sum_{\omega \in \mathcal{O}_\rho} x_\omega$ ;
26         $y_\rho \leftarrow \frac{1}{|\mathcal{O}_\rho|} \sum_{\omega \in \mathcal{O}_\rho} y_\omega$ ;
27      else
28        /* No object detected at  $\rho$  */
29         $l_\rho \leftarrow \text{None}$ ;
30         $c_\rho \leftarrow 0$ ;
31    /* Publish the updated global object map */
32     $\Lambda \leftarrow \{(\rho, l_\rho, c_\rho, x_\rho, y_\rho) \mid \forall \rho \in \mathcal{R}\}$ ;
33    mqtt.publish("global_detections",  $\Lambda$ );
34     $t_p \leftarrow t$ ;

```

---

with technologies such as LiDAR. The VOTE algorithm has two distinct components, the CAVs and the edge server.

**VOTE: CAV Component.** Each CAV runs the VOTE client algorithm, which continuously scans its surroundings using a computer vision model  $\mathcal{M}$ . The VOTE client algorithm follows the same structure as the PACE client, with minor modifications (the VOTE client pseudocode is omitted for

brevity). Each CAV subscribes to “global\_verdicts” via MQTT to receive finalized object labels, as VOTE focuses on label agreement. For each detected object  $\omega$ , the CAV assigns a temporary label  $l_\omega$ , a confidence score  $c_\omega$ , and estimates its real position relative to its own frame of reference. The algorithm then cross-references detected objects with known object locations  $\mathcal{R}$ , determining which real object each detection corresponds to based on spatial proximity. The CAV then packages the object’s assigned temporary label, confidence, and estimated position, publishing the results to “vote\_detections” via MQTT to the edge server at regular intervals  $\tau$ . The CAV component ensures that object detections from different viewpoints are consistently reported, enabling collaborative classification across multiple vehicles.

**VOTE: Edge Server Component.** The edge server component of VOTE is introduced in Algorithm 3. It can run on an edge server or any CAV acting as an aggregator, and it interacts with multiple CAVs simultaneously to process classification reports. VOTE begins by initializing each CAV with a default reputation score, which corresponds to the proportion of correct labels issued by each CAV (lines 2-3). Then, the algorithm creates an empty list of dictionaries, each corresponding to a known object location (lines 5-7). It then continuously processes incoming label reports from the CAVs (lines 8-21).

The collaborative decision-making process of VOTE is based on calculating an aggregated confidence score for each proposed label for each object using inputs from multiple CAVs (lines 11-13). VOTE considers three factors to calculate the aggregated confidence score: the reliability of the CAV, the confidence in the detected object label, and the visibility of the object from the CAV’s perspective.

The aggregated confidence score for a given object at location  $\rho$  with a temporary detected label  $l$  (i.e., proposed label) is calculated as follows:

$$S_{\rho,l} = \sum_{v \in V} \sum_{\omega \in \Omega_v} \underbrace{f_l(\omega)}_{\text{label match}} \cdot \underbrace{r_v}_{\text{reputation}} \cdot \underbrace{c_\omega}_{\text{confidence}} \cdot \underbrace{k(\rho, v)}_{\text{visibility}} \quad (1)$$

where  $V$  is the set of all CAVs,  $\Omega_v$  is the set of all proposed object labels reported by CAV  $v$ ,  $f_l(\omega)$  is an indicator function that equals 1 if  $l_\omega = l$ , and 0 otherwise,  $r_v$  is the reputation score of CAV  $v$ , representing the historical reliability of its detections,  $c_\omega$  is the individual confidence score for the detected temporary label of object  $\omega$ , and  $k(\rho, v)$  is the visibility score of the object at  $\rho$  relative to CAV  $v$ . The visibility score  $k(\rho, v)$  accounts for both distance and angular positioning and is computed as:

$$k(\rho, v) = p_d \left(1 - \frac{d_{pv}}{d_{max}}\right) + (1 - p_d) \left(\frac{\theta_{pv}}{360^\circ}\right) \quad (2)$$

where  $p_d$  is a weight parameter that balances the impact of distance and angle on visibility,  $d_{pv}$  is the distance between location of the object  $\rho$  and the CAV  $v$ ,  $d_{max}$  is the maximum detection range, and  $\theta_{pv}$  is the absolute value of the angle created by a line segment from the CAV’s camera to the

---

### Algorithm 3: VOTE Edge Server Pseudocode

---

