Edge-Assisted Relevance-Aware Perception Dissemination in Vehicular Networks

Ruiqi Wang and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
E-mail: {ruw400, gxc27}@psu.edu

***Abstract*—Vehicles are equipped with various sensors such as LiDAR, which enable them to perceive the surrounding environment and enhance driver safety through advanced driver assistance systems. However, these sensors are limited by line-of-sight, preventing them from seeing beyond occlusions. One solution is to leverage the edge server which can collect and share perception data with other vehicles. Most existing research focuses on improve the performance of uploading perception data to the server, and the problem of perception dissemination remains largely unexplored, despite the challenges posed by the large volume of perception data and the limited wireless bandwidth. In this paper, we propose an edge-assisted relevance-aware perception dissemination system that collects perception data from multiple vehicles and selectively disseminates only the necessary data to appropriate vehicles. The necessity of dissemination is determined by evaluating the relevance of perception data, which quantifies the probability of potential collisions between corresponding objects. We then formulate and solve the relevance-aware perception dissemination problem whose goal is to maximize the relevance of disseminated data under bandwidth constraints. Extensive evaluation results demonstrate that our system can significantly enhance traffic safety while reducing the overall bandwidth consumption.**

I. Introduction

Autonomous driving and advanced driver assistance systems (ADAS) have rapidly evolved in recent years. These systems rely on various sensors to perceive their surroundings, and leverage artificial intelligence techniques to understand the traffic conditions and prevent accidents. However, the perception data generated by sensors on a single vehicle may be insufficient due to the limitations in their line of sight. A scenario shown in Fig. 1 illustrates this issue, where a pedestrian p is crossing the road behind truck D, while vehicle B is passing through the intersection. The pedestrian p is not visible to B's sensors because truck D blocks its view. In this case, relying solely on B's sensors may fail to detect the pedestrian, leading to a potential accident. Similar scenarios exist such as a vehicle's sensor may not detect an object that is around a corner or beyond its range.

This limitation can be overcome by aggregating and sharing perception data from multiple vehicles [1]–[3], and existing research has explored various ways to achieve this goal, such as cooperative perception through the edge server [4]–[6]. The edge server collects and shares perception data with other vehicles, enabling the creation of a traffic map based on all objects on the road. This map can inform vehicles of traffic conditions beyond their line of sight, allowing for better decision-making in autonomous driving and driver assistance systems to prevent potential accidents. However, existing research primarily focuses on the problem of uploading perception data with bandwidth and delay constraints [7]–[9], without addressing the issue of what perception data should be disseminated to which vehicle. A potential solution is to broadcast the entire traffic map to all vehicles, however, this approach has significant drawbacks. Most vehicles do not require the perception data as they are not involved in potential accidents. Processing and storing such unnecessary perception data consumes resources on vehicles where the computational resources are limited. Besides, too much irrelevant information may also overwhelm human drivers, causing distraction.

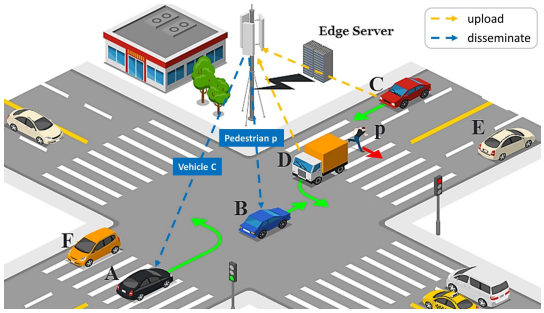


Fig. 1: A traffic scenario where the pedestrian p is relevant to vehicle B, and vehicle C is relevant to A.

An example can be seen in Fig. 1, the crossing pedestrian p is important (relevant) to vehicle B since B is moving forward and might hit the pedestrian in the near future. However, such information is not important (irrelevant) to vehicle A which will turn left and has no chance to hit p. Similarly, the perception data of C is irrelevant to B, but it is relevant to A since A is taking a left turn at the intersection. Therefore, instead of overwhelming the drivers with a large amount of irrelevant perception data which wastes resources, the edge server should only disseminate highly relevant data to the corresponding vehicles.

To address these issues, we propose an edge-assisted relevance-aware perception dissemination system that only disseminates necessary perception data to appropriate vehicles to reduce the bandwidth consumption. Our system estimates the relevance of the perception data based on the probability of potential collisions, and addresses the the perception dissemination problem by maximizing the total relevance of

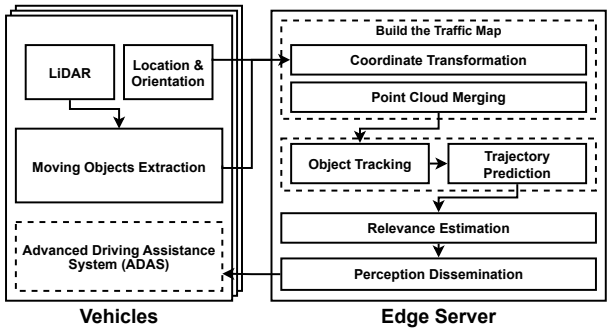


Fig. 2: A system overview

(a) (b) (c)

Fig. 3: (a) The raw point cloud generated by a LiDAR sensor. (b) Point cloud after removing ground points. (c) The points of moving objects (red) and static objects (black).

disseminated data while considering bandwidth constraints.

The main contributions of this paper can be summarized as follows. First, we introduce an edge-assisted relevance-based perception dissemination system that efficiently collects perception data from multiple vehicles and selectively disseminates only the relevant perception data to the appropriate vehicles. Second, we propose techniques for clustering vehicles and pedestrians based on their locations and orientations, resulting in reduced computational overhead for trajectory prediction. By only tracking representative entities within each cluster, we can significantly reduce the computational overhead. Third, we develop an efficient approach to measure the relevance of the perception data. We formulate and solve the relevance-aware perception dissemination problem with the goal of maximizing the overall relevance of disseminated data under bandwidth constraints. Finally, we conduct extensive evaluations to demonstrate the effectiveness and efficiency of our proposed system.

II. System Design

In this section, we introduce the system design of our edge-assisted relevance-aware perception dissemination system as shown in Fig. 2.

A. System Overview

Vehicles: LiDAR sensors mounted on vehicles are used for environment perception. These sensors have 360-degree field of view and can generate point cloud with detailed 3D information, enabling applications like path planning and augmented reality on the head-up display. Since it is impractical to transmit the extensive 3D point cloud data with millions of points via limited wireless bandwidth, we reduce the size of point cloud to save bandwidth. Specifically, for each frame generated by LiDAR sensors, we propose the *Moving Objects Extraction* module to reduce the point cloud by extracting only points corresponding to moving objects such as vehicles and pedestrians. Then, the reduced point cloud, along with the locations and orientations of the vehicles, are uploaded to the edge server.

