# EMP: Edge-assisted Multi-vehicle Perception

Xumiao Zhang[†], Anlan Zhang[‡], Jiachen Sun[†], Xiao Zhu[†]
Y. Ethan Guo[⋄], Feng Qian[‡], Z. Morley Mao[†]
[†]University of Michigan, [‡]University of Minnesota, [⋄]Uber Technologies, Inc.

## ABSTRACT

Connected and Autonomous Vehicles (CAVs) heavily rely on 3D sensors such as LiDARs, radars, and stereo cameras. However, 3D sensors from a single vehicle suffer from two fundamental limitations: vulnerability to occlusion and loss of details on far-away objects. To overcome both limitations, in this paper, we design, implement, and evaluate EMP, a novel edge-assisted multi-vehicle perception system for CAVs. In EMP, multiple nearby CAVs share their raw sensor data with an edge server which then merges CAVs' individual views to form a more complete view with a higher resolution. The merged view can drastically enhance the perception quality of the participating CAVs. Our core methodological contribution is to make the sensor data sharing scalable, adaptive, and resource-efficient over oftentimes highly fluctuating wireless links through a series of novel algorithms, which are then integrated into a full-fledged cooperative sensing pipeline. Extensive evaluations demonstrate that EMP can achieve real-time processing at 24 FPS and end-to-end latency of 93 ms on average. EMP reduces the end-to-end latency by 49% to 65% compared to the traditional vehicle-to-vehicle (V2V) sharing approach without edge support. Our case studies show that cooperative sensing powered by EMP can detect hazards such as blind spots faster by 0.5 to 1.1 seconds, compared to a single vehicle's perception.

## CCS CONCEPTS

• **Networks** → **Cyber-physical networks**; **Cloud computing**;
• **Applied computing** → **Transportation**.

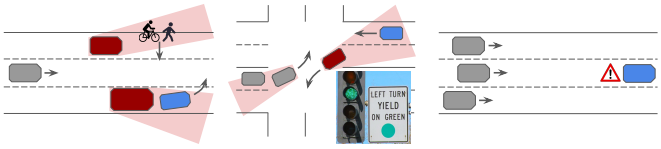
## KEYWORDS

Autonomous Cars, Edge Computing, Cooperative Sensing, LiDAR

**ACM Reference Format:**
Xumiao Zhang[†], Anlan Zhang[‡], Jiachen Sun[†], Xiao Zhu[†], Y. Ethan Guo[⋄], Feng Qian[‡], Z. Morley Mao[†]. 2022. EMP: Edge-assisted Multi-vehicle Perception. In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '21), January 31-February 4, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3447993.3483242

## 1 INTRODUCTION

Connected and autonomous vehicles (CAVs) are expected to transform the ground transportation systems by significantly improving road safety [4] and traffic efficiency [6]. 3D sensors such as LiDAR, radars, and stereo cameras are extremely important to CAVs as the sensors are their "eyes" that continuously sense the surrounding environment. However, these sensors suffer from two fundamental limitations. First, they are vulnerable to occlusion. Because of the rectilinear propagation of light, these sensors cannot perceive objects occluded by non-transparent objects. Second, similar to human eyes, the farther an object is, the fewer details the sensors can capture. Take LiDAR as an example, it emits uniform laser pulses and constructs the environment based on the pulses reflected from objects. Therefore, the density of the pulses and henceforth the perception resolution decrease with increasing distance.



(a) Blind Spots (b) Unprotected Left Turn (c) Broken Down Vehicle

**Figure 1: Use cases of multi-CAV sensor data sharing.**

To overcome the above limitations, nearby CAVs can *share* their sensor data so that each vehicle can have a more complete view with a higher resolution compared to the view constructed from its own sensors. We consider three use cases of sensor sharing among nearby vehicles, as shown in Figure 1.

• Scenario 1 (blind spots from blocking vehicles): A pedestrian and a cyclist are crossing a street, and a blue vehicle is changing to the center lane. However, the gray vehicle cannot see them due to the occlusions of the two red vehicles. This can be resolved by sharing the sensor data from either the red vehicle with the gray vehicle.
• Scenario 2 (blind spots from turning): A gray vehicle is turning left at an intersection without a protected left-turn signal [12]. Meanwhile, a blue vehicle, which the gray vehicle cannot see, is traveling straight from the opposite direction, causing a potential collision. This risk can be eliminated if the red vehicle shares its sensor data with the gray vehicle.
• Scenario 3 (distance-induced limited visibility): The blue vehicle breaks down in the middle of a road. Several oncoming gray vehicles cannot detect it from far away due to the low sensor data resolution. By aggregating their observed data, the broken down vehicle is more likely to be detected much sooner, preventing potential accidents.

**Sharing Raw Sensor Data as Opposed to Processed Data.** Several studies explored vehicles sharing *processed data* such as information of detected objects [28, 47]. We instead advocate sharing *raw sensor data* (when network resources permit) due to several limitations of processed data. First, its limited data granularity cannot support application-specific requirements. In other words, there will be a loss of information during data processing. For example, in Figure 1c, if none of the three gray vehicles can detect the blue vehicle, combining their processed data is ineffective, whereas sharing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*ACM MobiCom '21, January 31-February 4, 2022, New Orleans, LA, USA*
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8342-4/22/01…$15.00
https://doi.org/10.1145/3447993.3483242

raw sensor data may lead to successful detection. Second, sharing processed data lacks generality, and may not be compatible with diverse CAV applications. In contrast, raw sensor data has a simple, fundamental, and universal data format to flexibly support a wide range of CAV applications. Traditionally, sharing raw sensor data was constrained by limited network resources, but this is being changed by high-speed wireless networks such as 5G [51, 53, 70].

**Sharing Raw Sensor Data in a Scalable Manner.** There are a limited number of works that do allow vehicles to share raw sensor data [29, 54, 56], but at a very limited scale. They all take a vehicle-to-vehicle (V2V) sharing approach which suffers from *poor scalability*. As illustrated in Figure 1, oftentimes *multiple* vehicles need to get involved in sensor data sharing, particularly for congested roads. However, when many vehicles need to share their sensor data over V2V, each vehicle has either to perform multicast/broadcast, which suffers from low throughput in particular under mobility [71], or to unicast multiple copies of the data, incurring high delay and bandwidth overheads. Furthermore, a vehicle may not have enough computational resources to process other vehicles' data at line rate.

Differing from all existing work, in this paper, we develop a system called EMP that *scales up* multi-vehicle sensor data sharing through edge computing [37, 60, 62]. We define *edge* to be computing and storage resources in close proximity to the vehicles, which provides low network latency to each vehicle. In our scheme, nearby vehicles upload their sensor data to the edge which creates a global view by merging individual vehicles' data. The edge can then run customized CAV algorithms and return the corresponding results (*e.g.,* detected vehicles as shown in Figure 2). With the edge support, each vehicle's workload and network bandwidth usage can be drastically reduced compared to the V2V scheme. Note that EMP does not fully replace a CAV's local processing, which is instead *enhanced* by EMP. For example, the local object detection results and the results from the edge can be combined to increase the detection coverage and accuracy. When there is a network blackout or the edge is unavailable, vehicles can always fall back to the local mode.