**Data:** List of actively-connected CAVs ( $V$ ), global object location map ( $\mathcal{R}$ ), update interval ( $\tau$ )

**Result:** Consensus object labels

```

1 mqtt.subscribe("vote_detections"); /* Edge
  Server subscribes to CAV detections */
2 foreach CAV  $v \in V$  do
3    $r_v = 0.5$ ; /* Initial reputation score */
4 /* A nested dictionary to keep track of label votes */
5  $S = \text{new Dictionary}$ ;
6 foreach location  $\rho \in \mathcal{R}$  do
7    $S[\rho] = \text{new Dictionary}$ ;
8 while true do
9   /* Receive label votes from CAVs */
10  foreach CAV  $v \in V$  do
11     $\Omega_v \leftarrow \text{mqtt.getDetectedObjects}(v)$ ;
12    foreach  $\rho, l$  in  $\Omega_v$  do
13       $S[\rho][l] += (r_v \cdot c_\omega \cdot k(\rho, v))$ ;
14 /* Periodically determine a verdict (consensus
  label) */
15 if  $t - t_p \geq \tau$  then
16   /*Set the list of verdicts according to Eq. 3*/
17    $\Lambda \leftarrow \emptyset$ ; /* Initialize verdicts set */
18   foreach location  $\rho \in \mathcal{R}$  do
19     /* Select label with highest confidence
  score */
20      $\Lambda_\rho \leftarrow \arg \max_l S[\rho][l]$ ;
21   /* Publish final consensus labels */
22   mqtt.publish("global_verdicts",  $\Lambda$ );
23   /* Update CAV reputation scores */
24   foreach CAV  $v$  in  $V$  do
25      $r_v \leftarrow r_v + \Delta r_v$ ;
26   /* Update last processed time */
27    $t_p \leftarrow t$ ;

```

---

object, relative to the CAV’s camera orientation (representing deviation from the camera center). By incorporating both distance and angular clarity, VOTE prioritizes data from CAVs with better visibility of the object, ensuring that the final label decision is based on the most reliable observations.

At a specified verdict interval, VOTE determines the final consensus label  $\lambda_\rho$  for each object at  $\rho \in \mathcal{R}$  (line 18).

$$\lambda_\rho = \arg \max_l S_{\rho,l} \quad (3)$$

After determining the final verdict, the edge server broadcasts the verdict to each CAV via an MQTT one-to-many query (line 19). This ensures that all CAVs update their local perception models with the agreed-upon object classifications.

To maintain fairness and improve the reliability of future decisions, the server adjusts each CAV’s reputation score based

on the correctness of its previous label contributions (lines 21-22). This reputation update parameter  $\Delta r_v$  is calculated as:

$$\Delta r_v = \text{cap} \left( \frac{\# \text{ correct} - \# \text{ incorrect}}{\# \text{ objects}}, 30, 100 \right) \quad (4)$$

The function ensures that reputation scores remain between 30 and 100, preventing extreme fluctuations and maintaining stability in trust levels. This adaptive reputation mechanism ensures that CAVs with a history of accurate detections gain more influence over future decisions, while unreliable CAVs contribute less to the consensus process.

Together, PACE and VOTE form the backbone of ECOD's collaborative perception framework. While PACE enhances situational awareness via multi-view fusion, VOTE ensures classification consistency through trust-aware consensus, jointly enabling real-time, accurate perception for CAVs.

#### D. Computation and Communication Analysis

In time-sensitive and bandwidth-constrained edge systems, such as autonomous vehicle networks, it is crucial for algorithms to maintain both low computational and communication complexity.

The CAV components of both PACE and VOTE run in  $O(|\Omega_v|)$  time, where  $|\Omega_v|$  is the number of objects detected by the CAV. Most of the computation time on the client is attributed to the computer vision model, which can be independently optimized for autonomous vehicle applications. The edge server components of both algorithms run in  $O(|V| \times |\Omega|)$  time, where  $|V|$  is the number of CAVs connected to the server and  $|\Omega|$  is the maximum number of objects labeled by each CAV (an upper bound on  $|\Omega_v|$ ). Note that for sufficiently small  $|V|$ , the number of detected objects per CAV tends to be high, making  $|\Omega| \approx |\mathcal{R}|$ , where  $\mathcal{R}$  is the set of all known object locations. On the other hand, as  $|V|$  increases,  $|\Omega|$  may decrease due to visual obstructions between CAVs, leading to fewer detections per vehicle.