Edge Server: After receiving the information uploaded by vehicles, the edge server uses the *Coordinate Transformation* and *Point Cloud Merging* modules to construct the comprehensive traffic map which includes objects on the road like vehicles and pedestrians. This map serves as a basis for *Object Tracking* and *Trajectory Prediction* modules, which can track and predict the movements of objects. Using the predicted trajectories, the *Relevance Estimation* module evaluates the relevance of the perception data by calculating the probability of potential collisions when the trajectories intersect. Then, based on the relevance, the relevance-aware perception dissemination problem is solved by the *Perception Dissemination* module, so that the edge server can determine whether the perception data should be disseminated and to which vehicle the data should be sent. Our goal is to maximize the total relevance of disseminated perception data under bandwidth constraints. That is, only perception data with higher relevance are disseminated to the corresponding vehicles, allowing ADAS and human drivers to gain a better understanding about the traffic conditions.

In the subsequent sections, we provide a detailed description of each module in our system.

B. Moving Objects Extraction

To save bandwidth, instead of uploading all LiDAR data, we propose to only upload the data related to important objects such as vehicles and pedestrians. To achieve this, we should extract these moving objects from the LiDAR sensor data locally. Although various 3D object detection algorithms based on deep neural networks [10]–[13] can be leveraged to identify the moving objects, they typically require significant computational resources which are not commonly available on vehicles. Instead, we propose a different approach to extract the points corresponding to objects.

First, as most points in the point cloud are from the ground plane which are irrelevant in traffic map construction, they should be removed to save bandwidth. Since LiDAR sensors are typically mounted on vehicles at a fixed height above the ground, it becomes straightforward to identify ground points by examining the z-axis coordinate of each point. Specifically, if the LiDAR sensor is positioned at a height of h above the ground, ground points should exhibit a z-coordinate of $-h$ in the LiDAR coordinate system. Consequently, the removal of ground points can be accomplished by filtering out the points with z-coordinates less than or equal to $-h + \epsilon$, where ϵ is a small value to accommodate the potential measurement errors. For example, Fig.3 (a) illustrates a raw point cloud obtained from a LiDAR sensor. By eliminating the ground points, only

the points above the ground surface remain, as depicted in Fig.3 (b).

Second, further reduction in perception data size can be achieved by removing the points associated with static objects. This is because moving objects such as vehicles and pedestrians, are more likely to be involved in collisions and accidents, and static objects like buildings and parked vehicles are less critical. To differentiate between different objects, including vehicles, pedestrians, and buildings, we employ the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [14]. By applying DBSCAN, the vehicles can identify location changes across consecutive frames of LiDAR data, considering that LiDAR sensors generate multiple frames per second (e.g., 10 frames per second). If an object's location undergoes significant changes, it is deemed as a moving object, and its corresponding perception data should be uploaded. Conversely, if an object's location remains relatively stable, it is considered a static object and its perception data can be discarded. One such example is shown in Fig. 3 (c).

By implementing these processing steps, we can achieve a substantial reduction in the size of the LiDAR data, often decreasing it from several megabytes (2-3 MB) to less than 20 kilobytes (KB). Additionally, further reduction in data size can be attained by leveraging compression techniques [15].

C. Building the Traffic Map

In order to build the traffic map, the edge server should collect all objects and position them to the right place in the real world locations. However, the point cloud uploaded by the vehicles are using the LiDAR coordinates, which only have the relative object locations with respect to the LiDAR sensors. Therefore, the edge server should transform it to the world coordinate to build the traffic map. To achieve this transformation, vehicles should upload their own locations and orientations, which can be obtained accurately by using SLAM techniques [16], [17]. Then, the LiDAR-to-world transformation matrix, T_{lw}, can be calculated according to the 3D projection rules [18]. Formally, let $[x, y, z]^T$ be the location of a point in the LiDAR coordinate, and let $[W_x, W_y, W_z]^T$ be the location of the corresponding point in the world coordinate. The transformation from the LiDAR coordinate to the world coordinate is represented as follow.

$$[W_x, W_y, W_z, 1]^T = T_{lw} \cdot [x, y, z, 1]^T$$

After transforming to world coordinate, the edge server merges all uploaded point clouds together to build the traffic map using techniques in [19], [20]. Based on the traffic map, our system will selectively disseminate the data to different vehicles separately, and the details of how to display the notification are out of the scope of this paper.

D. Object Tracking and Trajectory Prediction

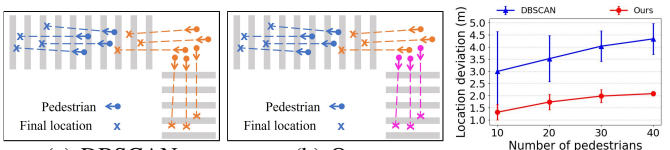
With the traffic map, the edge server can track the movements of vehicles and pedestrians, and then detect potential collisions. More specifically, the edge server employs trajectory prediction techniques to predict the paths of vehicles and pedestrians. By analyzing these predicted trajectories, it can proactively identify potential accidents that may occur when two trajectories intersect in the near future. While the majority of early research on trajectory prediction utilized Markov model [21]–[23], deep neural networks have been increasingly utilized for the prediction task in recent years [24]–[26]. However, predicting the trajectories of all objects requires significant computational resources, and it is infeasible to run in real time when the number of vehicles and pedestrians increases. To address this issue, we propose the following rules to reduce the number of objects whose trajectories need to be tracked and predicted.

Rule 1. To optimize the computation resources and reduce the complexity of trajectory prediction, we can exploit the fact that vehicles in the same lane tend to follow the leading vehicle, and the behaviors of following vehicles can be described by car-following models [27], [28]. Thus, we propose to track the leading vehicle in each lane that approaches the intersection and only predict the trajectories of the leading vehicle [26]. Note that the lanes can be identified based on the high-definition map at the edge server, which represent road environment [29], [30], and hence the first vehicle in each lane is the leading vehicle. Then, based on the predicted trajectory of the leading vehicle, the movements of the following vehicles can be estimated by leveraging car-following models. This approach can significantly reduce the number of vehicles whose trajectories need to be predicted, while still enabling the detection of potential collisions.

Rule 2. At intersections, vehicles may take various actions after passing the crosswalk, such as turning or going straight, leading to risky scenarios, like unprotected left turns [31]. Therefore, it is crucial to track and predict the movements of vehicles after they pass the crosswalk. To achieve this, a red boundary is defined along the crosswalk, as shown in Fig 5 (a). All moving vehicles within this boundary are tracked, and their trajectories are predicted. This allows the edge server to detect potential accidents in complex traffic scenarios.

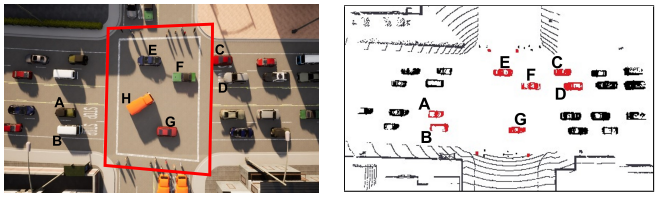
Rule 3. To reduce the computational cost of trajectory prediction for pedestrians, we propose a crowd-based clustering method. In each frame, instead of predicting trajectories for each individual pedestrian, we cluster them into crowds based on their proximity and moving directions. Each cluster has a representative. The edge server only needs to track and predict the trajectory of the representative, while other pedestrians are expected to be close to their representative. This method is suitable for predicting the trajectories of large crowds in a complex urban environment, where individual pedestrian trajectories might be difficult to model accurately. By clustering pedestrians into crowds, we can reduce the number of trajectories that need to be predicted and hence reduce the computational cost.

