Robust Real-time Multi-vehicle Collaboration on Asynchronous Sensors

Qingzhao Zhang[†*], Xumiao Zhang[†*], Ruiyang Zhu[†*],
Fan Bai[‡], Mohammad Naserian[‡], Z. Morley Mao[†]
[†]University of Michigan [‡]General Motors

ABSTRACT

Cooperative perception significantly enhances the perception performance of connected autonomous vehicles. Instead of purely relying on local sensors with limited range, it enables multiple vehicles and roadside infrastructures to share sensor data to perceive the environment collaboratively. Through our study, we realize that the performance of cooperative perception systems is limited in real-world deployment due to (1) out-of-sync sensor data during data fusion and (2) inaccurate localization of occluded areas. To address these challenges, we develop *RAO*, an innovative, effective, and lightweight cooperative perception system that merges asynchronous sensor data from different vehicles through our novel designs of *motion-compensated occupancy flow prediction* and *on-demand data sharing*, improving both the accuracy and coverage of the perception system. Our extensive evaluation, including real-world and emulation-based experiments, demonstrates that *RAO* outperforms state-of-the-art solutions by more than 34% in perception coverage and by up to 14% in perception accuracy, especially when asynchronous sensor data is present. *RAO* consistently performs well across a wide variety of map topologies and driving scenarios. *RAO* incurs negligible additional latency (8.5 ms) and low data transmission overhead (10.9 KB per frame), making cooperative perception feasible.

CCS CONCEPTS

• **Networks → Cyber-physical networks**; *Network protocol design*; • **Applied computing →** *Transportation.*

* These authors contributed equally to this paper.

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3613271

KEYWORDS

Cooperative Perception, Autonomous Cars, Vehicular Networks, LiDAR

ACM Reference Format:
Qingzhao Zhang[†*], Xumiao Zhang[†*], Ruiyang Zhu[†*],, Fan Bai[‡], Mohammad Naserian[‡], Z. Morley Mao[†]. 2023. Robust Real-time Multi-vehicle Collaboration on Asynchronous Sensors. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3570361.3613271

1 INTRODUCTION

Connected and autonomous vehicles (CAVs) rely on a variety of advanced on-board sensors (*e.g.,* GNSS, camera, Radar, and LiDAR) to localize their ego pose in relation to local road elements (*e.g.,* lanes, traffic signs), as well as to detect other traffic participants. However, the effective range of sensors can be limited. For example, off-the-shelf LiDAR sensors have an effective range of about 80 meters [10], beyond which the resolution of sensor data (*i.e.,* point cloud) decreases drastically. To this end, collaboration among CAVs as well as roadside infrastructure can enhance the perception capability of individual CAVs by extending the perception range and covering occluded areas. Through shared sensor data, each vehicle is able to broaden its perception of the surrounding environments, make more accurate predictions, and thus plan its driving behaviors more optimally.

Cooperative perception [16, 17, 27, 30, 51, 58, 62, 63, 69] is gaining traction in the research community in recent years. One challenge is system scalability. The bandwidth usage of sharing raw sensor data is not sustainable, even with state-of-the-art wireless communication technologies. If naively sharing full frames of high-fidelity data, the estimated bandwidth usage exceeds 300 Mbps [69], which is far beyond the capacity of current vehicular networks (*e.g.,* C-V2X at tens of Mbps) [57, 64]. To tackle this problem, one could first partition the raw point cloud into regions and then only transmit the most relevant regions. We call such approach *Partition and Selective Sharing*. For instance, EMP [69] shares close-range points while AutoCast [51] shares the points of objects with low visibility and high relevance to planning.

Though successful in addressing scalability, this selective sharing approach faces two fundamental challenges: (1)

During data fusion, a vehicle's local sensor data and data from remote vehicles (or infrastructure) are asynchronous, or captured at different times, due to different sensor configurations on both sides and unpredictable wireless transmission delay. The time gap between two frames can be larger than 100 ms [2, 54], leading to inaccurate data merging and damaged performance of downstream perception tasks. (2) The indirect estimation of occluded areas can be inaccurate. Particularly, instead of utilizing a vehicle's direct and more accurate insight of its own occluded areas, both EMP and AutoCast obtain location estimates by either inquiring others or using simple heuristics. Without knowing the exact locations and shapes of obstacles blocking the views, such practices may fail to share critical occluded areas.

In this paper, we focus on improving cooperative perception by tackling the challenges caused by asynchronous sensor data and inaccurate occlusion localization. To achieve this, we develop *RAO*, a Real-time Asynchronous Occlusion-aware multi-vehicle cooperative perception system, in which the merged sensor data consistently and accurately cover the occluded areas (*e.g.,* blind spots of vehicles), without incurring additional processing latency. While we use LiDAR as an illustrative example to represent generic 3D geometric features in surrounding environments, *RAO* can also be applied to other sensors like stereo camera or Radar.

We develop event-driven, asynchronous protocols for LiDAR sensors, enabling four separate point cloud processing and sharing tasks to be conducted in parallel without introducing additional delay: (1) When a vehicle gets a local LiDAR image, it generates and shares an occupancy map, which is a 2D map that labels free areas and occupied areas on the road. (2) Simultaneously, in each LiDAR scanning cycle, each vehicle sends data requests to ask remote vehicles or infrastructure for LiDAR data. The data requests are optimized to maximize the quality of the merged point cloud. (3) Upon receiving the data requests, the data is prepared and sent back to the corresponding vehicles. (4) Once a vehicle receives a local LiDAR sensor image, it merges the image with requested sensor data and then conducts downstream perception tasks, such as object detection and tracking.

To address the challenge of merging asynchronous sensor data, we design a novel motion-compensated occupancy flow prediction method, suitable for challenging automotive scenarios featured with high vehicle mobility as well as asynchronous and noisy data. Our prediction method provides sufficient resilience against asynchronous data in multi-vehicle perception collaboration by leveraging motion information calculated from previous frames. Moreover, to tackle the challenge caused by incomplete occlusion estimation, we take an ego-centric approach that allows a vehicle to determine its own occluded areas because the vehicle has direct and more accurate knowledge of these areas. Compared to prior art that relies on remote vehicles' indirect estimation, our solution enables the merged sensor data to cover more occluded areas with improved perception accuracy.

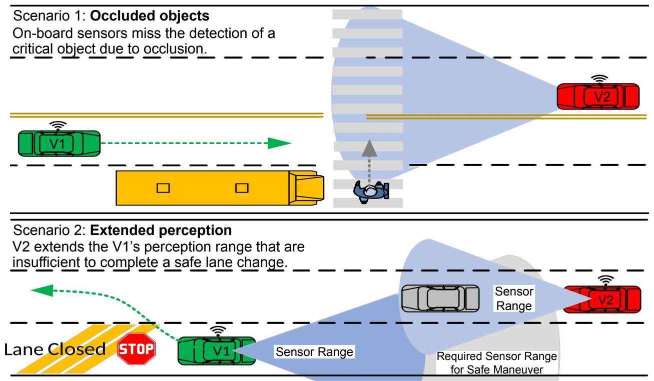
Scenario 1: Occluded objects
On-board sensors miss the detection of a critical object due to occlusion.
V1
V2
Scenario 2: Extended perception
V2 extends the V1's perception range that are insufficient to complete a safe lane change.
Lane Closed
STOP
Sensor Range
Sensor Range
Required Sensor Range for Safe Maneuver

Figure 1: Two scenarios where cooperative perception helps.

To evaluate *RAO*, we implement a prototype and run the system in a realistic setup where vehicles transmit and process LiDAR data asynchronously. Utilizing three datasets collected from both real-world and photo-realistic synthesized driving scenarios, we show that *RAO* yields a 34% and 14% improvement over state-of-the-art solutions in perception coverage and accuracy, respectively. *RAO* is lightweight; it only introduces an extra latency of 8.5ms to the perception process on the consumer side and generates a data volume of 10.9 KB per frame, on average. Extensive experiments demonstrate that *RAO* performs well across a rich variety of map topologies and driving scenarios.

The key contributions of this work are as follows:

• We identify the challenges of synchronization and inaccurate occlusion localization in practical cooperative sensor sharing and analyze their impacts on cooperative perception.
• We design and implement *RAO*, a real-time multi-vehicle cooperative perception framework, with robustness against asynchronous data and efficient data sharing scheduling.
• We propose innovative designs of occupancy flow prediction and on-demand data sharing that can efficiently synchronize LiDAR point clouds captured at different timestamps, enabling a seamless multi-vehicle data alignment and fusion.
• Through systematic experiments on two large-scale datasets and one field test, we demonstrate that *RAO* brings significant performance improvements to perception accuracy and coverage with minor system overhead.

2 BACKGROUND

Cooperative perception. The emerging cooperative perception aims to enhance CAVs' awareness of the surrounding environment. A vehicle's onboard sensors can miss the detection of critical objects moving into its path, due to occlusion caused by other obstacles on the road. The occluded object may be detected too late to avoid a collision. Figure 1 shows such a scenario: with sensor data shared from Vehicle 2 (V2), the perception of Vehicle 1 (V1) can cover the crossing pedestrian who is originally occluded by the bus. In addition, the sensors have a predefined perception range due to technological limitations. There are corner cases where the effective perception range is insufficient for conducting safe maneuvers. In Scenario 2 of Figure 1, V1 intends to change to an adjacent lane with high-speed traffic. Factors such as lane-change time and acceleration time dictate the required perception range behind V1 to complete the lane-change maneuver safely [47]. By sharing sensor data, approaching vehicles like V2 can benefit V1 with extended perception.