In terms of communication, ECOD minimizes bandwidth usage by transmitting only object-level summaries (labels, confidence scores, coordinates), unlike raw or feature-level fusion approaches. This makes the framework practical for bandwidth-constrained edge deployments.

Overall, VOTE and PACE scale very well in dynamic scenarios involving many vehicles and objects due to relatively low time complexity. Additionally, since all computationally expensive vision-related tasks are delegated to the CAVs, the edge server maintains a low processing load, ensuring real-time operation without introducing significant overhead.

### IV. EXPERIMENTAL RESULTS

This section describes the experimental setup and the experimental results of evaluating ECOD.

#### A. Experimental Setup

We create a comprehensive experimental testbed consisting of four robotic CAVs equipped with cameras, an edge server, and a controlled test environment. The CAVs are built using

the SunFounder Picar-X kit, each running on a Raspberry Pi 4 Model B, the same hardware used for the edge server. The testbed is configured on a 60" by 40" grid divided into 2"x2" cells to standardize object placement and ensure consistent testing conditions across experiments. Accuracy is defined as the proportion of correctly identified object detection verdicts over the course of multiple experimental cycles.

**Object Detection Tools.** Each CAV utilizes the object detection capability provided by the open-source Vilib computer vision library [31], built on top of TensorFlow Lite [32] and OpenCV [33]. Vilib works in real time to detect and label objects from a live camera feed. Vilib is selected for compatibility with Raspberry Pi and testbed constraints. Our focus is on evaluating the collaborative gains, independent of specific object detector performance. Vilib's object detection module, trained on the Common Objects in Context (COCO) dataset [34], can detect 80 classes of common household objects. The library also provides a QR code reader module, which is used in the PACE testbed. We extend Vilib's functionality by adding a feature that allows programmers to have more precise control over the object labels generated by the library's object detection and the QR reading module, integrating this feature into the onboard object detection module.

**Communication Setup.** We use MQTT to facilitate data transmission in real-time between CAVs and the edge server. The edge server runs the Eclipse Mosquitto MQTT broker (version 2.0.18) [35]. The CAVs and the edge server implement the Eclipse Paho MQTT library for Python integration [36] to facilitate seamless real-time communication. The testbed relies on a stable local Wi-Fi network and MQTT for communication. While effective for prototyping, we note that real-world vehicular networks, such as LTE-V2X and 5G NR-V2X, exhibit variable latency, bandwidth constraints, and potential packet loss, which may impact performance. ECOD is compatible with these protocols and can be extended using SLAM/GNSS fusion. All components of the system run on Python 3.11.

**Benchmark.** We implement a single-CAV object detection method as a benchmark for comparison. In this baseline, each CAV independently detects nearby objects and reports its observations to the edge server without collaboration. This setup isolates the benefits of multi-vehicle coordination. Notably, the low accuracy of this benchmark is not due to shortcomings in the computer vision model, but rather due to occlusions, blind spots, and limited visibility inherent to single-vehicle sensing—challenges that ECOD is explicitly designed to overcome.

**Localization.** For the purposes of our experiments, we assume that each vehicle knows its own global position, represented as a predefined two-dimensional Cartesian coordinate pair corresponding to a surface grid of two-inch squares. In practice, this localization would ideally be determined by GPS or triangulation from nearby devices, as explored in prior work [16]. To simplify our experiments and control localization errors,



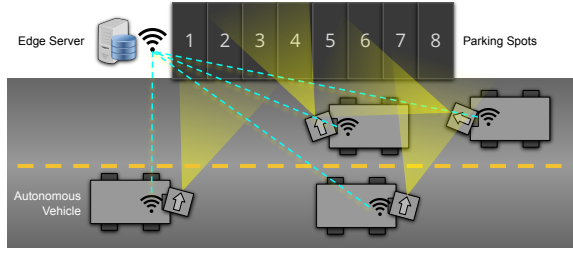


FIG. 2. PACE TESTBED DIAGRAM

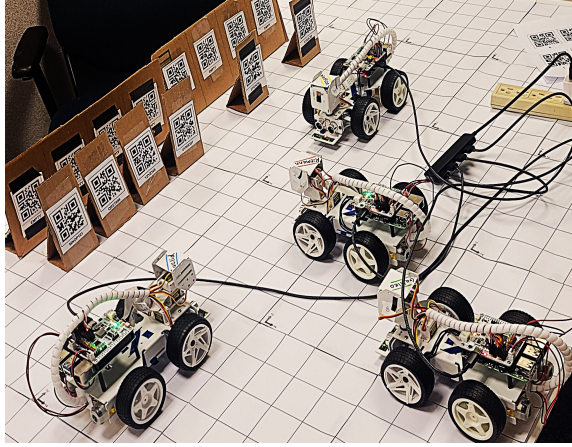


FIG. 3. PACE TESTBED PHOTOGRAPH

we hard-code the global positions of the CAVs.