Despite the existence of numerous cluster algorithms, directly applying them in traffic scenarios poses a challenge. Some centroid-based algorithms, such as K-Means, require



(a) DBSCAN (b) Ours (c)

Fig. 4: (a) Clustering pedestrians into two clusters represented by different colors. (b) Clustering into three clusters. (c) Deviations of pedestrians' final locations.



(a) (b)

Fig. 5: (a) Vehicles within the red boundary should be tracked. (b) Objects in red points are tracked.

prior knowledge of the number of clusters. However, in dynamic traffic situations, it is impossible to determine the number of clusters beforehand due to the continuously changing nature of traffic. On the other hand, density-based algorithms like DBSCAN can automatically determine the number of clusters, but they may encounter issues with excessively large cluster sizes. In pedestrian clustering scenarios, large clusters can result in significant deviations in both location and orientation. Moreover, the implementation of DBSCAN makes it difficult to strike a balance between the importance of location and orientation metrics. When using DBSCAN to cluster pedestrians, the resulting clusters may not adequately capture the similarities in both location and orientation. To illustrate this point, consider Fig. 4 (a), which depicts pedestrians crossing crosswalks at an intersection. Each point represents a pedestrian, and the arrow indicates their direction of movement. The cross symbol indicates the final location of the pedestrian after moving along that direction for a period of time. The clustering result of DBSCAN is shown using different colors. In this illustration, the orange cluster produced by DBSCAN contains pedestrians who are close in location but are generally heading in two different orientations. Consequently, the final locations of these individuals (represented by orange crosses) are widely separated, leading to significant deviations in location.

To address the problem of DBSCAN and effectively group individuals with similar locations and orientations, we propose the following algorithm. Initially, we cluster pedestrians solely based on their locations. Then, for each cluster, we compute the standard deviations of the pedestrians' locations and orientations (moving directions). These deviations are compared to predefined thresholds for location deviation (β) and orientation deviation (γ). Any pedestrians with deviations exceeding the thresholds are removed from the cluster and assigned to a new one. This process is repeated until all clusters meet the deviation constraints. Fig. 4 (b) illustrates how our algorithm works, with different colors representing distinct clusters. In the figure, the pedestrians on the right exhibit varying orientations, resulting in significant orientation deviations. Consequently, they are divided into two separate clusters. It is evident that our algorithm successfully partitions the pedestrians into three clusters, with individuals within each cluster sharing similar locations and moving in the same direction.

To evaluate the performance of the proposed cluster algorithm, we collect pedestrian trajectories at an intersection and apply algorithms to cluster the pedestrians. Then, we measure the location deviations of the pedestrians in the same cluster after they move for a period of time. We set the thresholds $\beta = 2$ and $\gamma = 5$, and compared our cluster algorithm with DBSCAN. As shown in Fig. 4 (c), as the number of pedestrians increases, DBSCAN may result in larger cluster sizes and, consequently, larger location deviations. In contrast, our algorithm consistently outperforms DBSCAN across all tested scenarios, demonstrating superior performance in terms of reduced location deviations.

To demonstrate the scalability of our proposed method, we apply the three rules in the scenario depicted in Fig. 5 (a), while Fig. 5 (b) shows the point cloud of tracked vehicles and pedestrians (represented by red points). Firstly, applying *Rule 1*, we track the leading vehicles in each lane, which in this case are vehicles A, B, C, and D. Subsequently, according to *Rule 2*, we track vehicles that have passed the crosswalk (within the red boundary), such as vehicles E, F, and G. Their trajectories are then predicted. Additionally, utilizing our proposed cluster algorithm, we can cluster the pedestrians at the intersection into four groups. Following *Rule 3*, the edge server only predicts the trajectories for the representatives within each cluster. In the figure, only the perception data of four pedestrians are shown in red. Note that Fig. 5 (a) only displays a portion of the traffic, while there are ten additional following vehicles not shown in the figure. By implementing the proposed rules, our system tracks and predicts the trajectories of only seven vehicles and four pedestrians, rather than processing data for all 30 vehicles and 20 pedestrians, which is a significant reduction in computation.

After predicting the trajectories of the tracked vehicles and pedestrians, the edge server detects potential collisions and evaluates the relevance of the objects involved in these collisions. The approach used to estimate relevance will be discussed in the next section.

III. Relevance-Aware Perception Dissemination

In this section, we propose an efficient approach to estimate the relevance of the perception data. Based on the relevance, we formulate the relevance-aware perception dissemination problem and propose a greedy algorithm to solve it.

A. Relevance Estimation

To determine which perception data should be distributed to each vehicle, the edge server needs to assess the relevance

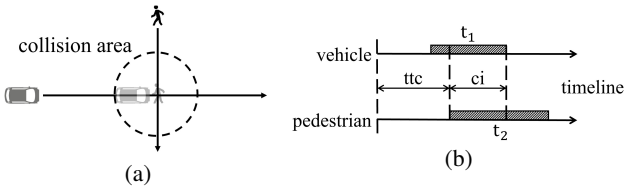


Fig. 6: (a) Collision area at the trajectory intersection. (b) Checking whether the objects are located in the collision area at the same time.

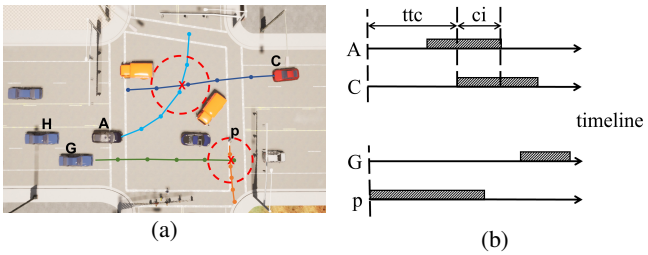


Fig. 7: (a) Collision areas (red circles) at the trajectory intersections. (b) A and C will collide, G and p will not collide, based on their passing time calculations.

of the data. In cases where objects (such as vehicles or pedestrians) are directly observable by the LiDAR sensors on the vehicles, it is unnecessary to disseminate the perception data related to those objects. Consequently, the relevance score for such data should be 0. Note that the server can infer an object's visibility to a vehicle by checking whether the uploaded perception data includes information about that object or not.

However, for objects that are obstructed by occlusions, their relevance must be estimated, and the dissemination decision will be based on this relevance assessment. One way to calculate the relevance is by analyzing potential collisions detected through trajectories. Additionally, as discussed before, not all trajectories of vehicles and pedestrians are explicitly predicted due to scalability issues. Thus, we introduce two different strategies to estimate relevance.

1) Relevance Estimation based on Trajectory Prediction: In Section II-D, we discussed several rules for tracking objects. By predicting their trajectories, it becomes possible to directly estimate their relevance. When the predicted trajectories of different objects intersect, it indicates a potential collision, making their perception data relevant. The relevance value in this case corresponds to the probability of potential collisions.