**Principled Spatial Partition.** The key technical merit of this paper is to address the core algorithmic challenge for EMP: how do CAVs efficiently share their raw sensor data? Ideally, CAVs can cooperatively create a disjoint *spatial partition* of the environment, where (1) each CAV uploads only sensor data in its proximity, and (2) the union of all CAVs' sensor data forms the entire surrounding environment. This strategy strikes a desired balance between bandwidth consumption and data quality: there is no overlap among CAVs' data so no bandwidth is wasted; meanwhile, as mentioned earlier, each CAV's close proximity has the highest sensor data quality. We find that mathematically, such a desired partition can be generated by a Voronoi Diagram [25] where the area that each vertex $v$ (a CAV in our context) belongs to consists of the points whose distances to $v$ are less than or equal to those to any other vertices. This key property nicely satisfies the above "proximity" requirement of EMP.

**Adapting to Available Network Resources.** While partitioning based on Voronoi diagrams is effective, it does not consider the available network bandwidth. For example, if the available bandwidth of a CAV is low, then it should upload less data. This can be realized by adjusting its uploaded area's boundary in the Voronoi diagram. We develop a robust algorithm that adaptively adjusts the sensor data uploading area of each vehicle (*i.e.,* the boundaries in the Voronoi diagram) in real time according to the estimated bandwidth of each CAV. In this way, vehicles with slow wireless connections can partially "offload" their uploading tasks to their neighboring vehicles.

**Adapting to Network Resource Uncertainty.** Wireless network conditions are known to be highly fluctuating, in particular under mobility. EMP embraces this through three mechanisms. First, it assigns priorities to each CAV's to-be-uploaded data, to ensure that important portions (*e.g.,* those that cannot be covered by other CAVs' data) are uploaded first, so they are the least vulnerable to the network resource uncertainty. Second, the above scheme naturally provides redundancy for regions that are perceivable from more than one CAV, thus boosting the resilience to the network condition fluctuations. Third, in order to minimize the bandwidth waste incurred by the above redundancy, the edge employs a lightweight graph-based scheduling algorithm to efficiently detect if the entire environment is fully uploaded in real time.

**Implementation and Evaluations.** We incorporate the above algorithms into an edge-assisted multi-vehicle perception system developed by us. Our system consists of a full-fledged cooperative sensing pipeline including sensor data uploading, edge-side data merging, 3D object detection, and vehicle-side perception enhancement using the edge-side results. The above components are judiciously pipelined to ensure good runtime performance. We evaluate our system through extensive emulations using photorealistic sensor data and real-world LTE/60GHz network traces, and real-world live tests. Our key results consist of the following:

- EMP can achieve real-time processing at 24 FPS and end-to-end latency of 86 – 102 ms for the full partitioning-uploading-merging-detection pipeline, when 2 to 6 vehicles are involved in sensor sharing. EMP reduces the end-to-end latency by 49% – 65% compared to its V2V sharing counterpart.
- Compared to having all vehicles upload their full frames, EMP's approach to adaptively uploading sensor data in vehicles' proximity incurs a negligible perception accuracy loss (0.1% – 2.2%) when vehicle detection is performed on the merged view. Meanwhile, EMP's approach leads to a significant bandwidth usage reduction of 32% – 58%.
- We conduct case studies under realistic traffic scenarios and show that cooperative sensing powered by EMP can detect dangers (*e.g.,* occluded vehicles in blind spots and far-away vehicles) earlier by 0.5 to 1.1 seconds, compared to a single vehicle's perception, leading to successful collision avoidance.
- To complement the real system results, we also conduct large-scale simulations involving up to 20 vehicles. The results further showcase the scalability and robustness of EMP under diverse road traffic and wireless network conditions.

Overall, EMP is to our knowledge the first system that enables edge-assisted multi-vehicle perception through raw sensor data sharing. We make two major contributions in this paper: (1) From the algorithmic perspective, we develop robust algorithms for scalable, adaptive, and resource-efficient sensor data sharing under potentially fluctuating network conditions. (2) From the system perspective, we incorporate our algorithms into a real system that can provide extended perceptual range and detection of occluded objects for CAVs.

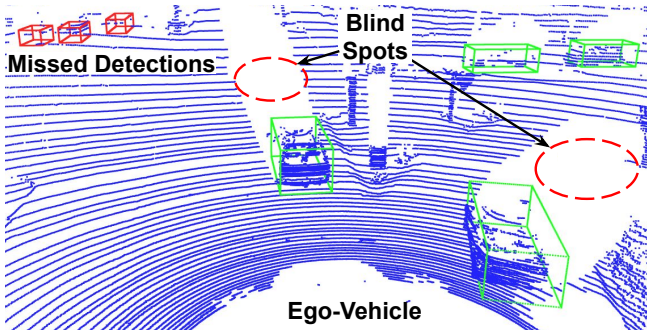



**Figure 2: A LiDAR point cloud (blue) and detected object bounding boxes (green). The ego-vehicle fails to detect distant objects and loses information in the blind spots (red).**

## 2 BACKGROUND AND MOTIVATION

Connected and autonomous vehicles (CAVs) are vehicles that can perceive the surrounding environment and make driving decisions to move safely with various on-board sensors and autonomous driving software. They have wireless communication capability and can thus exchange safety messages with other vehicles or infrastructures to enhance their situational awareness. LiDAR (Light Detection and Ranging) [17, 20] is one of the major on-board sensors. A LiDAR sensor functions by emitting uniform laser pulses at different angles and capturing their reflections from objects. Then it calculates the distance to those objects and generates a 3D point cloud which consists of point coordinates relative to the sensor, to represent the surroundings. Compared with cameras, LiDARs have advantages including a longer range and robustness under poor lighting conditions and inclement weather.

### 2.1 Benefit of Sensor Data Sharing

There are limitations of on-board sensors: (1) They suffer from occlusion. Because of the rectilinear propagation of light, they cannot "see through" non-transparent objects and can only provide line-of-sight information. (2) The farther, the fewer details they can capture. As LiDAR emits uniformly distributed lights and generates data based on the reflected lights, the data resolution decreases as the object distance from the sensor increases. Figure 2 illustrates these limitations with a LiDAR point cloud. There are several blind spots caused by the occlusion of the vehicles near the ego-vehicle. The ego-vehicle also fails to detect some distant vehicles. All these limitations bring road risks and affect driving efficiency for CAVs.

Different vehicles have views at different locations. It is possible that objects occluded in the views of some vehicles can be easily perceived by some others. Therefore, combining sensor data from vehicles perceiving objects at various perspectives can effectively eliminate occlusions and increase the perception resolution, thus further avoiding potential road hazards, as demonstrated in the examples in Figure 1.