**PACE Testbed Setup.** To evaluate the PACE algorithm, we develop a parking lot testbed. We utilize optical cameras (480p by 640p) and simulate parked vehicles using QR codes. Each QR code (2.5" x 2.5") is mounted on a cardboard rectangle (2.75" wide by 5.5" tall) to represent a parked vehicle. These QR codes are arranged into one of six distinct parking configurations with varying levels of vehicle density. To ensure varied perspectives, the CAVs are positioned in three unique locations for each parking lot configuration. Figure 2 illustrates a diagram of the testbed, while Figure 3 displays a photograph of the setup for the four CAVs interacting with an edge server to collaboratively label the eight parking spots. For each distinct testbed configuration, we execute the experimental code, (available [37]), recording the outcomes of 200 sequential verdicts. We then calculate the average accuracy for PACE and the benchmark method.

**VOTE Testbed Setup.** To evaluate the VOTE algorithm, we develop a traffic intersection testbed. This setup contains three objects (a computer mouse, a solo cup, and an orange ball) placed at the center of the intersection. The CAVs view these objects from different angles, allowing for collaborative decision-making through voting. Each of these objects is discernible by our Vilib object detection model trained on the COCO dataset [31], [34]. The relatively consistent detectability of these objects allows us to control the variable of onboard sensor efficacy, and instead focus on the efficacy of VOTE. We consider 0.5 default reputation score for each

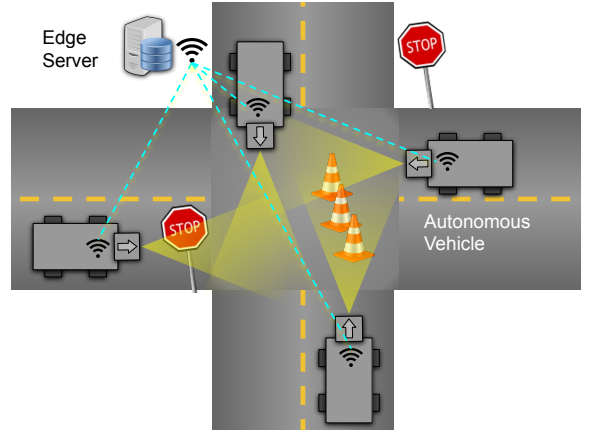


FIG. 4. VOTE TESTBED DIAGRAM

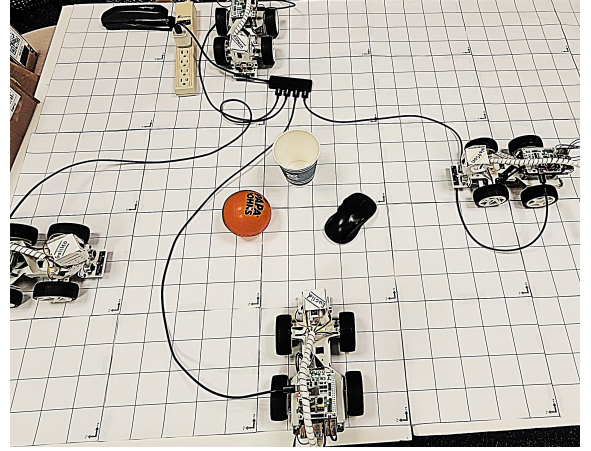


FIG. 5. VOTE TESTBED PHOTOGRAPH

CAV to track their respective detection histories. For flexibility when applying our server-side broker algorithm, we record all VOTE data as unprocessed object detection annotations from each CAV and then analyze the data using a post-synchronous adaptation of VOTE. For each of the three testbed configurations, we record 1,000 object detection cycles from each CAV, with verdicts being processed every 120ms. An example diagram of a VOTE testbed setup is illustrated in Figure 4, while Figure 5 presents a photograph of an actual VOTE testbed setup.