In recent trajectory prediction models, bivariate Gaussian distributions are commonly used to describe the uncertainty of the predicted locations [24]–[26], hence a natural idea to measure relevance is to calculate the joint probability of the distributions at the point of trajectory intersection. However, this method might underestimate the probability since it takes objects as points and ignores the size of the objects. This can lead to potential collisions occurring before their trajectories intersect. To address this limitation, some research proposes to treat the vehicles as shapes, such as circles and ellipses [32]–[34]. Then, the potential accidents can be quantified by calculating the overlap of shapes and cumulative probability within the overlap. However, this approach can introduce complexity in computing overlaps, especially when multiple vehicles are in motion, leading to heavy computational overhead.

In order to reduce the computational burden associated with calculating shape overlaps, we propose an alternative approach for estimating relevance. Instead of examining the exact overlap between shapes, we determine the likelihood of collision by checking if objects are simultaneously present in specific areas. To achieve this, we define a *collision area* as a circular region around the intersection of object trajectories, as depicted in Fig. 6 (a). The radius of the circle is set to the maximum length of the respective objects, ensuring that a collision will occur if both objects are within the collision area simultaneously. Next, we calculate the time intervals, denoted as t_1 and t_2 in Fig. 6 (b), during which each object passes through the collision area. By assessing the overlap in these intervals, we can determine the potential for collision. This overlap length is defined as the *collision interval* (ci). A longer collision interval indicates a higher probability of collision and a potentially more severe accident. To quantify the relevance of ci, we utilize the intersection over union of the passing times, represented by the ratio $R^{ci} = \frac{ci}{t_1 \cup t_2}$.

We also incorporate the concept of *time-to-collision* (ttc) to describe the potential occurrence of a collision. This metric is commonly used in risk assessment [32], [35]. Since a collision could happen at any moment within the collision interval, we focus on the earliest possibility. Then, ttc is defined as the duration leading up to the start of the collision interval, and its relevance is defined as $R^{ttc} = 1 - \frac{ttc}{T}$, where T represents the maximum time that the prediction algorithm can forecast. In cases where no collision interval exists, we have $ttc = T$, resulting in $R^{ttc} = 0$. By combining R^{ci} and R^{ttc}, we obtain the overall relevance of the perception data, represented as $R = \frac{(R^{ci}+R^{ttc})}{2}$. This approach offers an efficient and accurate method for estimating the relevance of perception data, enabling the prioritization of such data in subsequent dissemination tasks.

For instance, in Fig. 7 (a), the predicted trajectories of vehicles A and C intersect, forming a collision area indicated by the red circle. The server calculates the passing times based on the speeds of A and C, as shown in the timeline of Fig. 7 (b). The passing times of A and C overlap, suggesting a potential collision within the collision area. Consequently, the relevance between A and C is calculated using the formulas for R^{ci} and R^{ttc}. On the other hand, for the pedestrian p and vehicle G, their trajectories intersect but the passing times through the collision area do not overlap, as depicted in Fig. 7 (b). Therefore, both R^{ci} and R^{ttc} are 0, indicating the perception data of p and G are irrelevant.

2) Relevance Estimation based on Car Following Model: For the vehicles that are filtered out by the rules in Section II-D, their trajectories are not predicted and hence their relevance can not be estimated directly. However, it is impor-

tant to note that the relevance of these vehicles should not be ignored since they can still pose potential threats to safety in emergency situations. For example, consider the vehicle H in Fig. 7 (a), which is following the leading vehicle A. Since A is relevant to C, A will receive the perception data of C and slow down to avoid a potential collision. Without being alerted in advance, the driver of H needs some reaction time to respond when A starts to brake, and a rear-end collision between H and A may occur during this time. Therefore, the relevance between the following vehicles and other objects should also be estimated to ensure safety in case of emergencies.

According to the *Rule 1*, vehicles that are filtered out are mostly followers which drive after the leaders. There are many research that study on car following models for forecasting the behaviors of followers based on the leading vehicles [28]. In our proposed approach, we leverage these models to determine the relevance of following vehicles, and we identify two models that are well-suited for avoiding collisions in the event of sudden deceleration by the leading vehicle. The first is the *Pipes' Rule* [36] that emphasizes the safety distance between vehicles. It suggests that the safety distance between the follower and the leader should be the length of a car, *i.e.*, $4 \sim 5$ meters, for every 10 mph at which the follower is traveling. This rule allows enough distance for the follower to decelerate before colliding with the leader. The second model is the *Gipps model* [37] which take the driver's reaction time into consideration. In this model, the time gap between the follower and the leader should be 1.5 times of the reaction time. Typically, the human reaction time is one second on average, so the time gap in this model should be 1.5 seconds which ensures that the drivers have enough time to understand the traffic situation and take appropriate actions.

We utilize these car following models on each following vehicles to determine their relevance. If a following vehicle does not meet the criteria of either the *Pipes' Rule* or the *Gipps model*, it is considered relevant as it may potentially collide with its leader. Since the follower will only collide with the leader when the leader receives a disseminated perception data and decelerate, the relevance of the follower should be related to the leader's relevance. To this end, we define the follower's relevance as $R_{\text{follower}} = \alpha R_{\text{leader}}$, where $\alpha \in (0, 1]$ is a predefined decay factor, and it can be tuned based on different traffic scenarios. In the scenarios that rear-end collisions happen frequently, a larger decay factor could help since it lets the system give more attentions to the followers. To simplify the evaluation, we keep the decay factor as $\alpha = 0.8$ in this paper. For the example in Fig. 7 (a), if H fails to satisfy the car following models, its relevance should be $R_{\text{H,C}} = \alpha R_{\text{A,C}}$. As a result, H will receive the perception data of C such that H's driver can foresee its leader A's deceleration, and hence H can avoid the rear-end collision with its leader.

B. Relevance-Aware Perception Dissemination

In our system, we disseminate the perception data, *i.e.*, LiDAR points, to vehicles since the LiDAR points provide abundant details about the 3D information which can be used for many general tasks, such as path planning, 3D object detection and segmentation. The data size of the perception data is proportional to the number of points that represent the corresponding objects, and the size grows when more vehicles are uploading their perception data, resulting in a dense point cloud. Then, the bandwidth might become a bottleneck for disseminating these perception data. Thus, our goal is to only disseminating the most relevant perception data to corresponding vehicles under some bandwidth constraints.