Existing cooperative perception schemes can be grouped in several ways. (1) Data sharing stage: Early-fusion sharing schemes [14, 16, 32, 51, 69] share the original raw sensor data, which typically has a universal format, but demands relatively high bandwidth for transmission; feature-level sharing schemes [15, 17, 58, 62, 67] send intermediate features of perception, offering a balance between network efficiency and perception accuracy; object-level sharing schemes [40, 54, 55] directly share lightweight perception results such as object bounding boxes. (2) Network communication topology: Vehicle-to-vehicle (V2V) sharing schemes [16, 51, 63] do not have a centralized party and shared messages are exchanged directly among vehicles; vehicle-to-infrastructure (V2I) sharing schemes [11, 69] utilize infrastructure resources, such as edge nodes, to aggregate vehicle data and distribute perception results back to vehicles. (3) Sensor type: Cooperative perception can be built on various types of sensors [50] while existing works mainly focus on LiDAR.

Vehicular wireless networks. Vehicular networks have been developed to facilitate collaboration among vehicles and roadside infrastructure for various tasks [40, 50, 52]. Commercial products using DSRC [28] can achieve up to 27 Mbps. Some modern vehicles are also equipped with cellular interfaces that support direct modes with higher bandwidths of tens of Mbps [22, 43, 57]. With the emergence of 5G, the vehicular network bandwidth is growing even higher [1, 46]. Nevertheless, sharing data through vehicular wireless networks is challenging due to factors such as bulky data size and bandwidth fluctuation. Previous studies [51, 69] have explored ways to reduce data transmission overhead and adapt to bandwidth changes when sharing sensor data via V2V/V2I wireless networks. A common design used by them is that vehicles transmit only partial sensor data by identifying the most relevant information through mathematical heuristics.

3 MOTIVATION

In this work, we focus on early-fusion cooperative perception [16, 51, 58, 63], where vehicles share LiDAR data via a wireless network and process data locally. Operating on raw

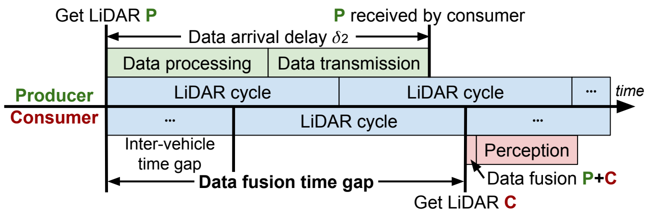

Figure 2: Synchronization problem in protocol diagram of cooperative perception.

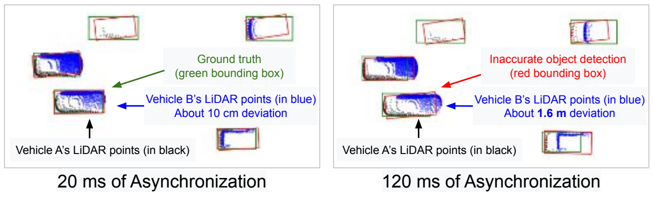

Figure 3: Merged point clouds from two vehicles (A and B) at various levels of synchronization (Bird-eye view).

sensor data enables flexible use of merged data. Efficient data sharing is made possible with mature network infrastructure and proper data partitioning. However, existing system proposals have severe limitations, including inaccurate data fusion due to the asynchronous nature of multi-vehicle data sharing and insufficient shared data due to incomplete occlusion detection. We discuss these issues in detail in §3.1 and §3.2, and address them by achieving the design goals itemized in §3.3.

Terms. We refer to a vehicle that receives sensor data as a *consumer*. Others, such as vehicles and roadside infrastructure, that share sensor data with consumers are *producers*. Note that a vehicle can play both roles simultaneously.

3.1 Asynchronous Sensor Data

Existing early-fusion cooperative perception proposals merge asynchronous sensor data, but few address the resulting inaccuracy problems. To this end, we break down the synchronization problem and analyze the temporal order of events in cooperative perception depicted in Figure 2.

Inter-vehicle time gap. Vehicular LiDAR sensors producing data periodically (the cycle is 100 ms for most commercial LiDAR sensors [10]). However, the capture time of LiDAR images on different LiDARs are not synchronized, leading to inevitable gaps between timestamps of LiDAR images from every two vehicles. For example, in Figure 2, the producer gets a LiDAR image from its local sensor at timestamp 0 ms, while the consumer's next image is generated between 0 ms and 100 ms. For each pair of a producer and a consumer, we define the *inter-vehicle time gap* as the time difference between the generation of the producer's LiDAR image and the generation of the consumer's next LiDAR image. Note that this time gap is primarily attributed to the configuration of LiDAR devices thus it cannot be mitigated by traditional clock synchronization methods such as NTP [44].

Data arrival delay. Even though a producer sends its LiDAR data, the data does not arrive at the consumer immediately. The data undergoes pre-processing (*e.g.,* partitioning, compression), transmission, and post-processing (*e.g.,* decompression) before the consumer can use it. As presented in previous works [16, 58, 69], the entire processing duration rarely falls below 100 ms, and thus cannot be ignored. We define the *data arrival delay* as the interval between a producer generating local LiDAR data and a consumer receiving processed, ready-to-use data.

Due to the above factors, a consumer may not have the producers' latest LiDAR data when generating its own local data. However, to minimize the latency of downstream perception tasks, most existing cooperative perception solutions opt to immediately perform data fusion and perception upon receiving their local LiDAR data. In this way, since LiDAR data fusion is a simple process of merging 3D points, these systems introduce little additional latency to perception tasks compared with single-vehicle perception.

The question is how dated the producer's data can be in such systems. Formally, given a producer and a consumer, we denote the LiDAR cycle time (*e.g.,* 100 ms) by T, the *inter-vehicle time gap* by δ_1, and the *data arrival delay* by δ_2. By definition, δ_2 inevitably contributes to the data fusion's time gap. There is also a time gap between data arrival and the consumer generating its next LiDAR image, $(\delta_1 - \delta_2 \bmod T)$. Therefore, the total time gap of data fusion is:

$$\delta = \delta_2 + (\delta_1 - \delta_2 \bmod T), 0 \leq \delta_1 < T, \delta_2 > 0. \quad (1)$$

Ideally, the producer's data arrives at the moment of the consumer's data fusion. In this case, $\delta_1 - \delta_2 \bmod T$ is zero, and δ is equal to δ_2. In contrast, in the worst case, the producer's data arrives right after the data fusion. δ goes up to $\delta_2 + T$.

Such synchronization problems are not explicitly discussed in existing cooperative perception proposals. False tolerance mechanisms in conventional robotic perception may not be ideal for CAVs, primarily due to the intricate driving scenarios and the strict latency requirements. For instance, registration algorithms (*e.g.,* ICP) are used in multi-robot SLAM [20] to calibrate generated maps. This approach spatially aligns maps of stationary objects but cannot address asynchronous time frames. Alternatively, other approaches [24, 53] employ motion models (*e.g.,* kinematics models) to forecast the behavior of moving objects. However, obtaining such motion models can be challenging for remote unidentified vehicles on the road. For connected vehicle applications, AutoCast [51] reports that both its data processing and transmission are barely below 100 ms. As a result, when the inter-vehicle time gap is close to 100 ms, the worst-case data fusion time gap is almost 300 ms according to Equation 1, during which a 60 km/h vehicle can shift its position by 5 meters. As shown in Figure 3, objects can hardly be recognized in the merged point clouds without proper synchronization. V2VNet [58], a deep learning model for intermediate-fusion cooperative perception, uses neural networks to compensate for the time gaps. It states that synchronization on raw point clouds is nontrivial, and therefore is not implemented in its evaluation. VIPS [54] tolerates the asynchronization by aligning bounding boxes from different CAVs, which is effective only in object-level sharing schemes. Autonomous vehicles [2, 3] commonly use high-definition (HD) maps for accurate positioning but maps alone cannot address the asynchronization problem as the errors come from inconsistent timestamps instead of inaccurate localization. To the best of our knowledge, there is no existing work addressing the early-fusion synchronization challenge.

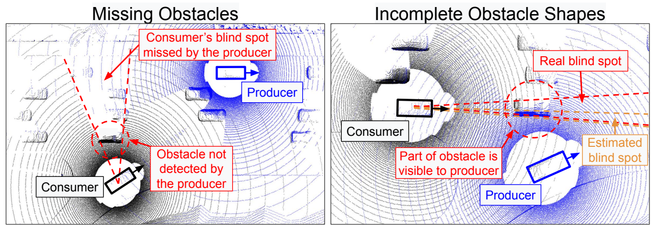
Missing Obstacles
Consumer's blind spot missed by the producer
Producer
Obstacle not detected by the producer
Consumer
Incomplete Obstacle Shapes
Real blind spot
Consumer
Part of obstacle is visible to producer
Estimated blind spot
Producer

Figure 4: Inaccurate blind spot estimation (the consumer's/producer's points and heading in black/blue).

3.2 Incomplete Occlusion Estimation

To reduce transmission latency, early-fusion cooperative perception schemes usually shrink the size of shared LiDAR data. Besides compression and sampling, previous works [50, 51] propose sharing only critical regions, *i.e.,* blind spots[1] of others. This is based on the insight that improving sensor data quality in well-observed regions does not greatly benefit downstream tasks. For instance, in EMP, vehicles share LiDAR data in close areas, based on Voronoi Diagram [12]. In AutoCast, producers estimate the blind spots of consumers based on the consumers' views. Specifically, producers in AutoCast acknowledge the location of consumers, localize on-road objects, and finally perform simple ray casting to "guess" consumers' blind spots occluded by these objects. Thanks to the blind spot estimation, AutoCast can cover more areas after data fusion than EMP. However, the estimated occluded areas can significantly differ from the ground truth. We analyze the sources of inaccuracies as follows.

First, each producer's perception range is limited and thus may not observe all obstacles blocking consumers' view. Without knowing the specific obstacle location, the provider can hardly identify the area behind that obstacle; in contrast, the consumer has accurate knowledge of its own occluded areas. As shown in Figure 4, one occluded area of the consumer, over 50 m^2, is missed by the producer because the key obstacle causing the occlusion is unknown to the producer.