### B. Analysis of Results

Tables II and III summarize the results of PACE and VOTE, respectively. For each trial, the tables list the accuracy of the ECOD framework, the accuracy of the benchmark, and the resulting improvement in accuracy (ECOD minus benchmark). Figures 6 and 7 succinctly compare the accuracy of PACE and VOTE versus the benchmark. Based on the results, PACE consistently outperforms the benchmark in accuracy across all test configurations. On average, PACE achieves an accuracy of 97.1%, compared to 25.9% for the benchmark, reflecting a substantial improvement of 71.2%. This improvement highlights PACE's effectiveness in enhancing object detection



TABLE II. PACE TEST RESULTS

CAV Setup #	PACE Accuracy (%)	Benchmark Accuracy (%)	Difference (%)
1	<b>92.9</b>	<b>23.8</b>	<b>+69.1</b>
2	<b>98.7</b>	<b>29.4</b>	<b>+69.3</b>
3	<b>99.6</b>	<b>24.6</b>	<b>+75.0</b>
All trials	<b>97.1</b>	<b>25.9</b>	<b>+71.2</b>

TABLE III. VOTE TEST RESULTS

CAV Setup #	VOTE Accuracy (%)	Benchmark Accuracy (%)	Difference (%)
1	<b>99.4</b>	<b>38.5</b>	<b>+60.9</b>
2	<b>63.1</b>	<b>15.8</b>	<b>+47.3</b>
3	<b>99.4</b>	<b>24.8</b>	<b>+74.5</b>
All trials	<b>87.3</b>	<b>26.4</b>	<b>+60.9</b>

accuracy in scenarios where single-CAV perception is limited by occlusions and blind spots. This is attributed to the PACE algorithm's collaborative perception, which leverages multiple CAVs share their detections to achieve more accurate and reliable object detection.

Similarly, VOTE shows a marked improvement over the benchmark. VOTE achieves an average accuracy of 87.3%, compared to 26.4% for the benchmark, resulting in a 60.9% improvement. This improvement reflects the VOTE algorithm's robust reputation-weighted voting mechanism, which aggregates detections from multiple CAVs to resolve ambiguities and conflicts in object labels. These results confirm the advantage of collaborative decision-making, especially in scenarios where multiple perspectives can contribute to a more accurate overall assessment.

In our prototype testbed, each perception-decision cycle completes in approximately every 100–150ms. This interval includes object detection on the CAV, MQTT transmission to the edge server, collaborative aggregation via PACE or VOTE, and publication of the consensus result.

The experimental results achieved by both PACE and VOTE validate the effectiveness of ECOD in significantly improving object detection accuracy and decision-making in complex, dynamic environments. By leveraging collaborative perception and real-time data aggregation through edge computing, the ECOD framework overcomes the key limitations of single-CAV perception, demonstrating its potential to enhance safety and reliability of autonomous vehicle systems.

## V. CONCLUSION

We introduced the ECOD framework, an edge-enabled collaborative object detection framework for CAVs, featuring two novel algorithms, PACE and VOTE. PACE enables efficient, real-time object classification by leveraging cooperative perception in occluded or complex environments, while VOTE facilitates consensus-based label agreement among multiple CAVs via confidence-weighted voting, enhancing detection

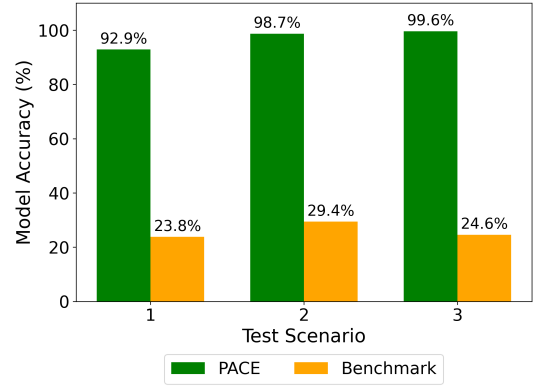


FIG. 6. MODEL ACCURACY OF PACE

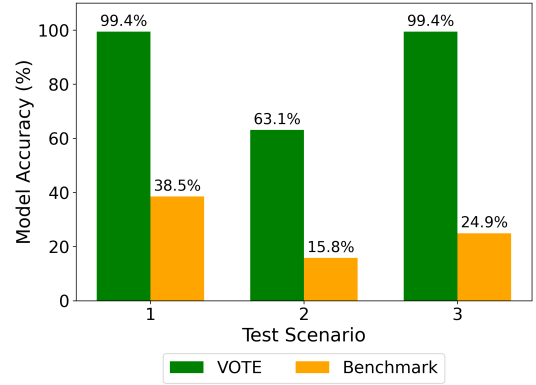


FIG. 7. MODEL ACCURACY OF VOTE