As opposed to sharing processed data such as detected objects, in this paper, we advocate sharing raw sensor data due to several limitations of sharing processed data. First, the information granularity decreases after processing the raw data to a higher layer of data, such as extracted features or detected objects. For example, in Figure 1c, a single gray vehicle may not detect the blue vehicle from far away on its own (*e.g.,* due to long distances or poor weather [11]) and merging their detection results will yield nothing, whereas combining the observations from all the vehicles altogether may lead to successful detection. Second, sharing processed data lacks generality. For example, vehicles may have different representations for processed data (*e.g.,* object classes). One detection algorithm may output car, human, *etc.* while another outputs sedan, bus, cyclist, pedestrian, *etc.* Instead, raw sensor data has a simpler data format to flexibly support a wide range of CAV applications. Furthermore, CAV's local processing takes time. A CAV can share its sensor data once the data is captured and some prepossessing is done. Then it starts the local processing while waiting for the results from the edge. Upon receiving the edge's enhanced results, it can adjust the driving decisions accordingly. In contrast, a CAV cannot share the processed data until it completely finishes the local processing.

### 2.2 Need for an Edge-assisted System

In order to avoid such hazards during daily driving, some existing works leverage sensor data sharing to improve the vehicle's visibility [29, 54, 56]. However, these systems only enable vehicle-to-vehicle (V2V) data sharing which suffers from poor scalability from both the network and computation perspective. **Network overhead**: There are many scenarios (Figure 1) involving multiple vehicles for sensor data sharing. When the number of vehicles grows larger than two, each vehicle has to either send more than once or rely on another vehicle to relay its data, which introduces redundancy, additional delay, and higher bandwidth consumption. **Computational overhead**: Processing data shared from other vehicles involves additional overhead, challenging the limited on-board resources. We examine how the sensor data volume affects the inference time of 3D object detection using a state-of-the-art detection framework, PointPillars [41]. As shown in Figure 5, the inference time is roughly proportional to the number of vehicles which has a positive correlation with the data volume.

Edge computing services are becoming increasingly popular [5, 7, 10]. Edge nodes usually have more computational resources to process aggregated sensor data compared to on-board hardware which is equipped to process single-vehicle data. Communicating with an edge server also involves lower latency compared to using a remote cloud. Unlike V2V sharing, the vehicles only need to upload their data once to an edge node which can process them together.

### 2.3 Challenges

Building such an edge-assisted system that processes vehicle sensor data in real-time still poses several scalability challenges regarding network and computational resources.

- It is extremely hard for existing wireless techniques to support multiple vehicles simultaneously uploading raw sensor data in real-time. A commercial 64-beam LiDAR collects point clouds (~2MB[1]) at 5-20Hz [20], which means the data can be generated at up to 300Mbps. How does the system reduce the data size with little impact on perception performance?
- Although an edge node usually has more computational power than individual CAVs, the processing time grows as the data volume increases as demonstrated in Figure 5. How can the system ensure real-time operation while providing an extended perception range?

[1] A point cloud contains ~130K points (64 vertical angles and 2083 horizontal angles) consisting of location and intensity information ($xyz$-$i$, 4 floating-point numbers).

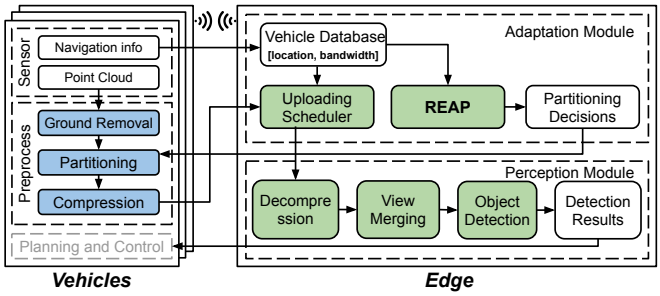



**Figure 3: System Architecture of EMP.**

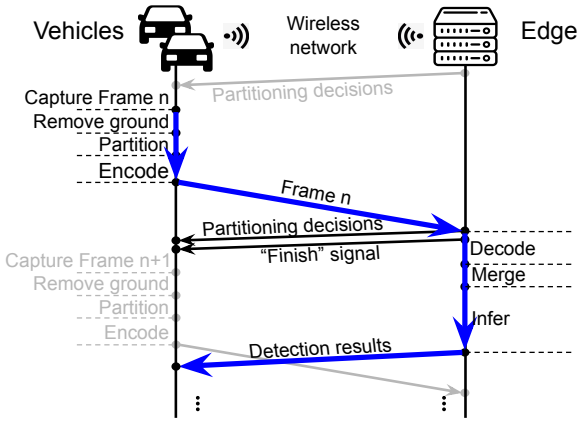



**Figure 4: EMP Data Plane.**

• The available network bandwidths vary across vehicles, leading to different transmission times of data from different vehicles. Plus, wireless networks can fluctuate in particular under high mobility. How does the system adapt to the variability of network resources?
• Vehicles may upload frames at different times. There will be tremendous computational overhead if the edge processes a frame once it is received. In order to process data collected at similar times from different vehicles together, how does the edge determine when it can start processing the current frame and schedule for the next?

## 3 SYSTEM DESIGN

We propose EMP, an Edge-assisted Multi-vehicle Perception system for efficiently sharing sensor data over wireless networks and improving perception accuracy for CAVs. EMP tackles the above challenges through several design decisions: (1) EMP offloads heavy computation of cooperative perception from vehicles to an edge node (§3.1); (2) EMP efficiently partitions point cloud data to reduce network latency (§3.2); and (3) EMP strategically coordinates the uploads from different vehicles (§3.3). EMP also incorporates view merging (§3.4), ground removal and system-level optimizations (§3.5) for boosting the performance of cooperative perception.

### 3.1 Edge-assisted Perception Architecture

At a high level, EMP offloads the cooperative perception from each vehicle to the edge side so that the edge performs object detection based on the aggregated sensor data and provides improved perception results for better driving decisions. To achieve this, EMP connects each vehicle with the same edge node with network channels in two layers, as shown in Figure 3:

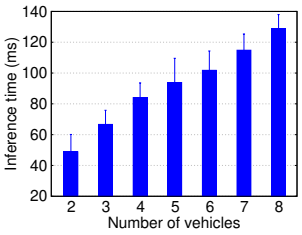



**Figure 5: Inference time.**

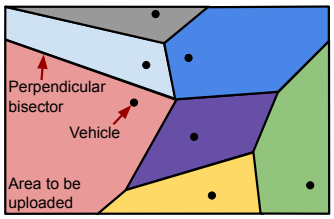



**Figure 6: Voronoi Diagram.**

• **Data Plane** transmits the sensor data from the vehicles to the edge, and performs perception tasks at the edge. As shown in Figure 4, each vehicle preprocesses a single frame of sensor data (the **Preprocessing Module** in Figure 3), and uploads the chunks to the edge. The partitioning is necessary for point cloud data as the size of a single frame can be large and there may not be enough bandwidth to upload full frames from all vehicles to the edge in time. Uploading chunk by chunk allows the edge to leverage partial point cloud data if available. Upon receiving the point cloud chunks, the edge merges these chunks with point cloud data from other vehicles based on the precise locations of all vehicles after decompression, forming a holistic point cloud as the view of the surrounding area. The edge can thus perform 3D object detection [41] on the holistic point cloud, and finally send the detection results back to each vehicle (the **Perception Module** in Figure 3). The results consist of locations, dimensions, headings, and confidence scores of the detected objects. EMP pipelines the vehicle's preprocessing and edge's perception, *i.e.,* a vehicle can start transmitting the next frame of the point cloud before receiving the detection results.
• **Control Plane** optimizes the network transmission of all vehicles according to their locations and network conditions by guiding the point cloud partitioning for vehicles, so that the data to be uploaded by each vehicle is balanced and the edge can construct the holistic point cloud promptly. The partitioning allows each vehicle to upload its surroundings first, with the uploaded area adapted to its available network resources. Before uploading the sensor data, each vehicle sends a control message containing its real-time location to the edge. For each control message received, the edge uses the location, along with other vehicles' locations, to determine the region of the point cloud to upload for each vehicle. The decision region is sent back to the vehicles in the form of multiple line equations representing the region boundaries. Once a new frame of the point cloud is generated, the vehicle partitions the frame following the latest partitioning decision provided by the edge to reduce the data size. All the logic resides in the **Adaptation Module** in Figure 3.

Note that when the network connectivity is poor or the edge is unavailable, the vehicles can always run their local processing for basic services. Besides, while we focus on the assistance from a *single* edge in this paper, the EMP's design can be flexibly extended to the scenarios of *multiple* edge nodes by introducing sensor data sharing among edge nodes based on vehicle locations and a handover mechanism, which we leave for future work.

### 3.2 Edge-assisted Point Cloud Partitioning

Since a full frame of the point cloud data may not be uploaded in time to the edge, the edge partitions the whole area into non-overlapping regions so that each vehicle only uploads a subset of

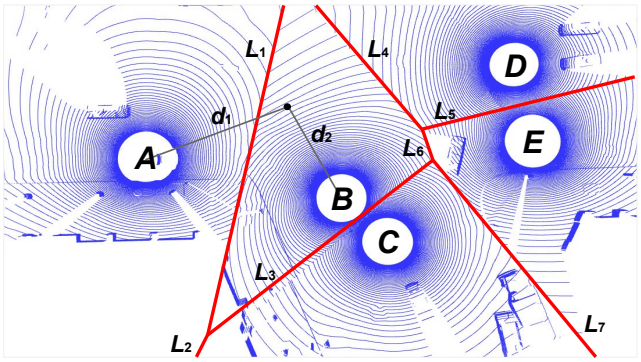



**Figure 7: Naive partitioning through Voronoi Diagram.**

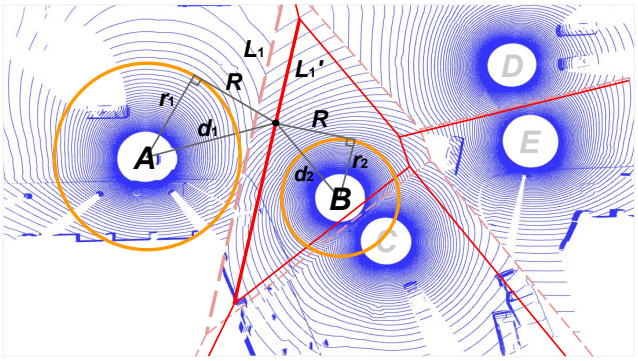



**Figure 8: BW-aware partitioning through Power Diagram.**

the points according to the corresponding decision region, to reduce the amount of upload data.

One intuitive idea of such partitioning is to assign each point in the 2D region (bird-eye view) to the closest vehicle. In this case, the whole region is partitioned into multiple non-overlapping regions close to each vehicle. Since the LiDAR point cloud has more detailed information for the closer region, such partitioning ensures that each region has the finest representation from point clouds of multiple vehicles. Mathematically, such a partition is a Voronoi diagram [25], as shown in Figure 6. For each vertex $v$ (a vehicle in our context), there is a corresponding area that consists of the points whose distances to $v$ are less than or equal to those to any other vertices. Figure 7 visualizes an example of such a partition on a global region consisting of five vehicles. The letters $A - E$ represent the vehicles and $L_1 - L_7$ represent region boundaries, *i.e.*, the perpendicular bisectors in the Voronoi diagram, between two vehicles. For example, any points in $B$'s region are closer to vehicle $B$ than to any other, so we have $d_1 > d_2$ where $d_1$ and $d_2$ are the distances from the point to each vehicle. Note that the boundaries of a region for a vehicle only depend on neighboring vehicle locations, and such boundaries can be derived by finding the perpendicular bisectors between each pair of neighboring vehicles.

While the partitioning based on Voronoi diagrams is simple, it suffers from two limitations. First, such partitioning depends only on the relative location of the vehicles, without considering vehicles' network conditions. This can still lead to a large network transmission time. For example, in Figure 7, when the network bandwidth of vehicle $A$ is much lower than that of vehicle $B$, $A$ can still take a longer time to upload its share of point cloud data than B to the edge, causing the edge to wait longer before leveraging $A$'s point cloud data. Second, even if the initial partition of the region is proportional to the uplink bandwidth of each vehicle, as the vehicles move, the bandwidth can fluctuate significantly and thus lead to longer transmission times for some vehicles.

To address these limitations, we propose REAP, a Region-based Edge-Assisted Partitioning which is bandwidth-aware and adaptive to the bandwidth fluctuations. REAP decides the partitioning boundaries based on both the vehicle's location and estimated uplink bandwidth (§3.2.1). REAP adapts to the fluctuating bandwidth by assigning multiple small chunks to each vehicle for transmission, taking the degree of bandwidth variation into account while partitioning, and dynamically determining when to finish transmission (§3.2.2).

*3.2.1 Bandwidth-aware Partitioning.* The available network resources of different vehicles can very likely vary. To cope with the different wireless uplink bandwidths across vehicles, REAP partitions the global region based on both the vehicle location and the estimated uplink bandwidth. At a high level, REAP achieves this through moving the region boundary between two vehicles towards the vehicle that has lower bandwidth. Such partitioning results in a smaller region to upload for vehicles with low bandwidth and a larger region for those with high bandwidth.

Specifically, REAP uses Power Diagram [24] in Mathematics to determine the precise partitioning boundaries. Recall that a Voronoi diagram draws the perpendicular bisector of the connection between every two neighboring vehicles as the partitioning boundary (Figure 7), which means the distance to the boundary from both vehicles are the same. A power diagram is a form of weighted Voronoi diagram, in which each vehicle is assigned a *weight*, and the ratio between the distances of two vehicles to the boundary is positively correlated to the ratio of the corresponding weights of the two vehicles. By adjusting the weights of the vehicles based on their corresponding estimated uplink bandwidth, we can thus make the partitioned region adapt to the vehicle's uplink bandwidth (the bandwidth usage is largely proportional to the uploaded area).