[1]We define blind spots as areas that cannot be directly perceived by a vehicle itself, caused by occlusion or long distance. We also use the terms *blind spots* and *occlusion* interchangeably.

Second, producers cannot recognize the complete shape of obstacles, and thus the ray-casting method is sometimes inaccurate in identifying occluded areas. In Figure 4, the producer observes a key obstacle that may block the consumer's view but can only see one side of the obstacle without knowing the depth. As a result, when the producer uses the partial obstacle for ray casting, it implicitly assumes that the consumer can "see-through" the rear part, causing the estimated occluded areas to be much narrower than the ground truth.

Due to these inaccuracies, producer-centric occlusion estimation often misses actual occluded areas of consumers, degrading the benefits of data sharing.

3.3 Design Goals

In building *RAO*, we aim to solve the above problems while achieving real-time robust cooperative perception.

• **Robustness against asynchronous data.** Under realistic driving scenarios where vehicles share asynchronous sensor data, *RAO* should produce accurate perception results.

• **High occlusion coverage.** A large proportion of occluded areas should be filled with sensor data from remote vehicles.

• **Real-time protocol.** The execution of *RAO*'s protocol should meet the real-time requirements of real-world vehicle perception systems. To be specific, each perception cycle should be finished within 100 ms [39].

• **Negligible additional latency for perception.** Compared to single-vehicle perception, *RAO* should not cause any observable delay for vehicles to get perception results.

4 SYSTEM DESIGN

We describe how *RAO* satisfies each design goal outlined in §3.3. After shedding light on how to overcome the limitations of prior work in §4.1, we present key components and the general workflow of *RAO* in §4.2. We then elaborate the detailed design of three critical components: occupancy map generation, occupancy flow prediction, and data scheduling, in §4.3, §4.4, and §4.5, respectively.

4.1 System Overview

To tackle the inaccuracy problem caused by asynchronous data (§3.1) and incomplete blind spots (§3.2), we propose two novel solutions, *occupancy flow prediction* and *on-demand data scheduling,* in our design of *RAO*.

Occupancy flow prediction. Although sensor data items arriving at consumers are inherently asynchronous, one can still manage to synchronize the received point clouds from different producers using an intelligent prediction mechanism. While there is a rich body of literature on scene flow prediction that predicts the subsequent frame of a LiDAR image based on previous frames [18, 23, 34, 37], these algorithms are not suitable for our targeted applications. They fail to generate results sufficiently fast for real-time perception, with most requiring near or over 100 ms. Also, we aim

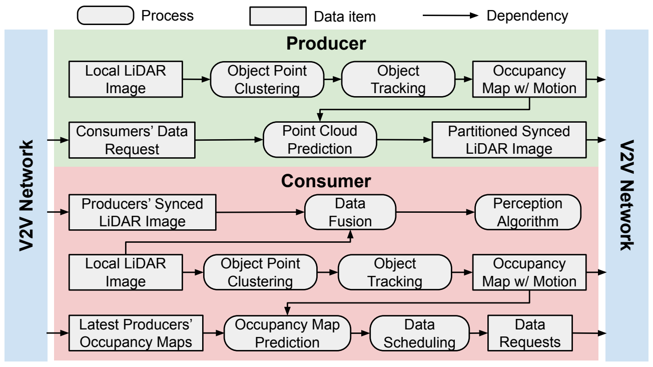

Figure 5: System Architecture of *RAO*.

to synchronize point clouds to any given future timestamp to compensate for the arbitrary time gaps between LiDAR images, while existing algorithms are frame-wise and can only synchronize point clouds at fixed intervals.

To achieve real-time prediction, we simplify the prediction problem by considering the uniqueness of CAV scenarios: the data asynchronization is mainly reflected on objects (*e.g.,* vehicles) moving on a surface (*e.g.,* ground). Therefore, unlike conventional approaches that focus on independent LiDAR points, we propose an efficient prediction algorithm that only tracks non-ground clusters of points. Using our mechanism, upon receiving a local LiDAR image in each frame, the vehicle clusters the LiDAR points to localize on-road objects and estimates the motion of clustered objects by comparing the corresponding clusters across consecutive frames. In this way, whenever synchronization is needed, even if the received data frame is missing or delayed, the consumer can leverage prediction to compensate for arbitrary time gaps.

On-demand data scheduling. To detect blind spots, we argue that consumers themselves have the most accurate and reliable information because they can readily recognize areas with few or even no LiDAR points and mark these areas as occluded areas. Inspired by this insight, we advocate that consumers should proactively request data on specific areas from producers, which is a departure from current mainstream approaches such as EMP [69] and AutoCast [51] where producers have to "guess" consumers' blind spots. The advantages of this consumer-centric approach are twofold: (1) It improves the accuracy of blind spot estimation since the consumer knows the best; and (2) proactive data requests by consumers prevent duplicate data shared by multiple producers, thus eliminating unnecessary bandwidth usage.

4.2 System Components

In *RAO*, the producers and consumers are connected by vehicular wireless networks that allow for the exchanges of sensor data and control messages. *RAO*'s protocol is event-driven, meaning that protocol tasks are executed by producers and consumers asynchronously. As shown in Figure 5, the execution of protocol tasks is triggered by certain events, either a new data frame being generated or a prior process being completed. The workflow of *RAO* can be divided into four separate tasks: occupancy map generation, data request generation, data response generation, and data fusion.

Occupancy map generation is responsible for producing occupancy maps, in which on-road objects, free-to-drive areas, and invisible areas are labeled in the 2D space. For each vehicle, after its LiDAR sensor generates a local LiDAR image, this component immediately starts to generate an occupancy map and broadcasts the map once finished. The occupancy map generation process involves several steps: (1) *RAO* runs LiDAR segmentation to separate LiDAR points into ground points and object points. (2) Then our clustering algorithm assigns the object points into separate groups of objects, and the object tracking algorithm applies correlation on clusters across consecutive frames. (3) To estimate the object motions, which are later used for occupancy flow prediction, *RAO* leverages 3D point cloud registration to obtain accurate transformation of each cluster over two consecutive frames, from which motion parameters (*e.g.,* velocity) are extracted. (4) Finally, the occupancy maps along with their corresponding tracks and motion parameters, are bundled together and broadcast over the V2V network.

Data request generation is executed on consumers to determine what data requests should be sent to producers. Once the consumer generates its local occupancy map, this process is triggered. An essential part of this process is *data scheduling*, which assigns needed LiDAR data that covers the identified blind spot areas to available producers. To achieve high-quality fusion of LiDAR data, the producers' LiDAR data should have a reasonably high resolution (*i.e.,* density of points). To achieve this goal, a consumer first (1) synchronizes different occupancy maps on both the consumer and producer sides to the timestamp of its next LiDAR cycle using our *occupancy flow prediction* method; (2) Then the consumer solves an optimization problem to assign each needed blind spot area to the producer who can provide the best quality of sensor data on that area. (3) The consumer sends out a sequence of requests to corresponding producers.

Data response generation is executed on producers and is triggered by receiving consumers' data requests. For each received data request, the producer fetches its latest locally generated LiDAR image and partitions it into the interested areas per request. Then the producer synchronizes the partitioned image into the timestamp requested by the consumer and sends it back to the consumer. This synchronization step, again, leverages our *occupancy flow prediction* method.

Data fusion is the final process that merges available LiDAR images from different vehicles, before feeding the generated outcome to downstream perception modules. Once a consumer vehicle gets its local LiDAR image, it immediately collects LiDAR data shared by other producers, transforms all LiDAR images to the same coordinate system, and then combines them. No additional data processing is needed since the producers have already executed partitioning and synchronization tasks, delivering the transformed data. Thanks to its simplicity, *RAO* introduce no observable latency onto downstream modules.

Note that *RAO* is a framework built on the fundamental CAV architecture. We assume the CAVs can actively discover nearby CAVs and establish bi-directional V2V connections via C-V2X or DSRC. The CAVs also communicate their poses and locations (GPS/IMU), which are required for transforming LiDAR data to the same coordination system.

In the remainder of this section, we elaborate on our designs of three major mechanistic components in *RAO*.

4.3 Motion-aware Occupancy Map

This component generates an occupancy map, which not only labels the location of on-road objects but also tracks their mobility trajectory and their motion parameters. Existing deep learning-based perception solutions are not suitable for generating occupancy maps because they require significant computing resources and serve different goals. In *RAO*, the occupancy map acts as metadata frequently shared among CAVs, in order to guide efficient scheduling of overall data sharing. To this end, occupancy maps need to be lightweight in computation and communication overhead. In contrast, perception algorithms based on deep learning, such as 3D object detection and tracking, are performed on the merged sensor data after the data fusion. Occupancy maps indirectly benefit those expensive perception algorithms by improving the quality of data fusion. We introduce the generation of occupancy maps as follows.

LiDAR point segmentation. We first remove less useful background points that fall outside of the road regions, leveraging HD maps provided by autonomous driving systems [2, 3]. Then, we use existing ground detection algorithms [21, 48] to detect the ground plane and remove LiDAR points on the ground. By clustering the remaining points, we can identify all the non-ground objects on the road, with each cluster representing a unique on-road object. This method has been proven to be effective in prior art [19, 60, 68].