reliability. Experimental evaluations on a multi-CAV testbed demonstrated that both algorithms significantly enhance object detection accuracy compared to a traditional single-CAV benchmark, validating the benefits of collaborative perception in autonomous systems. By advancing edge-enabled cooperation, ECOD contributes to the broader goal of safer and more reliable autonomous vehicle systems, paving the way for scalable, intelligent perception frameworks in next-generation smart mobility. These findings show the potential of distributed cooperative systems, where perception and decision-making are shared across many intelligent vehicles at the edge, reducing reliance on centralized infrastructure and enhancing robustness. Future work will focus on improving system scalability, optimizing computational efficiency for real-time inference, and incorporating adaptive learning mechanisms to refine object classification over time. We aim to expand ECOD's applicability to more challenging environments, such as high-traffic intersections and urban driving scenarios, with mixed vehicles where dynamic collaboration can significantly enhance both safety and efficiency. We also seek to explore adaptive collaboration strategies, where CAVs dynamically adjust their participation based on visibility, confidence, and task priorities.

**Acknowledgments.** This work was supported in part by the National Science Foundation under Grants No. 2050879 and 2145268.

## REFERENCES

- [1] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning," 2020.
- [2] J. Moody, N. Bailey, and J. Zhao, "Public perceptions of autonomous vehicle safety: An international comparison," *Safety science*, vol. 121, pp. 634–650, 2020.
- [3] H. Chu, H. Liu, J. Zhuo, J. Chen, and H. Ma, "Occlusion-guided multi-modal fusion for vehicle-infrastructure cooperative 3D object detection," *Pattern Recognition*, vol. 157, p. 110939, 2025.
- [4] Z. Xiao, J. Shu, H. Jiang, G. Min, H. Chen, and Z. Han, "Overcoming occlusions: perception task-oriented information sharing in connected and autonomous vehicles," *IEEE Network*, vol. 37, no. 4, pp. 224–229, 2023.
- [5] N. H. T. S. Administration, "Additional information regarding EA22002," <https://static.nhtsa.gov/odi/inv/2022/INCR-EA22002-14496.pdf>, "Accessed: 2024-07-08".
- [6] I. Tesla, "Tesla vision update: Replacing ultrasonic sensors with tesla vision," <https://www.tesla.com/support/transitioning-tesla-vision>, 2022, "Accessed: 2025-05-18".
- [7] A. Shetty, M. Yu, A. Kurzhanskiy, O. Grembek, H. Tavafoghi, and P. Varaiya, "Safety challenges for autonomous vehicles in the absence of connectivity," *Transportation research part C: emerging technologies*, vol. 128, p. 103133, 2021.
- [8] S. Lu and W. Shi, "Vehicle computing: Vision and challenges," *Journal of Information and Intelligence*, vol. 1, no. 1, pp. 23–35, 2023.
- [9] B. B. Thapa and L. Mashayekhy, "Latency-aware service placement for GenAI at the edge," in *Proc. of the Disruptive Technologies in Information Sciences VIII*, vol. 13058. SPIE, 2024, pp. 137–150.
- [10] E. F. Maleki, W. Ma, L. Mashayekhy, and H. La Roche, "QoS-Aware content delivery in 5G-enabled edge computing: Learning-based approaches," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9324–9336, 2024.
- [11] R. Yee, E. Chan, B. Cheng, and G. Bansal, "Collaborative perception for automated vehicles leveraging vehicle-to-vehicle communications," in *Proc. of the 2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1099–1106.
- [12] Y. He, B. Wu, Z. Dong, J. Wan, and W. Shi, "Towards C-V2X enabled collaborative autonomous driving," *IEEE Transactions on Vehicular Technology*, 2023.
- [13] B. Li, L. Yang, J. Xiao, R. Valde, M. Wrenn, and J. Leflar, "Collaborative mapping and autonomous parking for multi-story parking garage," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1629–1639, 2018.