Figure 8 visualizes how the weights of vehicle $A$ and $B$, $r_1$ and $r_2$ respectively, determine the boundary between the two vehicles. The boundary here is the radical axis of two circles centered on these two vehicles and the weights are the circle radii. Any point on the radical axis has the same power distance ($R$) to both circles. That is, $R^2 = d_1^2 - r_1^2 = d_2^2 - r_2^2$. As the estimated uplink bandwidth of $A$, and thus $r_1$, increases, the boundary $L_1$ is moved to $L_1'$. As a result, vehicle $A$ with a better network condition is scheduled to upload more data. Note that the bandwidth $bw$ (data volume divided by time) and the weight $r$ (distance) in such a diagram intrinsically have different units. In REAP, we interpret the factor between these two values as a configurable parameter $k$ which reflects the sensitivity of the system to the bandwidth differences, so we have $r = k \times bw$.

To estimate the uplink bandwidth of each vehicle, the edge measures the size and transmission time of each point cloud chunk sent from each vehicle, and computes its bandwidth as the exponentially weighted moving average (EWMA) of the ratio between size and transmission time.

*3.2.2 Adaptation to Bandwidth Fluctuation.* The bandwidth estimation from the edge can be inaccurate as the network condition

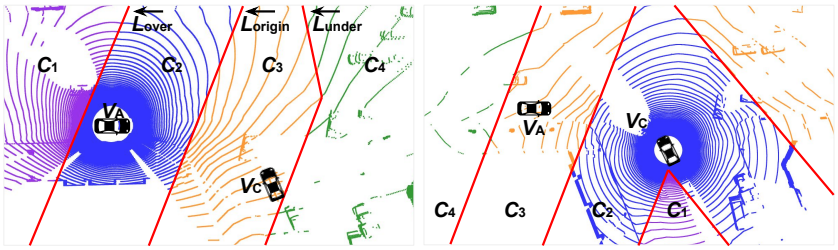



**Figure 9: Vehicle A's (left) and C's (right) point clouds partitioned into chunks by REAP for adaptation to fluctuating bandwidth. $L_{over}$, $L_{origin}$, and $L_{under}$ are region boundaries. $C_1$-$C_4$ represents chunk IDs.**

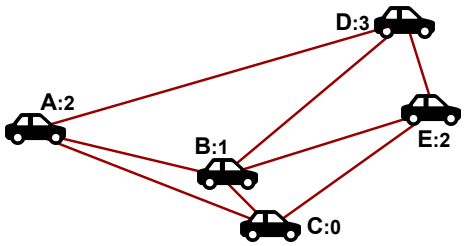



**Figure 10: Neighboring relationships (*e.g.,* A-D) and chunk uploading progresses (vehicleId: chunkId).**

is changing rapidly, especially for vehicles under high mobility. REAP addresses this challenge by further partitioning each vehicle's region into multiple chunks, based on two scenarios, bandwidth underestimation and overestimation. Each chunk is assigned an upload priority to ensure that important portions are uploaded first, so they are least vulnerable to the network resource uncertainty.

Specifically, for each pair of neighboring vehicles, besides the boundary ($L_{origin}$) calculated based on the estimated bandwidths (§3.2.1), REAP determines two additional boundaries, $L_{over}$ and $L_{under}$, by replacing the original estimations $bw_A$ and $bw_B$ (bandwidths of vehicle $A$ and vehicle $B$) with two new pairs of values: (1) $bw_A \times (1-\alpha)$ and $bw_B \times (1+\alpha)$, to account for the extreme case of overestimation of $A$'s bandwidth and (2) $bw_A \times (1+\alpha)$ and $bw_B \times (1-\alpha)$, to account for the extreme case of underestimation. These boundaries together partition a point cloud into smaller chunks, as illustrated in Figure 9[2]. Here $\alpha$ defines the degree of network fluctuation to tolerate and thus is correlated to the actual network characteristics. We adopt an auto-tuning strategy to set $\alpha$ during runtime. More specifically, the edge can adjust the $\alpha$ value using the standard deviation of estimated bandwidth values for different vehicles in the system across a past period of time.

As shown, each vehicle has chunks numbered from 1 to 4. Chunk 1 ($C_1$) is the area enclosed by $L_{over}$ and is on the side far from neighboring vehicles. $C_1$ should be uploaded in the highest priority because 1) it is the easiest area to upload as it is derived assuming the bandwidth is overestimated, and 2) it has the least overlapping with other point clouds and other vehicles may not be able to help. Chunk 2 ($C_2$) is enclosed by $L_{origin}$ and $C_1$ boundary. If all vehicles upload $C_1$ and $C_2$, the entire area is covered without overlapping, which is the best case. Chunk 3 ($C_3$) is enclosed by $L_{under}$ and $C_2$ boundary. It further extends towards the neighboring vehicles and is closer to them. $C_2$ of one vehicle can be replaced by its neighbors' $C_3$ in reduced quality. Chunk 4 ($C_4$) is essentially the point cloud excluding the first three areas. It is the least important to the vehicle because this chunk is mostly blocked by its neighbors which can also capture more details. In case one vehicle is suffering from very bad network conditions or just gets disconnected, its neighbors can help by uploading their $C_4$. From Figure 9, we can find that $C_2$ of vehicle $C$ is partially covered by $C_3$ of vehicle $A$ (together with $C_3$s of $A$'s other neighbors). $C_1$ of vehicle $C$ is partially covered by $C_4$ of vehicle $A$.

Each vehicle sequentially uploads from $C_1$ to $C_4$. In this way, vehicles first share areas that can be better captured and may only upload overlapped areas at the later time of the transmission. In different scenarios as the actual bandwidth differs from the estimated bandwidth, such a mechanism allows the edge to receive enough data to construct a holistic view as soon as possible, reducing the transmission time. In short, the goal of the adaptation in REAP is to achieve that a chunk is always finished by the "best" candidates who can provide the most details while other vehicles can help provide data with fewer details to meet real-time requirements, balancing the trade-off between the level of details in the point cloud and the time to start perception at the edge.

[2]The notations (A-E) used in previous figures are kept for consistency and we select vehicle $A$ and vehicle $C$ for better visualization.

## 3.3 Upload Scheduling

According to REAP, each vehicle uploads its chunks sequentially. However, there will be unnecessary bandwidth waste if vehicles keep uploading the remaining chunks after the edge has received enough data to construct the global view. Besides, vehicles may start uploading their frames at different times but it is meaningless for the edge to process a frame without combining frames from other vehicles. Therefore, the edge needs to schedule when the transmission of a frame should be ended and it can start processing.