Algorithm 1 Perception Dissemination

Input: Perception data $\{o_1, o_2, \cdots, o_n\}$ and their data size $\{s_1, s_2, \cdots, s_n\}$, relevance $R_{i,j}$, bandwidth constraint B
Output: Disseminated perception data Z

```
Z ← ∅
Data size of disseminated perceptions b ← 0
while b < B do
    Find the perception data o_i and the vehicle j
      which maximizes R_{i,j}/s_i
    Z ← Z ∪ (o_i, j)
    b ← b + s_i
    R_{i,j} ← 0
end while
return Z
```

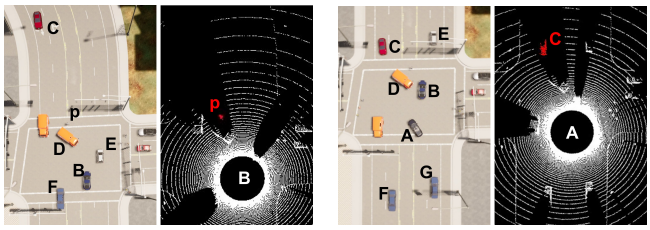
Definition 1 (Perception Dissemination Problem)**.** *Given the perception data of n objects $o_1, o_2, o_3, \cdots, o_n$ and their data size $s_1, s_2, s_3, \cdots, s_n$. Also given the relevance of these objects $R_{i,j}$ $(i, j = 1, 2, 3, \cdots, n)$. Our goal is to maximize the total relevance with the bandwidth constraint B, and the problem can be formulated as follow.*

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} R_{i,j} z_{i,j} \qquad \textit{s.t.} \sum_{i=1}^{n} \sum_{j=1}^{n} s_i z_{i,j} \leq B$$

where $z_{i,j} \in \{0, 1\}$ indicates whether the perception data o_i should be disseminated to vehicle (or pedestrian) j or not. When $z_{i,j} = 1$, the perception data should be disseminated; otherwise, there is no dissemination.

This problem can be converted to a binary knapsack problem, which is proved to be NP-hard. In the binary knapsack problem, there are n items and each item has a weight and a value. Given a knapsack with a weight limit, the problem is to select some items so that the total weight of selected items is less than or equal to the weight limit and the total value is maximized. Similarly, in the perception dissemination problem, disseminating the perception data o_i to vehicle j transmits the data of size s_i (weight) and has a relevance $R_{i,j}$ (value). With the bandwidth constraint B (weight limit), the perception dissemination problem is equivalent to maximizing the total relevance (value) of the disseminated perception data while the transmitted data size (weight) should not exceed the bandwidth constraint (weight limit).

To solve the problem, we propose a greedy algorithm as shown in Algo. 1. We define a relevance/size award $R_{i,j}/s_i$,



(a) Vehicle B's view. (b) Vehicle A's view

Fig. 8: The views of vehicles B and A.

which takes both relevance and the bandwidth consumption into consideration, and our algorithm schedules the dissemination iteratively based on this award to maximize the relevance while not exceeding the bandwidth constraint. Specifically, as shown in Algo. 1, our algorithm starts from an empty set Z, which is the perception data to be disseminated. In each iteration, the algorithm selects the relevant perception data o_i and the vehicle j which maximize the relevance/size award. Then, the algorithm adds the perception data o_i and the corresponding vehicle j to set Z and increases the total perception data size b by s_i. Our algorithm stops when the total amount of perception data b reaches the bandwidth constraint B, and the edge server disseminates all perception data in Z to the corresponding vehicles.

IV. Performance Evaluations

In this section, we first use a demo to show the effectiveness of our system and then evaluate its performance with extensive simulations.

A. Simulation Setup

To facilitate the collection of LiDAR data from multiple vehicles simultaneously and control the vehicles to avoid collisions upon receiving disseminated perception data, our evaluations are conducted using the open-source simulator CARLA [38], which is specifically designed for autonomous driving. CARLA enables multiple vehicles within the same scenario, allowing us to equip them with various sensors, like LiDAR, to perceive the surrounding environment accurately. Furthermore, CARLA provides a range of builtin high-definition maps that include diverse traffic scenarios. We setup two scenarios at an intersection where accidents are inevitable, and in each scenario we position multiple vehicles, equipped with 64-channel LiDAR sensors mounted on the roof. The LiDAR sensors have a maximum perception range of 50 meters and they scan the surrounding environment 10 times per second, generating 10 frames per second with over 1,000,000 points in each frame.

We emulate the computation capacity of the vehicles with NVIDIA Jetson TX2 [39] which has 8 GB memory and a NVIDIA Pascal GPU to process the LiDAR data. We deploy the edge server on a machine with an Intel Core i7 3.80GHz CPU and an NIVIDIA RTX 3080 GPU. To simulate the bandwidth constraints of wireless communications, we use the same maximum bandwidth as measured in [9].

B. Demo

Fig. 8 uses two traffic scenarios to demonstrate the effectiveness of our system. For each scenario, the left figure provides the camera view, and the right figure provides the LiDAR view.

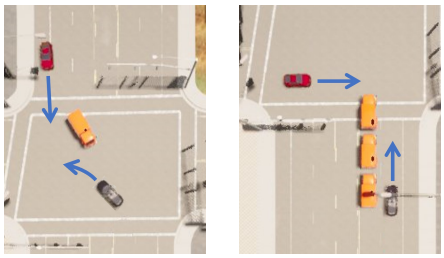
In Fig. 8 (a), vehicle B is driving straight forward, and the pedestrian p is crossing the road. As shown in the camera view (left), p is obscured by the truck D and is not visible to B's LiDAR. Consequently, B might collide with p. However, other vehicles such as E, can capture p and upload it to the edge server. Using this data, the edge server predicts the trajectories of B and p, and detects a potential collision. Then, the edge server estimates the relevance between B and p, and disseminates the perception data of p to vehicle B based on our perception dissemination algorithm. As shown in the LiDAR view (right) in Fig.8 (a), B will have p's perception data (represented by red points) and avoid the collision. In contrast, although C is not visible to B, its perception data is not disseminated. Because the trajectories of B and C will not intersect, their relevance is 0 and there is no need to disseminate C's perception data.

In the second scenario shown in Fig. 8 (b), vehicle A intends to turn left at the intersection while C goes straight. Due to the occlusion caused by D, both vehicles cannot perceive each other and an accident might happen. With our system, both A and C are detected by the edge server, and the potential collision can be detected based on their trajectories. With the relevance estimation, the edge server disseminates the perception data of C to A, represented by the red points in Fig. 8 (b).

C. Simulation Results

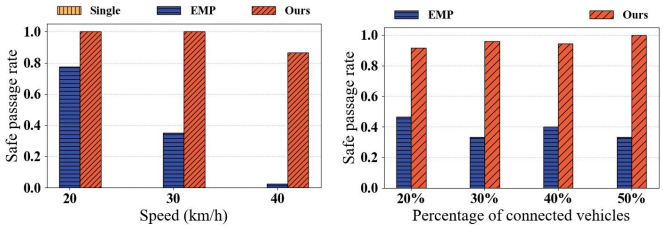
In order to comprehensively evaluate the performance of our proposed system, we conducted extensive simulations using various metrics. We focused on two scenarios: the *unprotected left turn* and the *red-light violation* at an intersection. Fig. 9 (a) illustrates the left turn scenario, where the black vehicle intends to make a left turn while its view is obstructed by the orange truck, rendering the red vehicle invisible to the black one. Consequently, the black vehicle may collide with the red vehicle at the intersection. In the red-light violation scenario depicted in Fig. 9 (b), the black vehicle approaches the intersection while the red vehicle runs red light. Due to the presence of orange trucks waiting at the intersection, obstructing the view of both the black and red vehicles, an accident may become inevitable.