Area segmentation. After identifying on-road objects, we generate a fine-grained representation of 2D space occupancy, which classifies the 2D space into three categories: *occupied*, *free*, and *occluded*. (1) Occupied areas are generated by computing the convex hulls [13] of the object clusters. (2) Free areas represent the region containing only ground points. As demonstrated in Figure 6, we split the 2D space evenly into sectors whose vertex is the LiDAR location (the number of sectors is configurable for various granularity). In each sector, we measure the distance from the LiDAR location to the closest non-ground point and label the area

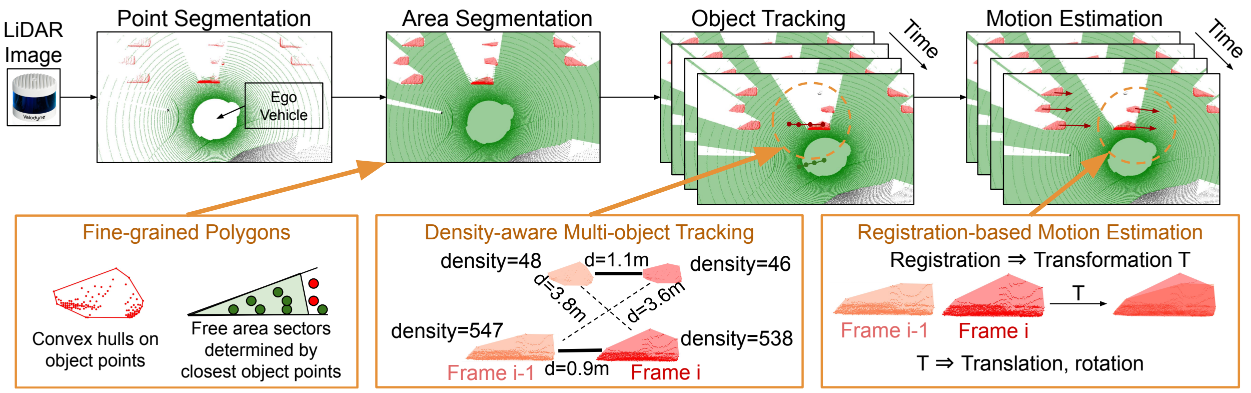

Figure 6: Demonstration of the generation of motion-aware occupancy maps.

within the distance as a free area. (3) Occluded areas are our key interests in cooperative perception. They constitute the areas that are neither free nor occupied and can be readily derived by subtracting the occupied and free areas from the entire area. Figure 6 illustrates the 2D area segmentation on LiDAR point clouds obtained in real-world environments.

Unlike conventional grid-based occupancy maps [29, 36, 51], our occupancy map splits areas using polygon representation. This approach offers two significant benefits over grid representation: (1) polygons can represent arbitrary shapes with better precision; (2) by adjusting the outline smoothing factor, polygon representation is more flexible to strike the right balance between reconstruction accuracy and data size. For instance, LiDAR points are too large, bounding boxes cannot represent arbitrary shapes, and grids are not data-efficient as a large empty area could involve lots of grids.

Motion estimation. In addition to on-road object detection, motion state is also needed in *RAO*. The traditional motion modeling-based approaches [56, 65] such as kinematic models cannot be used for occupancy maps, since (1) one can hardly obtain accurate motion parameters of on-road objects required by the motion modeling and (2) the shape of occupied areas alters over time due to ever-changing visual angles. To address these challenges, we leverage similarity metrics that are specific to point clusters (*e.g.,* the distribution of points), to estimate the motion of objects over frames.

In *RAO*, we design a density-aware multi-object tracking algorithm for point clusters. Inspired by AB3DMOT [59], a baseline for 3D multi-object tracking, we utilize the affinity matrix[2] for multi-object tracking and re-design the matrix as follows: Given any two objects (*i.e.,* occupied areas), we define the affinity score as a weighted sum of two items, (1) the distance between the mass centers of two clusters' convex hulls, and (2) the difference in point density as the number of points divided by the area of the convex hull. Convex hulls can be directly calculated from point clouds without the need for additional object detection algorithms, minimizing potential sources of inaccuracies. The affinity score of the two clusters is given by Equation 2, where X_1 and X_2 denotes two clusters in consecutive frames and w_1, w_2 are coefficients.

$$f(X_1, X_2) = w_1 \cdot L_2(X_1, X_2)^{-1} + w_2 \cdot \left| \frac{\textsc{Size}(X_1)}{\textsc{Area}(X_1)} - \frac{\textsc{Size}(X_2)}{\textsc{Area}(X_2)} \right| \tag{2}$$

We then estimate an object's motion based on its point clusters in two consecutive frames, denoted by X_i^t and $X_i^{t'}$ where i is the object ID and t/t' ($t' > t$) are the timestamps of the two frames. We first select the mass center of the earlier occupied area as one reference point (it will act as the "center" of the object in the later perspective transformation process). Both point clusters are transformed in order to use this reference point as their origin. We denote the centering transformation by transformation matrix $T_{i,center}^t$. Formally,

$$Y_i^t = T_{i,center}^t \cdot X_i^t, \quad Y_i^{t'} = T_{i,center}^t \cdot X_i^{t'}, \tag{3}$$

where Y_i^t and $Y_i^{t'}$ are clusters centered by the reference point.

We then apply the point cloud registration algorithm [25] to discover a transformation $T_{reg}^{t'}$ that transforms Y_i^t to minimize its squared distance with $Y_i^{t'}$ as follow:

$$T_{i,reg}^{t'} = \arg\min_T L_2(T \cdot Y_i^t, Y_i^{t'}). \tag{4}$$

Leveraging the rich features of 3D point distribution, the registration between Y_i^t and $Y_i^{t'}$ can accurately capture the pose change over two consecutive frames. Next, we normalize the transformation matrix $T_{i,reg}^{t'}$ to represent transformation per time unit. Specifically, we extract a translation vector and a rotation vector from $T_{i,reg}^{t'}$, and divide them by the time difference $(t' - t)$ to obtain translation and rotation per time unit. Then we reconstruct the transformation matrix per time unit as $T_{i,unit}^{t'}$. We refer to this operation as Scale:

$$T_{i,unit}^{t'} = \textsc{Scale}(T_{i,reg}^{t'}, \frac{1}{t' - t}). \tag{5}$$

We also update the reference point in timestamp t to t'. Despite on different timestamps, two reference points refer

[2]The affinity matrix hosts scores of similarity between possible object pairs across two frames. Similar objects tend to be linked in the same track.

to the same position on the object itself as

$$T^{t'}_{i,center} = T^{t}_{i,center} \cdot (T^{t'}_{i,reg})^{-1}. \quad (6)$$

We store both $T^{t'}_{i,center}$ and $T^{t'}_{i,unit}$ as the estimated motion at timestamp t', which will be used in occupancy flow prediction (§4.4). This motion estimation task is executed iteratively when a new frame arrives. In *RAO*, we parallelize the motion estimation process of different objects in order to reduce processing latency.

4.4 Occupancy Flow Prediction

Occupancy flow prediction is designed for "synchronizing" data from different vehicles. A consumer can synchronize its available occupancy maps to a future LiDAR scanning cycle during data request generation. Similarly, during data response generation, a producer can synchronize the latest LiDAR image to the timestamp requested by the consumer.

Occupancy map prediction. *RAO* is able to predict the future state of the occupancy map at timestamp t', based on an existing motion-aware occupancy map at timestamp t, where $t' > t$. As discussed in §4.3, each occupied area is associated with a motion vector, containing the centering transformation matrix $T^{t}_{i,center}$ and unit-time transformation matrix $T^{t}_{i,unit}$, where i indicates the ID of the occupied area. Assuming the vertices of the occupied area i at timestamp t is X^{t}_{i} (occupied areas are represented by polygons), we can predict the location of such vertices at timestamp t' as:

$$X^{t'}_{i} = (T^{t}_{i,center})^{-1} \cdot \textsc{Scale}(T^{t}_{i,unit}, t'-t) \cdot T^{t}_{i,center} \cdot X^{t}_{i}. \quad (7)$$

We first centralize the occupied area by using the latest reference point as the origin and then apply the unit-time transformation. Finally, the occupied area being calculated is mapped back to the LiDAR coordination. Next, to predict the occluded areas. We calculate the area occluded by newly predicted occupied areas and compare it with the occluded area at timestamp t. We remove the occluded area different from the free area S_F and add the new occlusion into the occluded area S_C. Free areas can be updated similarly.

LiDAR image prediction. *RAO* can also predict a future LiDAR image at timestamp t' based on an existing LiDAR image along with the motion-aware occupancy map at timestamp t. The prediction assumes that the time difference $(t'-t)$ is as small as 100 ms (a LiDAR cycle) so that the LiDAR point clouds at two timestamps have a highly similar point distribution on any object. To achieve this, we first extract LiDAR points associated with each occupied area. Following the same approach outlined in Equation 7, we transform the points to the predicted locations and replace the original points in the LiDAR image with the predicted points.

4.5 Data Scheduling

Data scheduling assigns the occluded areas of a consumer to different producers capable of providing sensor data, which determines the efficiency of data sharing. We formulate it as an optimization problem with a goal to maximize the overall data quality for data fusion. The metrics of quality are customized by the consumer since different downstream tasks may value data quality from different perspectives.

Algorithm 1: Consumer data scheduling.

```
Input: Synchronized occupancy maps $M^{(i)} = (S^{(i)}_O, S^{(i)}_F, S^{(i)}_C)$,
       $i \in \{0, 1, \dots, N\}$, including occupied areas $S_O$, free areas
       $S_F$, and occluded areas $S_C$; $M^{(0)}$ is the consumer's map
       while others are producers' maps.
Output: A mapping assigns areas to producers: $A$.
1  $H \leftarrow$ MaxHeap;
2  $A \leftarrow$ HashMap;
3  for Producer $i \in \{1, 2, \dots, N\}$ do
4      for $S^{(i)}_{O,j} \in S^{(i)}_O S$ do
5          $c^{(i)}_j \leftarrow$ Score($S^{(i)}_{O,j}, \cdot$);
6          MaxHeapAdd($H, c^{(i)}_j, S^{(i)}_{O,j}$);
7      end
8  end
9  while $H \neq \emptyset$ do
10     $S^{(i)}_{O,j} \leftarrow$ MaxHeapGet($H$);
11     $S^{(0)}_C \leftarrow S^{(0)}_C - S^{(i)}_{O,j}$;
12     HashMapAdd($A, S^{(0)}_C \cap S^{(i)}_{O,j}, i$);
13 end
```

We propose to solve the optimization problem through a lightweight greedy search algorithm. As defined in Algorithm 1, data scheduling relies on occupancy maps from both the consumer and available producers. Prior to algorithm execution, all occupancy maps need to be (1) synchronized to the timestamp of the next LiDAR cycle using occupancy flow prediction (§4.4), and (2) transformed into the same coordination system using perspective transform (§4.2).