We propose a scheduling algorithm based on Delaunay Triangulation [26] for the edge to determine from the received data. For a given set ($P$) of discrete points, $P$'s Delaunay Triangulation ($DT$) is a triangulation of P such that no point in $P$ is inside the circumcircle of any triangle in $DT$. In other words, there will be no points inside the triangle formed by joining any three "neighboring" points. Figure 10 shows the vehicle locations and their neighboring relationships. For example, vehicle $A$, $B$, and $D$ are neighbors to each other while vehicle $E$ is not $A$'s neighbor. Vehicle $A$ has neighbor $B$, $C$, and $D$. In this way, when calculating a Voronoi diagram (§3.2, or Power diagram in §3.2.1), only $A$-$B$, $A$-$C$, and $A$-$D$ are considered for vehicle $A$ while $A$-$E$ is not, which reduces the processing overhead compared to deriving perpendicular lines of connections between every two vehicles.

Based on how the chunks of each vehicle are divided, we define three conditions where the edge can determine the transmission of the current frame is finished: (1) $C_1$ and $C_2$ of all vehicles have arrived; (2) $C_2$ of one vehicle has not arrived (e.g., due to limited bandwidth) but the $C_3$ of all its neighboring vehicles have been delivered; (3) Neither $C_1$ nor $C_2$ of one vehicle has arrived but its neighbors finish uploading their $C_3$s and $C_4$s. Once any of these conditions is satisfied, the edge broadcasts a "finish" signal to stop all vehicles from uploading the remaining chunks.

As shown in Figure 10, the numbers after the vehicle letters (names) represent the largest IDs of chunks received by the edge.

According to the conditions defined above, if all the numbers are 2, then the entire area is perfectly covered and the edge can notify the vehicles of the end of the transmission. However, although the REAP algorithm enables vehicles to upload chunk by chunk, from $C_1$ to $C_4$, their uploading progress will not be at the same pace and the conditions are not satisfied by all the vehicles at once. Therefore, in order to check the satisfaction of these conditions, we develop a scheduling algorithm as follows: Every time a new chunk is received, the edge will check whether the sum of a vehicle's largest chunk ID and another vehicle's, is greater than 4. If so, that means these two vehicles locally satisfy the conditions and the ridge between them is removed from the set of ridges to be checked. When no ridges in the diagram are left, the entire area is fully covered and the frame is ready to be processed. Otherwise, the edge keeps waiting for the remaining data and rechecks when the next chunk arrives.

## 3.4 View Merging

After receiving frames from different vehicles, the edge performs 3D object detection on the holistic point cloud. However, generated from the perspective of a vehicle, the point cloud frame origin is the vehicle LiDAR sensor. Thus, in order to merge the data collected by different vehicles, the edge needs to transform the points of each point cloud from their original perspectives to a unified coordinate system. Given a target origin and axis orientations, the relative position $(\Delta x, \Delta y, \Delta z)$ and orientation $(\alpha, \beta, \gamma)$ of a vehicle can be derived based on its navigation data (GPS/IMU) by calculating the differences. The edge further generates a translation matrix, $T = [\Delta x, \Delta y, \Delta z]^T$, and three rotation matrices, $R_z(\alpha)$, $R_y(\beta)$, $R_x(\gamma)$. Then, the transformation of a point $P = [X, Y, Z]$ can be calculated as follows: $P_{dst} = R_z R_y R_x \times P + T$. Note that this approach assumes the location data is reasonably accurate, thanks to high-performance localization techniques [16, 21] which can achieve centimeter-level accuracy. Existing point cloud calibration/registration techniques [32, 35, 49, 58] can be applied when the navigation signals are less accurate.

## 3.5 Performance optimizations

We make several optimizations to save bandwidth, reduce end-to-end latency, and improve processing efficiency.

**Ground Removal.** LiDAR sensors collect a significant amount of data from the ground plane which is less useful than the data of surrounding objects for perception. Therefore, EMP detects and removes the ground points before sharing to save bandwidth. We use an algorithm called Random Sample Consensus (RANSAC) [33] assuming that the ground plane is the plane containing the most points in a point cloud. Specifically, EMP randomly picks several points to construct a plane and counts how many points in the point cloud fall near this plane. It repeats until the plane contains enough points. As the height of the sensor (atop the CAV) is known, we can estimate the approximate location of the ground to effectively reduce ground detection time.

**Edge-side Parallelization.** As the edge receives multiple point cloud chunks and locations from different vehicles, the processing of incoming data can be done concurrently. EMP takes decoding, merging, and location updating as individual tasks and parallelizes the tasks for different chunks by scheduling a corresponding task once a data chunk is received or decoded, or real-time navigation data is received. In this way, the edge saves a significant amount of time while waiting for new chunks.

**Pipelining.** To improve the system throughput, *i.e.,* the frame rate that EMP can support, we further pipeline the three parts (vehicle, network, edge) in the system, any of which does not have to wait until the current frame goes through the entire workflow before processing the next available frame. The vehicle is responsible for ground removal, point cloud partitioning, and decoding. After pushing frame $n$ into the send buffer queue, a vehicle can process frame $n+1$ once it is available. Meanwhile, the edge is working on a received frame such as frame $n-1$.

**Cloud Mode.** Although edge nodes are being increasingly deployed [5, 7, 10], there could be areas where no edge nodes are available. In this case, EMP will fall back to rely on a cloud server for data aggregation and processing to provide seamless support. This may lead to a longer transmission latency but CAVs can still benefit from the cooperative perception. We evaluate the impact of EMP's cloud mode on detecting road hazards in §5.4.

# 4 IMPLEMENTATION

We implement EMP [15] in Java and the prototype consists of about 10K lines of code. **Vehicle-side**: Our prototype supports obtaining the incoming sensor data (*e.g.,* point clouds and navigation data) from various sources including both real LiDAR / GPS and recorded traces. The sensor data is provided to the processing pipeline at a configurable and fixed rate (*e.g.,* 10Hz in our experiments). The partitioning module takes as input the coefficients of line equations representing the chunk boundaries received from the edge and then crops out the chunks through linear algebra operations. We modify Draco [14] for LiDAR point cloud compression. The Draco APIs are invoked through JNI. **Edge-side**: The real-time 3D inference is built upon PointPillars [41], a state-of-the-art open-source 3D object detection framework. It is computationally efficient and is adopted by existing industry-level autonomous driving platforms such as Baidu Apollo [8] and Autoware [13]. In the original implementation of PointPillars, the code for the entire object detection pipeline is integrated. We thus separate different modules in the pipeline (model loading, model configuration, inference, *etc.*), and make model loading/configuration a one-time operation to enable fast inference. The edge also uses Draco to decompress the uploaded point clouds. For all other components in Figure 3, we implement them by ourselves. The vehicle and edge communicate through a custom protocol over TCP. Changing the underlying transport protocol to other protocols such as QUIC [42] is straightforward.

Note, EMP is designed for efficient point cloud data sharing. In this paper, we focus on LiDAR point clouds for demonstration while the system is generally compatible with other CAV sensors which capture point cloud data, such as stereo cameras.