For both scenarios, we simulated a busy urban traffic scenario by spawning 40 vehicles at the intersection. Based on statistical projections [40], the usage of advanced driver assistance systems is expected to reach 50% by 2030. Therefore, we vary the percentage of connected vehicles from 20% to 50% and evaluate the bandwidth consumption and performance accordingly. Considering the varied speeds of vehicles in urban traffic, we adjust the vehicle speed in both scenarios, ranging from 20 km/h to 40 km/h. For each scenario and setup, we performed five simulation runs and reported the average results to ensure reliability and accuracy of our evaluations.



(a) (b)

Fig. 9: (a) Black vehicle: unprotected left turn. (b) Red vehicle: red-light violation.



(a) (b)

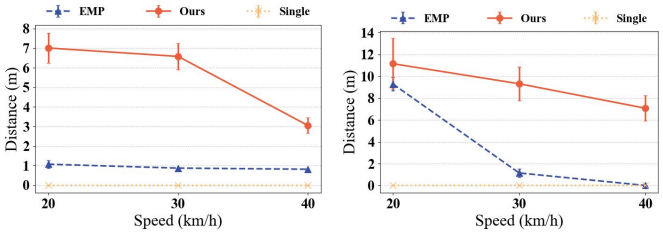
Fig. 10: Safe passage rate with different (a) driving speeds (*Single* are all 0%) (b) percentage of connected vehicles.

We conduct a comparative analysis of our method (*Ours*) against several baseline approaches.

- *Single*: Vehicles only rely on their own LiDAR sensors to perceive the surrounding environment while no data are shared;
- *EMP* [9]: The road is partitioned into non-overlapping regions using Voronoi diagrams, and each vehicle only uploads the point cloud data corresponding to its designated region.
- *Unlimited*: Vehicles directly upload the entire raw LiDAR point cloud and the edge server disseminates the entire traffic map to all connected vehicles without bandwidth constraints.

Although *AUTOCAST* [41] also shares perception data, it is through V2V communication. Moreover, it is designed for autonomous driving and assumes that the trajectories of all vehicles are known as prior. However, such assumption is not valid in real-world traffic scenarios, and thus is not compared here.

1) Safe Passage: To show that our proposed system can improve safety, we evaluate the traffic safety in this section with the metric of safe passage rate, which measures the percentage of vehicles passing the intersection without collisions. We use the default controller in CARLA to control the vehicles, and then implement a simple logic to simulate human drivers' reactions to possible collisions. To account for human drivers' reaction time, vehicles decelerate one second after receiving the disseminated perception data. It is important to note that our system disseminates relevant perception data to vehicles which would be involved in potential accidents, thus only vehicles with disseminated perception data decelerate while other vehicles proceed normally.



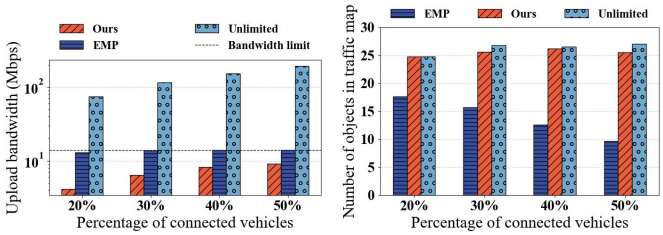
(a) Unprotected left turn (b) Red-light violation

Fig. 11: Minimum distance between the vehicles.

In Fig. 10 (a), we study how the driving speed affects the safe passage rate. The safe passage rates of *Single* are always 0%, indicating that accidents are inevitable if there is no sharing among vehicles. When the driving speed is below 40 km/h, our system achieves 100% safe passage rate and there is no collision in all cases, As the driving speed increases to 40 km/h, the safe passage rates of all methods drops. This is because the stopping distance increases as the driving speed increases. Therefore, even with the disseminated perception data, it is possible for the vehicles to collide when driving in high speed. Nevertheless, our system still achieves at least 85% of safe passage when vehicles are driving fast. In contrast, *EMP* performs much worse when the driving speed increases due to the following reasons. First, *EMP* uses the Round Robin strategy, and the relevant perception data might not be disseminated immediately due to bandwidth constraints. Then, perception dissemination can be delayed, leading to insufficient time for the vehicles to slow down. As the speed increases, the impact of the delay is more significant and the safe passage rate of *EMP* decreases dramatically. Second, as discussed before, *EMP* might lose some information during uploading due to the upload bandwidth constraints. As a result, some relevant objects might be lost and even the edge server is not aware of the potential accident.

Fig. 10 (b) shows the safe passage rate with varying percentages of connected vehicles. Because the *Single* case has 0% of connected vehicles, it is not depicted in the figure. A lower percentage of connected vehicles (*i.e.*, 20%) may leave some areas of the intersection uncovered, causing delayed detection of the relevant objects and a lower safe passage rate. As the number of connected vehicles increases, more perception data is shared with the edge server, leading to a more precise and robust traffic map. Then, potential collisions can be detected earlier. This results in an increased safe passage rate, as indicated by *Ours* method. However, as the percentage of connected vehicles increases, some relevant objects may not be disseminated due to bandwidth limits in *EMP*, leading to a decreased safe passage rate.

Fig. 11 compares the minimum distance between vehicles in different traffic scenarios. Since collisions are inevitable in the *Single* method, its minimum distance is always 0. In both scenarios, with the perception data disseminated by our system, the minimum distance between vehicles is much larger than that of *EMP*. The distance become shorter as the driving



(a) (b)

Fig. 12: (a) Bandwidth consumption of the LiDAR data uploading. (b) Number of objects in the uploaded LiDAR data.

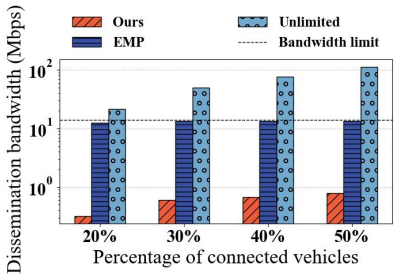
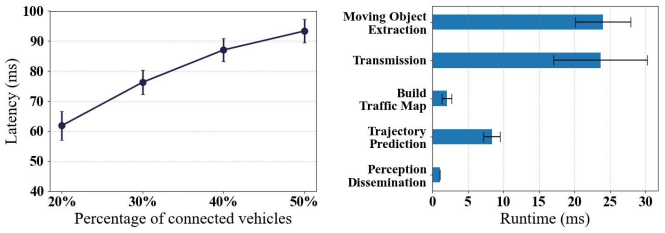


Fig. 13: Bandwidth consumption of perception dissemination.



(a) (b)

Fig. 14: (a) End-to-end latency. (b) Runtime of different modules.

speed increases, but our system still maintains distance longer enough for safe passage, *i.e.*, 3 meters in *unprotected left turn* and 7 meters in *red-light violation* with 40 km/h.

While improving the traffic safety, our system also reduces the bandwidth consumption, thereby enhancing the scalability, which is evaluated in the next sections.

2) Data Uploading: Given that LiDAR sensors generate 10 frames per second, vehicles upload their LiDAR data 10 times per second. In this section, we evaluate the bandwidth consumption when vehicles upload their perception data.