- [14] C. D'Ortona, D. Tarchi, and C. Raffaelli, "Open-source MQTT-based end-to-end IoT system for smart city scenarios," *Future Internet*, vol. 14, no. 2, p. 57, 2022.
- [15] G. Luo, C. Shao, N. Cheng, H. Zhou, H. Zhang, Q. Yuan, and J. Li, "EdgeCooper: Network-aware cooperative LiDAR perception for enhanced vehicular awareness," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 1, pp. 207–222, 2024.
- [16] L. Liu and M. Gruteser, "EdgeSharing: Edge assisted real-time localization and object sharing in urban streets," in *Proc. of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.
- [17] R. Song, A. Hegde, N. Senel, A. Knoll, and A. Festag, "Edge-aided sensor data sharing in vehicular communication networks," in *Proc. of the 2022 IEEE 95th Vehicular Technology Conference (VTC)*. IEEE, 2022, pp. 1–7.
- [18] S. Malik, M. A. Khan, and H. El-Sayed, "Collaborative autonomous driving—a survey of solution approaches and future challenges," *Sensors*, vol. 21, no. 11, p. 3783, 2021.
- [19] Y. Shin and S. Jeon, "MQTree: Secure OTA protocol using MQTT and MerkleTree," *Sensors*, vol. 24, no. 5, p. 1447, 2024.
- [20] A.-A. O. Affia and R. Matulevičius, "Securing an MQTT-based traffic light perception system for autonomous driving," in *Proc. of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2021, pp. 255–260.
- [21] M. Hadded, G. Lauras, J. Letailleur, Y. Petiot, and A. Dubois, "An assessment platform of cybersecurity attacks against the MQTT protocol using SIEM," in *Proc. of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2022, pp. 1–6.
- [22] R. Sadik and L. Mashayekhy, "Collaborative object labeling in IoBT: a distributed approach for enhanced battlefield perception," in *Proc. of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications VI*, vol. 13051. SPIE, 2024, pp. 38–48.
- [23] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao, "EMP: Edge-assisted multi-vehicle perception," in *Proc. of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 545–558.
- [24] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, "V2VNet: Vehicle-to-vehicle communication for joint perception and prediction," in *Proc. of the Computer vision—ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part II 16*. Springer, 2020, pp. 605–621.
- [25] C. Lin, D. Tian, X. Duan, J. Zhou, D. Zhao, and D. Cao, "V2VFormer: Vehicle-to-vehicle cooperative perception with spatial-channel transformer," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 2, pp. 3384–3395, 2024.
- [26] R. Xu, H. Xiang, Z. Tu, X. Xia, M.-H. Yang, and J. Ma, "V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer," in *Proc. of the European conference on computer vision*. Springer, 2022, pp. 107–124.
- [27] J. Zhang, B. Thapa, and L. Mashayekhy, "FTFormer: Fault-tolerant layer offloading in edge-fog-cloud federated split learning," in *Proc. of the 9th IEEE International Conference on Fog and Edge Computing (ICFEC)*, 2025, pp. 19–26.
- [28] H. Bornholdt, K. Röbert, S. Schulte, J. Edinger, and M. Fischer, "A software-defined overlay networking middleware for a simplified deployment of distributed application at the edge," in *Proc. of the 40th ACM/SIGAPP Symposium on Applied Computing*, 2025, pp. 746–748.
- [29] W. Ma and L. Mashayekhy, "Privacy-by-design distributed offloading for vehicular edge computing," in *Proc. of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, ser. UCC'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 101–110. [Online]. Available: <https://doi.org/10.1145/3344341.3368804>
- [30] D. Bhatta and L. Mashayekhy, "Generalized cost-aware cloudlet placement for vehicular edge computing systems," in *Proc. of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019, pp. 159–166.
- [31] S. Inc., "Vilib computer vision library," <https://github.com/SunFounder/vilib>, "Accessed: 2024-06-20".
- [32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
- [33] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. of the 13th European Conference on Computer Vision, Part V 13*. Springer, 2014, pp. 740–755.
- [35] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.
- [36] Eclipse Foundation, "Paho-MQTT," <https://www.eclipse.org/paho/>, 2024, "Accessed: 2024-07-15".
- [37] E. Richards, "Edge-CAV code repository," <https://github.com/EverettRichards/Edge-CAV>, "Accessed: 2025-06-05".