# 5 EVALUATION

We evaluate the performance and scalability of EMP under different vehicle and network settings, demonstrating its advantage over vehicle-to-vehicle sharing schemes (§5.2). We examine EMP's enhancement on perception (§5.3). We showcase how EMP improves road safety with driving case studies (§5.4) and show the benefit of sharing raw data over sharing processed data. In addition, we

present the latency contributed by key EMP system components and the processing throughput improvement brought by system-level optimizations (§5.5). A series of large-scale simulations are conducted to further prove the effectiveness of REAP under a wider range of vehicle numbers (§5.6).

## 5.1 Experimental Setup and Methodology

Due to a lack of open infrastructure support for multi-vehicle experiments, we adopt trace-driven emulation to evaluate EMP in our local testbed and compare our system with existing work. To emulate vehicle behavior in our experiments, instead of running a LiDAR device to generate data in real time, we replay LiDAR traces from a multi-vehicle LiDAR dataset we collected in advance. Our setup considers both diverse driving scenarios and realistic network conditions to comprehensively evaluate the performance and scalability of EMP. We also conduct live tests to demonstrate that EMP can work well under real networks.

• **Comparing EMP with Existing Work.** We consider two variants of EMP to evaluate our design choices: (1) **EMP-Naïve**: EMP without REAP adaptive partitioning and scheduling, *i.e.,* vehicles upload full point cloud frames to the edge node; (2) **EMP**: EMP with all components enabled. We further compare them with vehicle-to-vehicle sharing schemes: (1) **V2V-Naïve**: each vehicle shares full frames with every other vehicle; (2) **V2V-Pro**: each vehicle shares partial point clouds with other vehicles using REAP partitioning[3]

• **Multi-vehicle LiDAR Dataset.** All existing LiDAR datasets such as KITTI [34] only contain traces collected by a single vehicle, while the evaluation of multi-vehicle perception requires the traces collected from multiple vehicles which are physically proximate at the same time. To fill this gap, we collect the *first* multi-vehicle LiDAR dataset using DeepGTAV-PreSIL [38], a tool to collect synthetic LiDAR traces simultaneously from multiple vehicles in a video game, GTA V. GTA V contains realistic 3D modeling of city landscape, vehicles, stationary objects to emulate real-world scenarios. Besides LiDAR data, DeepGTAV-PreSIL also generates object labels for training machine learning models of perception. We extend the tool to enable panoramic (360°) LiDAR scans besides the default front-view-only settings. We construct our dataset by randomly driving a car in the game and collect sensor data from multiple nearby cars. Our multi-vehicle LiDAR dataset contains driving scenarios in both densely-populated urban areas and open rural areas, with various numbers of vehicles in the scene.

• **Network Conditions.** The vehicle-to-infrastructure networking conditions [55] are emulated by throttling the bandwidth for individual TCP connections between each vehicle and the edge node and adding 10ms latency using Linux `tc` [1]. To acquire realistic uplink bandwidth of cellular networks for our experiments, we collect LTE uplink traces from driving at different urban and rural locations. We run 10-minute 100Mbps UDP uploads for a number of times over AT&T LTE networks on two smartphones (Pixel 2 and Nexus 6), to saturate the uplink. We run `tcpdump` at the server side to record raw packet traces and calculate the uplink throughput every 100 ms. To emulate high-bandwidth networking used by future vehicular communication [30], using a similar approach, we collect uplink bandwidth traces under 60GHz networks (802.11ad) with a stationary NETGEAR Nighthawk X10 AD7200 WiFi router [18] and a moving 802.11ad-compliant laptop. Note the bandwidth statistics of our LTE and 60GHz network traces are 14.0 ± 3.4Mbps and 267.0 ± 71.4Mbps, respectively. Thus the standard deviation is around 24% and 27% of the mean throughput which is close to the $\alpha$ value ($\sim$ 0.3, defined in §3.2.2) we observed during emulation.

• **Trace-driven Emulation and Real-world Test.** We deploy the EMP-edge instance on a server equipped with an Intel Xeon 4110 CPU clocked at 2.10GHz, an NVIDIA RTX 2080 GPU and 96GB of DDR4 RAM. The edge takes up to 390% CPU usage and 5GB memory when running with 6 vehicles. For the trace-driven emulation, We run multiple EMP-vehicle instances on another machine equipped with an Intel Xeon E5-2640 v2 CPU clocked at 2.00GHz to share the computation resources. Each vehicle uses up to 2 cores and 2GB memory, representing the often less computing power of a vehicle compared to an edge. For our real-world tests, we place a laptop (4 cores, 8GB memory) connected to LTE networks via tethering on a vehicle to run an EMP-vehicle instance.

• **Large-scale Simulation.** We perform large-scale simulations to understand the performance of REAP algorithm under scenarios having a large number of vehicles. To mimic real-world driving scenarios, we randomly generate vehicle locations in a fixed area (120m×30m) and ensure the horizontal/vertical distance between any two vehicles is greater than 3m/6m. We simulate the network transfer of each vehicle based on the size of to-be-uploaded data which is measured after applying REAP partitioning and Draco [14] compression algorithms to the point cloud data, with the replay of our real network uplink bandwidth traces.

## 5.2 End-to-end Performance

EMP is able to provide real-time enhanced perception under various processing workload and network conditions. We first compare the end-to-end performance and scalability of EMP with V2V. Note that the bandwidth between vehicles will be constrained when a vehicle simultaneously shares its data to multiple vehicles, which we also emulate using `tc`.

Figure 11 shows the end-to-end latency of the four schemes when the number of vehicles in the system varies from 2 to 6[4]. End-to-end latency here is defined as the duration between when a vehicle starts preparing the collected sensor data and when the edge or a receiver vehicle finishes processing (*e.g.,* decoding, merging) data from all vehicles, which means the data is ready for perception. This is the additional latency introduced by EMP or the V2V counterpart and all the following steps such as perception need to be performed either on the edge or on a CAV. From the figure, we can find that EMP performs the best among all schemes and EMP-Naïve performs slightly worse because vehicles are dealing with full frames, which increases overhead for encoding/decoding and uploading. V2V-Pro benefits from the partitioning algorithm. However, as the number of vehicles increases, the latency of either vehicle-to-vehicle schemes (V2V-Naïve and V2V-Pro) skyrockets while the EMP latency stays around 100ms, saving 49–65%. EMP also outperforms EMP-Naïve by 36%–43%, which highlights the advantage of REAP partitioning. Next, we show the distribution of EMP latency in Figure 12. The vast majority of frames experience less than 100ms latency, the

[3]The partitioning for V2V-Pro is only based on vehicles' relative locations as bandwidth awareness cannot be achieved without an edge.