The scheduling algorithm is executed by the consumer and consists of the following steps: (1) For each occupied area in producers' occupancy maps, we calculate a customizable priority score and sort all producers' occupied areas in descending order of priority scores. (2) We process producers' occupied areas one by one, from higher scores to lower scores. For each occupied area, we calculate its intersection with the consumer's occluded areas. If an empty intersection is present, this occupied area is discarded; we then assign all the non-empty intersection areas to the producer. (3) To prevent sharing duplicate data from multiple producers, we remove the assigned area from the list of the consumer's occluded areas. We repeat steps (2) and (3) on all producers' occupied areas until the data assignment process is finished. Note that producers do not share ground points, which are in large size but make little contribution to perception.

We define the priority score as the distance between the producer's LiDAR and the occupied area. A shorter distance leads to a higher LiDAR point density, bringing richer features to downstream tasks. Note that the score definition is customizable for various purposes (*e.g.*, producer reputation).

5 IMPLEMENTATION

We build a prototype of *RAO* for the purpose of evaluation.

Emulator. We implement an emulator of asynchronous multi-vehicle cooperative perception in Python with 1037 lines of code (LOC). The emulator accepts pre-recorded sensor traces of multiple vehicles as input, including LiDAR point clouds and 3D poses of LiDAR sensors over consecutive frames. Note that the emulator is an event-driven system that executes data processing and data transmission asynchronously on multiple vehicle instances. The latency of data processing is directly measured by executing our algorithm implementations while the transmission latency is measured using recorded network traces.

Algorithms. We implement the core algorithms of *RAO*, including occupancy map generation, occupancy flow prediction, and data scheduling, with 870 LOC in Python and 5,053 LOC in C++. The C++ programs are algorithm implementations optimized to minimize the computation overhead. The Python implementation serves as an interface bridging the emulator and C++ programs. For ground detection, we use RANSAC [21]. For object tracking, we use parameters $w_1 = 1$ and $w_2 = 0.01$ for the affinity score and ICP [25] for point cloud registration. Data scheduling employs a distanced-based priority score as described in §4.5.

6 EVALUATION

We conduct an extensive evaluation of *RAO* to demonstrate its performance under realistic traffic scenarios (including using real-world datasets). We answer the following questions to prove how *RAO* satisfies our design goals listed in §3.3: (1) How much performance improvement can *RAO* bring to downstream perception tasks on asynchronous sensor data when compared to state-of-the-art works? (2) What is the coverage of occlusion achieved by *RAO*'s perception sharing? (3) Is *RAO* fast enough to support real-time operations?

We first introduce our experimental setup (§6.1). Then we examine the end-to-end performance (§6.2) on a suite of large-scale multi-vehicle data sets (including both simulation and empirical traces) to analyze *RAO* (§6.3), followed by system overhead analysis (§6.5) and ablation study (§6.4). Finally, we conduct a case study in a real-world setup to show *RAO*'s advantages over the prior art.

6.1 Experimental Setup

Datasets. We evaluate *RAO* using three datasets: (1) a real-world vehicle-infrastructure dataset, DAIR-V2X-C [66], (2) a simulation-based multi-vehicle dataset, CARLA-SUMO, and (3) a real-world V2V dataset collected at *Mcity* [8].

• *DAIR-V2X-C* is a large-scale dataset consisting of 40k asynchronous frames The data is captured by a vehicle-mounted LiDAR and an infrastructure LiDAR deployed at 28 intersections in Beijing. We further process it by selecting consecutive frames of vehicle and infrastructure data, with each scenario spanning over 10 seconds (*i.e.,* 100 frames).
• *CARLA-SUMO* is an asynchronous multi-vehicle dataset that we collect using OpenCDA [4] based on the co-simulation of CARLA autonomous driving simulator [7] and SUMO traffic simulator [5]. The dataset features diverse road scenarios including straight roads, interactions with two-way single and two lanes, and T-intersections. We set the vehicle speed to up to 50 kph, which is consistent with the average speed limit on urban roads [9]. It contains 2-5 CAVs and 10-50 non-CAVs in each scenario and a total of 6880 frames.
• *Mcity* is a mock city for road testing CAV applications in real-world setup, where we deploy 3 Lincoln MKZ vehicles (CAVs) equipped with OxTS RT3000v3 GPS, Velodyne VLP-32C LiDAR, and Cohda MK6C OBU as a C-V2X receiver. We create 8 challenging driving scenarios that involve occluded or far-away objects. For each scenario, we collect LiDAR, GPS, and C-V2X traces from all CAVs for 15 seconds.

Perception models. We adopt Pointpillar [33] as the backbone object detection neural network for perception-related tasks. For DAIR-V2X-C and CARLA-SUMO, we use pre-trained models provided by DAIR-V2X [66] and OpenCOOD [63], respectively. For *Mcity* data, we fine-tune the OpenCOOD model for 10 epochs to adapt it to *Mcity* sensors.

Baselines. We evaluate *RAO* against the traditional single-vehicle perception using local sensors (*Local-only*) and two state-of-the-art early-fusion cooperative perception solutions: (1) *EMP* [69], a V2I-based cooperative perception system where CAVs selectively upload LiDAR data to an edge node which executes perception algorithms. Based on the algorithm of Voronoi diagrams, each CAV shares LiDAR data in close range. (2) *AutoCast* [51], a V2V-based approach that optimally determines shared data areas and schedules data exchanges. Its data-sharing protocol is discussed in §3.

High-fidelity Emulation. We feed the sensor data into our *RAO* prototype (§5) and emulate the network conditions using real cellular network traces [45]. Specifically, the traces are collected from LTE uplink while driving on urban roads. The bandwidth statistics of the traces are 15.67 ± 10.38 Mbps. We conduct the experiments on a Linux machine equipped with a 32-core Intel Xeon CPU and an Nvidia 2080Ti GPU.

6.2 End-to-end System Performance

Using a combination of simulation data and empirically collected traces, we first evaluate the end-to-end system performance of *RAO* to demonstrate how *RAO* benefits downstream perception tasks after data sharing.

Table 1: Object detection accuracy under different multi-vehicle collaboration schemes and datasets.

Traffic Scene	Perception AP@0.5/AP@0.7			
	Local-only	EMP	AutoCast	*RAO*
DAIR-V2X-C	48.99/40.78%	48.82/40.68%	50.36/41.18%	**53.11/43.49%**
CARLA-SUMO	48.63/37.17%	64.08/54.26%	64.91/51.50%	**74.79/62.01%**
- Town05	40.68/30.18%	48.63/38.25%	63.61/39.88%	**69.81/58.72%**
- Town06	65.46/48.30%	73.22/53.22%	67.55/58.47%	**81.72/65.19%**
- Town10HD	40.12/32.58%	64.34/57.18%	69.90/52.50%	**78.53/65.25%**
Mcity	51.51/41.13%	64.88/50.50%	65.76/48.32%	**69.13/51.25%**

Table 2: Coverage of data sharing.

Overhead Metric	Collaboration scheme		
	EMP	AutoCast	*RAO*
Density	0.2132	0.1857	**0.2756**
Coverage	0.3352	0.2601	**0.3585**

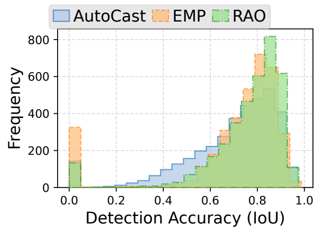

Figure 7: Distribution of perception IoU (accuracy).

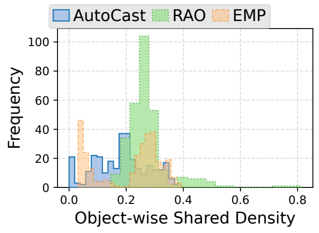

Figure 8: Distribution of object density in merged data.

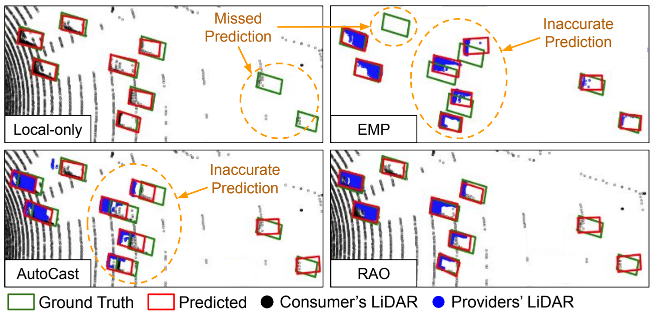

Figure 9: Synchronization of LiDAR data in various schemes.

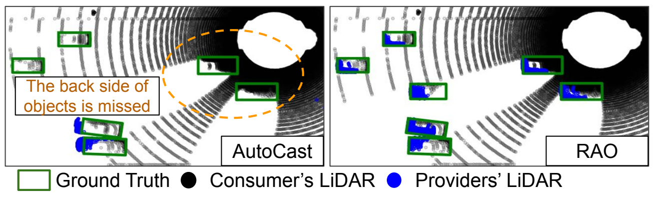

Figure 10: Coverage of blind spots in *RAO* and AutoCast.

Perception accuracy. We run 3D object detection (*i.e.,* perception) on the merged data produced by *RAO* and baseline schemes. We calculate Average Precision (AP) [6] with a threshold of Intersection over Union (IoU) to evaluate the accuracy of perception. IoU measures the overlap between predicted bounding boxes and ground-truth bounding boxes. A higher IoU value indicates more accurate object detection. The AP metric counts prediction bounding boxes with IoU higher than the threshold as true positives. We adopt IoU thresholds 0.5 and 0.7, which are commonly used in evaluating CAV perception. Table 1 summarizes the overall average precision supported by each scheme.