Fig. 12 (a) presents the average bandwidth consumption for different percentages of vehicles uploading their LiDAR data. By employing the *Moving Object Extraction* technique to remove unnecessary points from the LiDAR data, our method significantly reduces bandwidth consumption compared to the *Unlimited* baseline. Additionally, our approach requires less bandwidth than the *EMP* method since we further eliminate the perception data of static objects like buildings and parked vehicles. For example, in the *red-light violation* scenario depicted in Fig. 9 (b), three trucks are waiting to turn. The *EMP* method uploads the perception data of these trucks, even though they are stationary and irrelevant to other vehicles and pedestrians. In contrast, our system removes these static objects, resulting in greater bandwidth savings. Furthermore, Fig. 12 (a) demonstrates that the bandwidth consumption of *EMP* almost reaches the bandwidth constraint. Consequently, some perception data cannot be uploaded to the edge server, leading to information loss, such as missing objects.

Fig. 12 (b) shows the number of moving objects detected from the uploaded perception data. Due to the bandwidth constraint, *EMP* detects fewer objects. Moreover, as the percentage of connected vehicles increases, the size of the perception data grows, intensifying the impact of the bandwidth constraint. Consequently, the number of objects detected by *EMP* decreases, as shown in Fig. 12 (b). However, since our system has sufficient bandwidth to support perception data uploading, we can detect more objects. Remarkably, the results are comparable to the *Unlimited* baseline, which utilizes raw LiDAR data for object detection.

3) Perception Dissemination: After collecting the perception data from multiple vehicles, the data should be disseminated to appropriate vehicles to notify drivers about the potential accidents. We evaluate the bandwidth consumption for the dissemination from the edge server to vehicles. For comparison, we apply the Round Robin strategy in the *EMP* method, which means the perception data of all objects in the traffic map will be disseminated to all vehicles round by round. Besides, we use the *Unlimited* baseline to show how much bandwidth is needed to disseminate the whole traffic map to all vehicles in one round.

Fig. 13 shows the bandwidth consumption during dissemination as the percentage of connected vehicles increases. In both our system and the *Unlimited* baseline, the bandwidth requirement for dissemination increases. However, the increase in bandwidth consumption for *Unlimited* is exponential, making it impossible for wireless communication to support such high bandwidth requirements. The bandwidth consumption of *EMP* does not change significantly because it is constrained by the bandwidth limitation. In contrast, our system selectively disseminates relevant perception data to the appropriate vehicles, resulting in significantly lower bandwidth consumption.

4) Latency: As the LiDAR sensors on vehicles operate at a rate of 10 frames per second, we optimize the whole system and reduce the end-to-end latency, ensuring that the entire dissemination process can be completed within the time interval between two LiDAR frames. Specifically, the end-to-end latency encompasses the time from the moment LiDAR sensors generate the point cloud to when the relevant data is disseminated.

Fig.14 (a) shows the end-to-end latency of our system. As the percentage of connected vehicles increases, more data are uploaded, and longer processing delay is involved at the edge server for analysis. As a result, the end-to-end latency increases.

Fig.14 (b) shows a breakdown of the latency caused by different modules in our system when the percentage of connected vehicles is 20%. The *Moving Object Extraction*

module consumes the most time due to the limited computation capacity of the vehicles and the dense LiDAR point cloud. Moreover, the complexity of processing the point cloud varies as the surrounding environment of the vehicles changes, leading to a high variance in the runtime. After that, the reduced point cloud is uploaded through the wireless communication which takes about 24 ms. Then, the edge server takes about 3 ms to build the traffic map. Based on our proposed rules that only predict the trajectory of representative vehicles and pedestrians, the computation complexity of trajectory prediction is significantly reduced, resulting in a short overhead, *i.e.*, less than 10 ms. By leveraging our proposed greedy algorithm, the perception dissemination process takes only about 1 ms to make dissemination decisions.

V. Related Work

Cooperative Perception: Our work is related to the field of cooperative perception in vehicular networks [42]. In this paper, we focus on data sharing among vehicles, and existing studies [9], [41], [43], [44] have explored the transmission of raw perception data, such as camera images and LiDAR point clouds. For example, in *AVR* [45], Qiu *et al.* propose to share the raw 3D sensor data among vehicles through V2V communication. In *EMP* [9], LiDAR point clouds are uploaded to the edge server, and several techniques are proposed to reduce the data size and bandwidth consumption during uploading. *AUTOCAST* [41] shares the LiDAR data through V2V communication and improve the scalability; however, it is designed for autonomous driving and assumes that the trajectories of all vehicles are known as prior, which is impractical in real-world traffic scenarios nowadays. In [44], Wang *et al.* proposed an edge-assistant camera selection system to achieve cooperative perception in vehicular networks. By leveraging camera metadata, the system selectively chooses essential camera images, effectively reducing bandwidth, storage, and processing requirements at the edge server.

In recent years, advancements in vehicle technology have allowed for some computational tasks to be performed locally. Some researchers propose processing perception data, such as feature extraction, locally and sharing only the extracted features [46], [47]. For instance, Chen *et al.* [46] introduced a framework that fuses 3D features of perception data from multiple vehicles for object detection. Certain works even perform object detection locally, as demonstrated by *VIPS* [48]. VIPS applies 3D object detection models on vehicles to extract objects from LiDAR data, followed by reconstructing a 3D traffic map of the surroundings by aggregating information from roadside infrastructures. However, these approaches do not address the challenges of determining which perception data are relevant and to which vehicles the perception data should be disseminated.

Mobile Edge Computing: Leveraging edge computing techniques, heavy computations can be offloaded to the edge server [4], [49]–[53]. For example, in [54], computationally intensive object detection tasks are offloaded to edge servers, enhancing mobile AR applications with improved frame rates and detection accuracy. Similarly, in vehicular networks, edge servers play a crucial role in data processing and information sharing. For instance, *LiveMap* [7] utilizes an edge server to schedule vehicles for offloading camera images, minimizing latency and enabling real-time processing for constructing a dynamic traffic map. In [8], vehicles dynamically offload images to an edge server for collaborative localization and tracking. While we also leverage the capabilities of an edge server to build a comprehensive traffic map, our focus is on disseminating relevant perception data to corresponding vehicles.

VI. Conclusions

In this paper, we have proposed an edge-assisted relevance-aware perception dissemination system that collects perception data from multiple vehicles and selectively disseminates only the necessary data to appropriate vehicles. To reduce computational overhead, we have developed techniques for clustering vehicles and pedestrians based on their locations and orientations, enabling us to track representative entities instead of individual objects. Moreover, we have proposed an efficient approach to measure the relevance of perception data, allowing us to prioritize the dissemination of the most relevant information. By formulating and solving the relevance-aware perception dissemination problem, we aim to maximize the overall relevance of disseminated data while considering bandwidth constraints. To evaluate the effectiveness and efficiency of our proposed system, we have conducted comprehensive experimental studies. The results demonstrate that our system enhances the traffic safety while significantly reducing the overall bandwidth consumption.

VII. Acknowledgments

This work was supported in part by the National Science Foundation under grant number 2125208.

References

[1] S. Kumar, S. Gollakota, and D. Katabi, "A Cloud-Assisted Design for Autonomous Driving," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012.

[2] S.-W. Kim, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "The Impact of Cooperative Perception on Decision Making and Planning of Autonomous Vehicles," *IEEE Intelligent Transportation Systems Magazine*, 2015.