As shown in Table 1, *RAO* is able to achieve significant improvements in perception accuracy on all three datasets, compared to other schemes. Overall, *RAO* improves the AP at IoU over 0.5 (AP@0.5) by 4% – 30% compared to Local-only, 5% – 19% compared to EMP, and 4% – 14% compared to AutoCast. Among the three datasets, CARLA-SUMO demonstrates the best improvements brought by *RAO*. The goal of creating CARLA-SUMO was to complement DAIR-V2X-C's two-LiDAR setup with a larger number of CAVs and non-CAVs on the road in more diverse driving scenarios (§6.1), which is a realistic setting in large-scale deployment of cooperative perception. It explicitly exposes the challenges of synchronization and blind spot detection. We mainly focus on CARLA-SUMO for later analysis in §6.3, §6.4, and §6.5.

To better understand how accurate the object detection is in *RAO*, we further plot the distribution of IoU values in Figure 7. In general, IoU is higher in *RAO*. Compared to AutoCast, *RAO* has significantly more accurate predictions (*e.g.,* IoU 0.7-0.9) and fewer inaccuracy predictions (*e.g.,* IoU 0.2-0.5). This is attributed to the following fact: when a less synchronized point cloud gets fused to the ego vehicle's data, it is likely to cause the predicted bounding boxes of an object to shift away from its ground-truth position. Such a shift significantly lowers the IoU and also the AP of object detection. Note that, the IoU values that fall in the leftmost boxes, are equal to zero for all schemes. These are caused by out-of-range objects.

Coverage of data sharing. *RAO*'s data scheduling is expected to cover blind spots more efficiently. To quantify the efficiency of data sharing, we utilize two metrics: the density of objects and the coverage of objects. A higher point density potentially leads to a higher confidence score in object detection, improving the true positive rate. A higher object coverage, on the other hand, reduces the number of undetected objects, which means a lower false negative rate. To obtain such metrics, we associate each LiDAR point in each frame to a specific object using the ground truth labels (non-object points have no associated objects). Assuming a set of LiDAR images X are generated from different vehicles and they are finally merged to one single LiDAR image Y via the cooperative perception, for each object in the scenario, we can calculate the number of associated LiDAR points in X and Y as N_X and N_Y respectively. We define the metrics as follows, where M is the total number of objects.

$$Density = \frac{1}{M}\sum_{i}^{M} N_Y^{(i)}/N_X^{(i)} \tag{8}$$

$$Coverage = \frac{1}{M}|\{i \in \mathbb{N}, 1 \le i \le M \,|\, N_Y^{(i)} > 0\}| \tag{9}$$

Table 2 lists the results of the above metrics. Compared to AutoCast, *RAO* achieves a 50% higher object density score

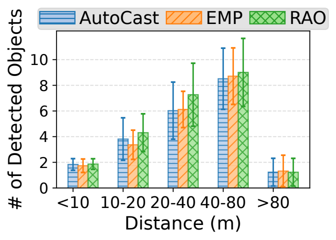

Figure 11: Number of objects detected based on distance.

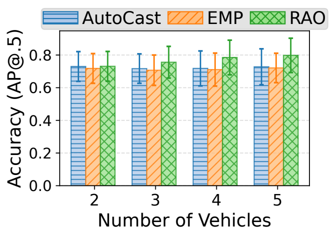

Figure 12: Perception performance *w.r.t.* CAV density.

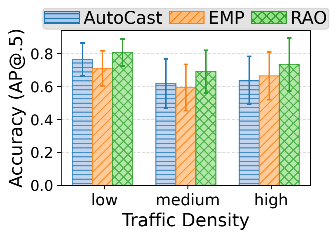

Figure 13: Perception performance *w.r.t.* traffic density.

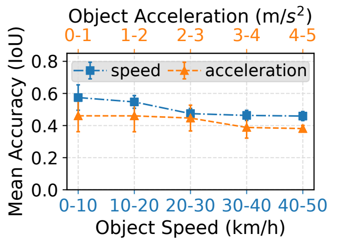

Figure 14: Prediction accuracy *w.r.t.* vehicle motion.

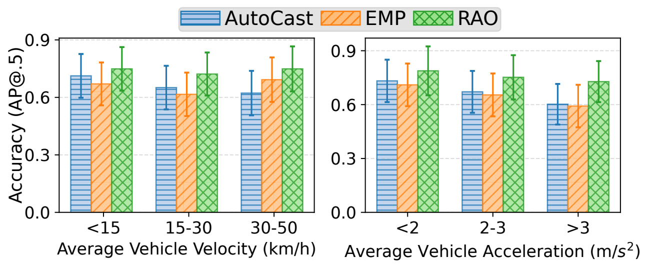

Figure 15: Perception performance *w.r.t.* vehicle motion.

and a 34% higher object coverage. The density and coverage are also 29% and 7% higher than those of the edge-based EMP. In addition, Figure 8 summarizes the distribution of point density per object ($N_Y^{(i)}/N_X^{(i)}$). Over 90% objects in *RAO* have a density score around 0.2-0.4 and very few objects host a high density of points. The result indicates that *RAO* distributes shared LiDAR points on more objects instead of augmenting a few objects that are already clearly visible. Data scheduling in *RAO* successfully covers most objects in the scenarios in an efficient fashion.

Examples. We next show visualized examples to illustrate how *RAO* tackles the problems of asynchronous sensor data (§3.1) and incomplete occlusion estimation (§3.2). As shown in Figure 9, in the baseline schemes without synchronization, the shared points from other CAVs (blue) will fall out of the ground-truth bounding boxes (green) of the objects, resulting in an inaccurate prediction (red). Instead, the LiDAR points from different CAVs are well aligned in *RAO*, thanks to prediction-based synchronization. Figure 10 demonstrates a higher coverage of blind spots in *RAO*. The blue points represent the producer's object points shared with the ego vehicle (consumer). With the consumer's knowledge, fine-grained occluded areas are accurately identified.

6.3 Impacting Factors

We analyze the following impacting factors to understand how *RAO* performs in various driving scenarios.

Distances to the ego CAV. Figure 11 presents the number of correctly detected objects (IoU>0.5) based on the object's distance to the ego vehicle. As shown, the most perception benefits come from the mid-range distances (20-40m and 40-80m range), where *RAO* outperforms AutoCast and EMP with 7.04% to 24.03% (0.62 – 1.40) and 3.45% to 18.62% (0.30 – 1.14) more objects detected through collaboration.

Number of CAVs. We can learn from Figure 12 that, as the number of CAVs participating in the cooperative perception increases, the perception accuracy gains of *RAO* become larger. This is due to the fact that, with more vehicles collaborating, the impact of synchronization grows higher since more vehicles fall out of sync with each other. For instance, when there are 4-5 CAVs, *RAO* brings 6.95% and 7.63% improvements in perception accuracy over AutoCast and EMP, respectively. This is 2× more than the 2-3 CAV collaboration scenarios. More CAVs also increase system overhead but *RAO* is scalable, as analyzed in §6.5.

Traffic density. Next, we consider the overall traffic density. We group the scenarios into three density levels based on the number of all vehicles in each frame. In Figure 13, *RAO*'s performance remains the best across all traffic density settings, with 4.20% to 9.66% improvements.

Vehicle motion. CAV motion affects the accuracy of occupancy flow prediction (§4.4). We group perception targets by their velocity and acceleration, and use IoU between the predicted occupied regions and the ground truth to evaluate prediction accuracy on each group. As shown in Figure 14, prediction accuracy drops gradually as velocity and acceleration increase, but still achieves 0.4 IoU on average in the worst case (0.5 IoU is considered very accurate in perception [6]). Figure 15 further illustrates the impact of vehicle motion on end-to-end perception accuracy. Although minor inaccuracy of prediction exists, *RAO* consistently outperforms AutoCast and EMP in all situations by 3.65% to 13.60%.

6.4 Ablation Study

We do an ablation study to showcase how the two novel designs, *occupancy flow prediction* and *on-demand data sharing* (§4.1), improve the performance of cooperative perception.

Benefits of occupancy flow prediction. We create two variants of *RAO*: (1) one disables prediction algorithms, meaning that providers' LiDAR images are not synchronized with the consumer's data; (2) the other enables ground-truth synchronization, which directly moves objects points according to ground-truth labels and perfectly synchronizes all LiDAR images in data fusion. Figure 16 demonstrates the difference

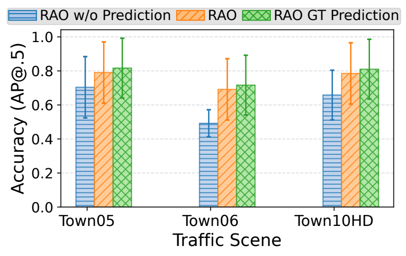

Figure 16: Ablation study on occupancy flow prediction.

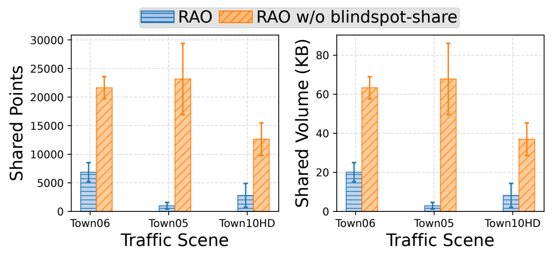

Figure 17: Ablation study on the on-demand consumer-centric data scheduling.

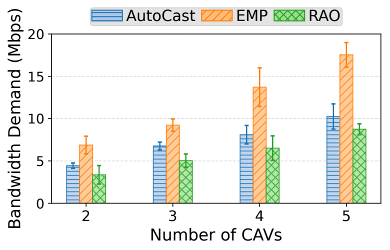

Figure 18: Network bandwidth requirements and scalability study.

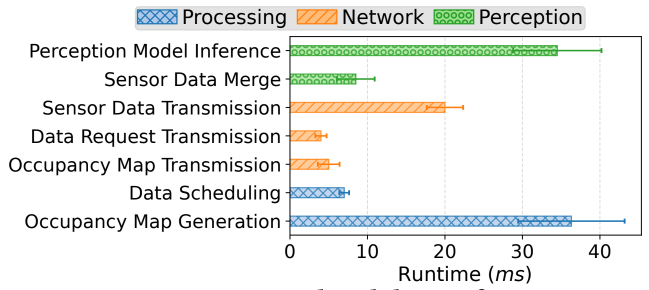

Figure 19: Latency breakdown of *RAO*.

Table 3: Data sharing overhead for each provider-consumer pair per frame (mean±standard deviation).

Metrics	EMP	AutoCast	***RAO***
LiDAR Points	8320±3228	3140±2171	3110±2501
Control data (KB)	<0.1	<0.1	1.77±0.50
Total Volume (KB)	24.37±9.46	9.17±6.36	10.90±7.32

between our proposed *RAO* and the two altered versions. The average precision achieved by employing ground-truth prediction is only 1.97% higher than *RAO*. This suggests that the synchronization module of *RAO* achieves a performance close to the optimal. When disabling prediction, the accuracy of perception drops significantly by 13%, which also highlights the importance of synchronization.

Benefits of on-demand data scheduling. We disable data scheduling in *RAO* (*i.e.,* producers directly share LiDAR images after ground removal) and compare *RAO* with the downgraded version. As shown in Figure 17, in all three traffic scenarios, selectively sharing data in the occlusion region reduces the data volume of sharing by over 68.45% (up to 95.8%). *RAO* can achieve high perception accuracy (Table 1) with such a limited amount of data sharing, validating *RAO*'s accurate identification of critical areas for perception.

6.5 System Overhead

The latency of *RAO* consists of latencies from data processing, network transmission, and perception, as discussed in §3.1. Figure 19 provides a breakdown of different latency types.

Data processing involves occupancy map generation and data scheduling. The occupancy map generation (clustering, motion estimation, *etc.*) is the most compute-intensive in *RAO*, which takes an average of 36.3 ms. Data scheduling is lightweight and can be finished in 7.0 ms. The network transmission latency depends on the size of shared data. A comparison of data sizes is shown in Table 3. For each connected pair of provider and consumer, in each frame, *RAO* shares one LiDAR image containing 3110 points and one occupancy map with 211 polygon vertices on average. Along with tiny metadata such as LiDAR sensor pose and motion information, the total shared data size is around 10.9 KB. The size is similar to that of AutoCast's data (9.17 KB) and is only 44% of EMP's (>24 KB). Note that we remove background points (§4.3) in EMP and AutoCast as well for a fair comparison. The perception latency includes data fusion and model inference. The data fusion module simply concatenates point clouds, which introduces a minor latency of 8.5 ms. The model inference time depends on the specific perception model, and it is 34.5 ms in our implementation.

For each LiDAR image, the sum of computation latency and network transmission latency (*i.e.,* δ_2 in Equation 1) is 80.8 ms, indicating that the asynchronization between LiDAR images is around 80 − 180 ms. Thanks to the occupancy flow prediction, the asynchronization issue is effectively addressed. Compared with single-vehicle perception, the only latency that *RAO* introduces is the data fusion latency (8.5 ms) because each consumer starts perception immediately upon the generation of its local LiDAR image, with other data processing and network transmission tasks running simultaneously in the background.

RAO is scalable and works well under a large number of participating CAVs. The overall bandwidth overhead grows linearly with the number of connected CAVs (Figure 18). Vehicular communication based on C-V2X can provide a bandwidth of up to 100 Mbps [22, 43], far beyond sufficient for our system. The overhead of occupancy map generation is independent with the number of CAVs as each CAV processes its own LiDAR images locally.

6.6 Case study: *RAO*'s Real-world Benefits

In addition to the experiments on large-scale datasets, we conduct another empirical case study using data collected at *Mcity* to demonstrate *RAO*'s superior performance in the real world. As shown in Figure 20, the ego CAV is turning right

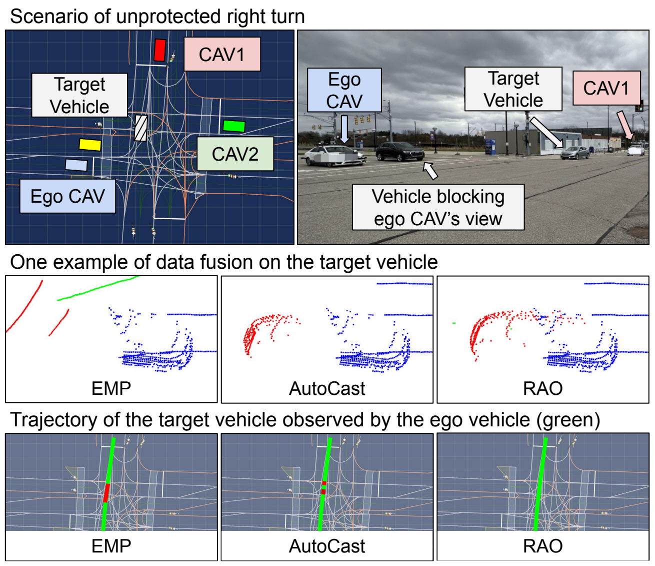
Scenario of unprotected right turn
CAV1
Target Vehicle
CAV2
Ego CAV
Ego CAV
Target Vehicle
CAV1
Vehicle blocking ego CAV's view
One example of data fusion on the target vehicle
EMP
AutoCast
RAO
Trajectory of the target vehicle observed by the ego vehicle (green)
EMP
AutoCast
RAO

Figure 20: Case study of multiple cooperative perception schemes on real-world drive scenes at *Mcity*.

on red and should yield to a vehicle (target) coming from the left. The ego CAV's view toward the target, however, is blocked by obstacles, making it necessary to rely on the other two CAVs to localize the target.

We execute *RAO*, EMP, and AutoCast and find that, (1) EMP is not aware of occlusion and its data scheduling solely depends on distances. Since the LiDAR data of the target area should be shared by the CAV closest to it, at specific locations (red section of the trajectory), other CAVs will not share data for the target even though the target is in the blind spot of the ego vehicle, leading to a failed detection. (2) AutoCast can recognize the blind spot and ask the red CAV to share LiDAR data, but it suffers from a serious synchronization issue (§3.1) that makes object detection unstable in certain frames. (3) In contrast, *RAO* addresses both blind spot detection and synchronization issues simultaneously, producing well-aligned augmented point clouds.

Over the 25 examined frames (2.5 s) in which the target is in the blind spot, *RAO* consistently identifies the target, while EMP and AutoCast fail on 12 and 5 frames, respectively. Notably, detected objects in AutoCast have a relatively higher deviation from the ground truth due to the asynchronous data. In this case, *RAO*'s robust perception earns more time for the ego CAV to react to the incoming traffic.

7 RELATED WORK

RAO relates to prior works in the following areas:

Cooperative Vehicular Sensing. Various efforts have been made on cooperative vehicular sensing. Most existing data sharing systems can be divided into vehicle-to-vehicle (V2V) based schemes [15–17, 32, 40, 50, 51, 63] and vehicle-to-infrastructure (V2I) based schemes [31, 41, 69]. For example, EMP [69], Carcel [31], and LiveMap [41] are V2I examples that use a cloud or edge node to aggregate vehicle data. Moreover, VI-Eye [26] proposes a point cloud registration method for merging vehicle data with infrastructure data. CRCNet [42] explores ways to reduce redundancy in data sharing from the neural network perspective. The release of multi-vehicle perception datasets [38, 63, 66] greatly facilitates the research in multi-vehicle collaboration.

Merging intermediate results. Some of the works [15, 58] advocate merging intermediate results generated from sensor data to loosen the bandwidth requirements. Inevitably, different vehicles may adopt various perception architectures and thus their intermediate data may not be compatible. Xu *et al.* [61] and Qiao *et al.* [49] develop frameworks to tolerate the differences among feature structures. However, sharing raw data is more generalizable since no prior knowledge is needed to make use of the universal format of sensor data. Domain adaptation for feature merging cannot be directly applied to unseen structures and requires retraining.

Vehicle Data Synchronization. In real-world driving scenarios, it is extremely hard to guarantee vehicles generate sensor data at the same pace. There are a few works that tackle the synchronization problem in cooperative data sharing. V2VNet [58] proposes a joint perception and prediction framework to compensate for delay from asynchronous inputs. SyncNet [35] adapts vehicles' asynchronous perceptual features to the same timestamp using a latency compensation module. VIPS [54] leverages graph matching to align detected objects across scenes. However, they consider feature- or object-level fusion while we focus on synchronizing raw sensor data by tracking clusters in occupancy maps.

8 CONCLUDING REMARKS

In this work, we design, implement, and evaluate *RAO*, a real-time occlusion-aware multi-vehicle collaboration system running on asynchronous sensors. With the sophisticated integration of motion-aware occupancy map, occupancy flow prediction, and data scheduling, *RAO* intelligently shares sensor data filling blind spots of other vehicles with minimized network bandwidth consumption and accurately aligns data collected under real-world asynchronous sensor settings through prediction. We believe that our approach to tackling the synchronization issue in sensor sharing can spur a new wave of autonomous driving applications based on multi-vehicle collaboration, and ultimately accelerate their real-world deployment.

ACKNOWLEDGMENTS

We would like to thank our anonymous shepherd and reviewers for their valuable comments and feedback. This work was supported in part by NSF under Grants CNS-1930041, CMMI-2038215, and CNS-2112562.

REFERENCES

[1] 2022. 5G Automotive Association. https://5gaa.org/.

[2] 2022. Autoware: Open-source software for self-driving vehicles. https://github.com/Autoware-AI.

[3] 2022. Baidu Apollo. https://apollo.auto.