[3] S. Kassir and G. de Veciana, "Opportunistic Collaborative Estimation for Vehicular Systems," in *IEEE INFOCOM*, 2023.

[4] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge Computing for Autonomous Driving: Opportunities and Challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[5] F. Ahmad, H. Qiu, R. Eells, F. Bai, and R. Govindan, "CarMap: Fast 3D Feature Map Updates for Automobiles," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020.

[6] P. Zhou, T. Braud, A. Zavodovski, Z. Liu, X. Chen, P. Hui, and J. Kangasharju, "Edge-Facilitated Augmented Vision in Vehicle-to-Everything Networks," *IEEE Transactions on Vehicular Technology*, 2020.

[7] Q. Liu, T. Han, J. L. Xie, and B. Kim, "LiveMap: Real-Time Dynamic Map in Automotive Edge Computing," in *IEEE INFOCOM*, 2021.

[8] L. Liu and M. Gruteser, "EdgeSharing: Edge Assisted Real-time Localization and Object Sharing in Urban Streets," in *IEEE INFOCOM*, 2021.

[9] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao, "EMP: Edge-assisted Multi-vehicle Perception," in *ACM MobiCom*, 2021.

[10] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D Object Detection from Point Clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-Based 3D Single Stage Object Detector," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.

[15] "Draco 3D Data Compression." https://github.com/google/draco.

[16] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A Lidar-Inertial State Estimator for Robust and Efficient Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[17] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[18] D. Hearn and M. Baker, *Computer Graphics, C Version*. Pearson Education, 1997.

[19] W. Gao and R. Tedrake, "FilterReg: Robust and Efficient Probabilistic Point-Set Registration Using Gaussian Filter and Twist Parameterization," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for Fast and Accurate 3D Point Cloud Registration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[21] W. Gao and G. Cao, "Fine-Grained Mobility Characterization: Steady and Transient State Behaviors," *ACM Mobihoc*, 2010.

[22] A. J. Nicholson and B. D. Noble, "BreadCrumbs: Forecasting Mobile Connectivity," in *ACM MobiCom*, 2008.

[23] J.-K. Lee and J. C. Hou, "Modeling Steady-State and Transient Behaviors of User Mobility: Formulation, Analysis, and Application," in *ACM MobiHoc*, 2006.

[24] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] N. Deo and M. M. Trivedi, "Convolutional Social Pooling for Vehicle Trajectory Prediction," in *CVPR Workshops*, 2018.

[26] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "SpAGNN: Spatially-Aware Graph Neural Networks for Relational Behavior Forecasting from Sensor Data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[27] M. Brackstone and M. McDonald, "Car-following: a historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999.

[28] H. U. Ahmed, Y. Huang, and P. Lu, "A Review of Car-Following Models and Modeling Tools for Human and Autonomous-Ready Driving Behaviors in Micro-Simulation," *Smart Cities*, vol. 4, no. 1, pp. 314–335, 2021.

[29] R. Liu, J. Wang, and B. Zhang, "High Definition Map for Automated Driving: Overview and Analysis," *Journal of Navigation*, vol. 73, p. 324–341, 2020.

[30] G. Elghazaly, R. Frank, S. Harvey, and S. Safko, "High-Definition Maps: Comprehensive Survey, Challenges, and Future Perspectives," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 527–550, 2023.

[31] R. Najm, Wassim G.and Ranganathan, G. Srinivasan, J. D. Smith, S. Toma, E. Swanson, and A. Burgett, "Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications," 2013. Tech Report.

[32] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *IEEE International Conference on Intelligent Computer Communication and Processing*, 2009.

[33] M. Althoff, O. Stursberg, and M. Buss, "Model-Based Probabilistic Collision Detection in Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[34] A. Lawitzky, D. Althoff, C. F. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss, "Interactive scene prediction for automotive applications," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[35] A. Berthelot, A. Tamke, T. Dang, and G. Breuel, "A Novel Approach for the Probabilistic Computation of Time-To-Collision," in *IEEE Intelligent Vehicles Symposium*, 2012.

[36] L. Pipes, "An Operational Analysis of Traffic Dynamics," *Journal of Applied Physics*, vol. 24, pp. 274–281, 1953.

[37] P. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[38] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Annual Conference on Robot Learning*, 2017.

[39] "Jetson TX2 Module." https://developer.nvidia.com/embedded/jetson-tx2/.

[40] "Huge opportunity as only 10% of the 1 billion cars in use have ADAS features." https://www.canalys.com/newsroom/huge-opportunity-as-only-10-of-the-1-billion-cars-in-use-have-adas-features, 2021.

[41] H. Qiu, P.-H. Huang, N. Asavisanu, X. Liu, K. Psounis, and R. Govindan, "AutoCast: Scalable Infrastructure-Less Cooperative Perception for Distributed Collaborative Driving," in *ACM MobiSys*, 2022.

[42] C. Qiu, S. Yadav, A. Squicciarini, Q. Yang, S. Fu, J. Zhao, and C. Xu, "Distributed Data-Sharing Consensus in Cooperative Perception of Autonomous Vehicles," in *IEEE ICDCS*, 2022.

[43] Q. Chen, S. Tang, Q. Yang, and S. Fu, "Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds," in *IEEE ICDCS*, 2019.

[44] R. Wang and G. Cao, "Edge-Assisted Camera Selection in Vehicular Networks," in *IEEE INFOCOM*, 2024.

[45] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "AVR: Augmented Vehicular Reality," in *ACM MobiSys*, 2018.

[46] Q. Chen, "F-Cooper: Feature based Cooperative Perception for Autonomous Vehicle Edge Computing System Using 3D Point Clouds," *ACM/IEEE Symposium on Edge Computing*, 2019.

[47] H. Liu, P. Ren, S. Jain, M. Murad, M. Gruteser, and F. Bai, "FusionEye: Perception Sharing for Connected Vehicles and its Bandwidth-Accuracy Trade-offs," in *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2019.

[48] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu, and Z. Ouyang, "VIPS: Real-Time Perception Fusion for Infrastructure-Assisted Autonomous Driving," in *ACM MobiCom*, 2022.

[49] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing—A Key Technology Towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[50] Y. Liu, Y. Mao, S. Xiaojun, L. Zhenhua, and Y. Yuanyuan, "Energy-Aware Online Task Offloading and Resource Allocation for Mobile Edge Computing," in *IEEE ICDCS*, 2023.

[51] T. Tan and G. Cao, "Deep Learning Video Analytics Through Edge Computing and Neural Processing Units on Mobile Devices," *IEEE Transactions on Mobile Computing*, March 2023.

[52] T. Tan and G. Cao, "Deep Learning on Mobile Devices Through Neural Processing Units and Edge Computing," *IEEE INFOCOM*, 2022.

[53] X. Qin, Q. Xie, and B. Li, "Distributed Threshold-based Offloading for Heterogeneous Mobile Edge Computing," in *IEEE ICDCS*, 2023.

[54] L. Liu, H. Li, and M. Gruteser, "Edge Assisted Real-time Object Detection for Mobile Augmented Reality," in *ACM MobiCom*, 2019.