[4] 2022. OpenCDA. https://github.com/ucla-mobility/OpenCDA/.

[5] 2022. SUMO: Simulation of Urban Mobility. https://www.eclipse.org/sumo/.

[6] 2022. The KITTI benchmark suite. https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d.

[7] 2023. CARLA: Open-source simulator for autonomous driving research. https://carla.org/.

[8] 2023. Mcity - University of Michigan. https://mcity.umich.edu/.

[9] 2023. Speed limits in the United States. https://en.wikipedia.org/wiki/Speed_limits_in_the_United_States_by_jurisdiction.

[10] 2023. Velodyne LiDAR. https://velodynelidar.com/.

[11] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. 2020. Cooperative perception for 3D object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[12] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. 2013. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company.

[13] David Avis and David Bremner. 1995. How good are convex hull algorithms?. In *Proceedings of the eleventh annual symposium on Computational geometry*. 20–28.

[14] Hanlin Chen, Brian Liu, Xumiao Zhang, Feng Qian, Z Morley Mao, and Yiheng Feng. 2022. A Cooperative Perception Environment for Traffic Operations and Control. *arXiv preprint arXiv:2208.02792* (2022).

[15] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. 2019. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. 88–100.

[16] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. 2019. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 514–524.

[17] Jiaxun Cui, Hang Qiu, Dian Chen, Peter Stone, and Yuke Zhu. 2022. COOPERNAUT: End-to-End Driving with Cooperative Perception for Networked Vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 17252–17262.

[18] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. 2022. Exploiting Rigidity Constraints for LiDAR Scene Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12776–12785.

[19] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. 2011. On the segmentation of 3D LIDAR point clouds. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2798–2805.

[20] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2017. An online multi-robot SLAM system for 3D LiDARs. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1004–1011.

[21] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[22] Moinak Ghoshal, Z Jonny Kong, Qiang Xu, Zixiao Lu, Shivang Aggarwal, Imran Khan, Yuanjie Li, Y Charlie Hu, and Dimitrios Koutsonikolas. 2022. An in-depth study of uplink performance of 5G mmWave networks. In *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*. 29–35.

[23] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. 2021. Weakly supervised learning of rigid 3D scene flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5692–5703.

[24] In-Chul Ha. 2008. Kinematic parameter calibration method for industrial robot manipulator using the relative position. *Journal of mechanical science and technology* 22, 6 (2008), 1084–1090.

[25] Ying He, Bin Liang, Jun Yang, Shunzhi Li, and Jin He. 2017. An iterative closest points algorithm for registration of 3D laser scanner point clouds with geometric features. *Sensors* 17, 8 (2017), 1862.

[26] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. 2021. VI-eye: semantic-based 3D point cloud registration for infrastructure-assisted autonomous driving. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 573–586.

[27] Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. 2022. Where2comm: Communication-Efficient Collaborative Perception via Spatial Confidence Maps. (2022).

[28] John B Kenney. 2011. Dedicated short-range communications (DSRC) standards in the United States. *Proc. IEEE* 99, 7 (2011), 1162–1182.

[29] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. 2017. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 399–404.

[30] Seong-Woo Kim, Wei Liu, Marcelo H Ang, Emilio Frazzoli, and Daniela Rus. 2015. The impact of cooperative perception on decision making and planning of autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine* 7, 3 (2015), 39–50.

[31] Swarun Kumar, Shyamnath Gollakota, and Dina Katabi. 2012. A cloud-assisted design for autonomous driving. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 41–46.

[32] Swarun Kumar, Lixin Shi, Nabeel Ahmed, Stephanie Gil, Dina Katabi, and Daniela Rus. 2012. Carspeak: a content-centric network for autonomous driving. *ACM SIGCOMM Computer Communication Review* 42, 4 (2012), 259–270.

[33] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12697–12705.

[34] Seokju Lee, Francois Rameau, Fei Pan, and In So Kweon. 2021. Attentive and contrastive learning for joint depth and motion field estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4862–4871.

[35] Zixing Lei, Shunli Ren, Yue Hu, Wenjun Zhang, and Siheng Chen. 2022. Latency-aware collaborative perception. In *European Conference on Computer Vision*. Springer, 316–332.

[36] Hao Li, Manabu Tsukada, Fawzi Nashashibi, and Michel Parent. 2014. Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging. *IEEE Transactions on Intelligent Transportation Systems* 15, 5 (2014), 2089–2100.

[37] Ruibo Li, Guosheng Lin, and Lihua Xie. 2021. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 15577–15586.

[38] Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. 2022. V2X-Sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters* 7, 4 (2022), 10914–10921.

[39] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. 2018. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural*

Support for Programming Languages and Operating Systems. 751–766.

[40] Hansi Liu, Pengfei Ren, Shubham Jain, Mohannad Murad, Marco Gruteser, and Fan Bai. 2019. FusionEye: Perception Sharing for Connected Vehicles and its Bandwidth-Accuracy Trade-offs. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[41] Qiang Liu, Tao Han, Jiang Linda Xie, and BaekGyu Kim. 2021. LiveMap: Real-time dynamic map in automotive edge computing. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.

[42] Guiyang Luo, Hui Zhang, Quan Yuan, and Jinglin Li. 2022. Complementarity-Enhanced and Redundancy-Minimized Collaboration Network for Multi-agent Perception. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3578–3586.

[43] Lili Miao, John Jethro Virtusio, and Kai-Lung Hua. 2021. PC5-based cellular-V2X evolution and deployment. *Sensors* 21, 3 (2021), 843.

[44] David L Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on communications* 39, 10 (1991), 1482–1493.

[45] Arvind Narayanan, Eman Ramadan, Rishabh Mehta, Xinyue Hu, Qingxu Liu, Rostand AK Fezeu, Udhaya Kumar Dayalan, Saurabh Verma, Peiqi Ji, Tao Li, et al. 2020. Lumos5G: Mapping and predicting commercial mmWave 5G throughput. In *Proceedings of the ACM Internet Measurement Conference*. 176–193.

[46] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. 2021. A variegated look at 5G in the wild: performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 610–625.

[47] Mohammad Naserian, Curtis L Hay, Sayyed Rouhollah Jafari Tafti, and Allan K Lewis. 2021. Process and system for sensor sharing for an autonomous lane change. US Patent App. 16/695,897.

[48] Anshul Paigwar, Özgür Erkent, David Sierra-Gonzalez, and Christian Laugier. 2020. Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2150–2156.

[49] D Qiao and F Zulkernine. 2022. Adaptive Feature Fusion for Cooperative Perception using LiDAR Point Clouds. *arXiv preprint arXiv:2208.00116* (2022).

[50] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. 2018. AVR: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 81–95.

[51] Hang Qiu, Pohan Huang, Namo Asavisanu, Xiaochen Liu, Konstantinos Psounis, and Ramesh Govindan. 2021. Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. *arXiv preprint arXiv:2112.14947* (2021).

[52] Alexander Rabitsch, Karl-Johan Grinnemo, Anna Brunstrom, Henrik Abrahamsson, Fehmi Ben Abdesslem, Stefan Alfredsson, and Bengt Ahlgren. 2020. Utilizing multi-connectivity to reduce latency and enhance availability for vehicle to infrastructure communication. *IEEE Transactions on Mobile Computing* (2020).

[53] Frédéric Rivard, Jonathan Bisson, François Michaud, and Dominic Létourneau. 2008. Ultrasonic relative positioning for multi-robot systems. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 323–328.

[54] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. 2022. VIPS: real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 133–146.

[55] Zhiying Song, Fuxi Wen, Hailiang Zhang, and Jun Li. 2022. An Efficient and Robust Object-Level Cooperative Perception Framework for Connected and Automated Driving. *arXiv preprint arXiv:2210.06289* (2022).

[56] Jan Erik Stellet, Fabian Straub, Jan Schumacher, Wolfgang Branz, and J Marius Zöllner. 2015. Estimating the process noise variance for vehicle motion models. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 1512–1519.

[57] Vladimir Vukadinovic, Krzysztof Bakowski, Patrick Marsch, Ian Dexter Garcia, Hua Xu, Michal Sybis, Pawel Sroka, Krzysztof Wesolowski, David Lister, and Ilaria Thibault. 2018. 3GPP C-V2X and IEEE 802.11 p for Vehicle-to-Vehicle communications in highway platooning scenarios. *Ad Hoc Networks* 74 (2018), 17–29.

[58] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. 2020. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *European Conference on Computer Vision*. Springer, 605–621.

[59] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 2020. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS* (2020).

[60] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. 2018. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1887–1893.

[61] Runsheng Xu, Jinlong Li, Xiaoyu Dong, Hongkai Yu, and Jiaqi Ma. 2022. Bridging the Domain Gap for Multi-Agent Perception. *arXiv preprint arXiv:2210.08451* (2022).

[62] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. 2022. V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer. In *European Conference on Computer Vision*. Springer, 107–124.

[63] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. 2022. OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2583–2589.

[64] Zhigang Xu, Xiaochi Li, Xiangmo Zhao, Michael H Zhang, and Zhongren Wang. 2017. DSRC versus 4G-LTE for connected vehicle applications: A study on field experiments of vehicular communication performance. *Journal of advanced transportation* 2017 (2017).

[65] Jingang Yi, Hongpeng Wang, Junjie Zhang, Dezhen Song, Suhada Jayasuriya, and Jingtai Liu. 2009. Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. *IEEE transactions on robotics* 25, 5 (2009), 1087–1097.

[66] Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, et al. 2022. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21361–21370.

[67] Yunshuang Yuan, Hao Cheng, and Monika Sester. 2022. Keypoints-Based Deep Feature Fusion for Cooperative Vehicle Detection of Autonomous Driving. *IEEE Robotics and Automation Letters* 7, 2 (2022), 3054–3061.

[68] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. 2017. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5067–5073.

[69] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2021. EMP: edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 545–558.