[4]Based on the average traffic and vehicle speed in city areas in the U.S [3, 9], the average vehicle density is 0.06 /m (6 vehicles in a 100m long road)

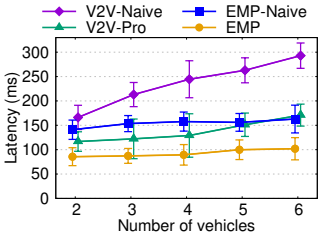



**Figure 11: End-to-end latency of EMP/V2V systems.**

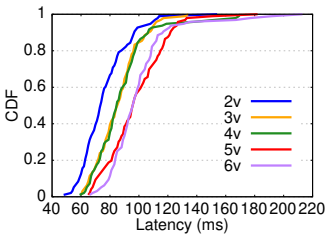



**Figure 12: Latency distribution of EMP.**

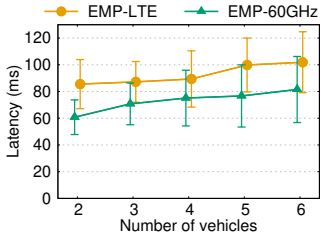



**Figure 13: Latency of EMP under different networks.**

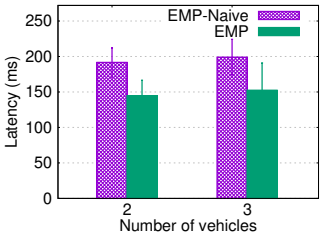



**Figure 14: Latency of EMP in real-world driving tests.**

**Table 1: Size reduction brought by REAP partitioning and the size of shared data per frame (raw point cloud size: ~2.0MB).**

| # of vehicles | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| REAP size reduction | 32.4% | 52.4% | 58.0% | 50.0% | 50.3% |
| Shared data size (KB) | 38.8 | 29.4 | 29.7 | 37.1 | 36.4 |

recommended processing delay for autonomous driving [46]. Even in the worst case (*e.g.*, due to a network blackout) a vehicle does not receive the results for some frames, it can still rely on local processing for driving decision making. EMP is designed to enhance CAVs' local processing instead of completely replacing it.

To better understand how EMP saves bandwidth, we also calculate and show the size reduction of REAP partitioning in Table 1: the average size of partitioned chunks which are shared to the edge is 49.7% of the original point cloud size across all setups (2 to 6 vehicles). Further with ground removal and point cloud compression applied, each vehicle only needs to upload 30–38KB for each frame. The data transmission can be finished within ~23ms over LTE.

EMP is robust under various network conditions. The current CAV communication technologies are mainly DSRC and C-V2X which have limited bandwidth [39]. The emergence of 5G NR and other short-range mmWave networks can provide higher bandwidth and increase the vehicular communication capability [50]. However, there could be severer fluctuations under mobility, so we evaluate EMP under LTE and 60GHz (also used in [56]) networks and plot the results in Figure 13. EMP performs better under 60GHz networks as the high bandwidth helps reduce uploading times and the adaptation mechanism still maintains the system robustness under bandwidth variability. Note the bandwidth standard deviation of the LTE and 60GHz traces are 3Mbps and 71Mbps, respectively.

Lastly, we conduct real-world driving tests with EMP. Figure 14 shows the end-to-end latency of EMP and EMP-Naïve under 2/3-vehicle scenarios. As the vehicles only need to share the data once to the edge instead of multiple times to different receiver vehicles, the latency does not inflate when increasing the number of vehicles. REAP helps reduce the processing delay by reducing the uploaded data size so that EMP outperforms EMP-Naïve. We notice that the latency under real networks is higher than that measured in the emulation. This is likely because we are using commercial cellular networks (56ms of average RTT as measured) instead of directly communicating between vehicles and a real edge node (<10ms).

## 5.3 Perception Enhancement

EMP can enhance CAVs' local perception while reducing bandwidth consumption to achieve real-time processing. We examine how EMP improves the perception of autonomous driving, by comparing

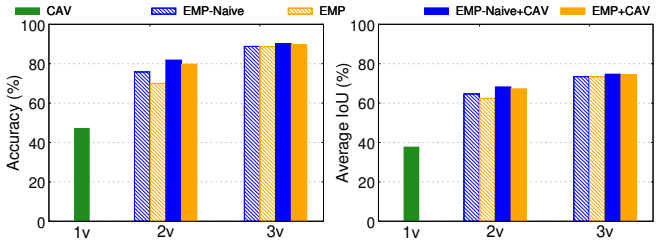



**Figure 15: Detection Accuracy (left, IoU threshold = 0.5) and Average IoU (right) of single-CAV perception, multi-CAV perception, and combined perception.**

the detection accuracy of single-CAV perception on one vehicle's point clouds (CAV), multi-CAV perception on views merged from 2/3 vehicles' data (EMP), and combined perception (EMP+CAV). The point clouds are merged in two ways: merge from full frames (EMP-Naïve) and merge from partitioned frames using REAP.

To measure the detection accuracy, we calculate the *Intersection over Union* (IoU) between the detection results (locations and dimensions of detected object bounding boxes) and the ground truth, which ranges from 0 to 1. A detection is *true* if its IoU with the label is higher than a threshold (0.5, as widely used in the computer vision community). Then we calculate the ratio of true detections out of all ground truth objects. We also directly calculate the *average IoU* for the detected objects, in order to evaluate in a more fine-grained manner. When combining results from the edge and the CAV, if two detections match the same object, we record the one with a higher IoU. Besides, the locations of vehicles in the system are known (IoU = 1) as they are reported to the edge together with point cloud data. We focus on the $80m \times 50m$ area in front of vehicles (front view).

Figure 15 plots the detection accuracy and average IoU for each setup. We find that EMP perception outperforms CAV perception under both metrics. With EMP, the detection accuracy of combined results from edge's and CAV's is even higher. Comparing the performance of detection on views merged from full frames and those on REAP-generated frames, the accuracy is reduced by 0.1% – 2.2%, for 2-vehicle and 3-vehicle results. This indicates that REAP only introduces a negligible perception accuracy loss while bringing significant bandwidth saving as shown in §5.2. Therefore, we can conclude that EMP successfully enhances CAVs' local perceptions.

We also study the impact of ground removal on perception, by running object detection on the point clouds with ground points (original data). Compared with the detection accuracy shown in Figure 15 (left), the accuracy differs by -2.58% – 1.26% (not shown in the figure). Hence, the perception will not be affected by ground removal while sometimes it can even be slightly improved, possibly

due to the reduction of noise from the ground. We run the same emulation on EMP without ground removal and the results show that EMP creates 27% – 33% less end-to-end latency.

It is worth noting that the dataset and the object detection model have limitations that may negatively affect the results. First, the vehicle sensors are not fully synchronized. As a result, the object locations in the frames of two vehicles can be slightly different and thus the detected object will have a location offset, lowering the IoU. The issue can be caused by the speed difference between vehicles and the movement of the object itself. To mitigate it, we use data collected by stationary vehicles while other objects can still move. Second, the detection model is trained with single-vehicle data[5] whose patterns are different from merged data, so the model may not perform perfectly on multi-vehicle data. Hence, the multi-vehicle perception is expected to perform better without these issues. Fully solving them can be non-trivial and we leave it for future work.

## 5.4 Case Study: Road Hazards Avoidance

Autonomous driving can benefit from EMP which provides vehicles with more knowledge of road traffic and more time to make decisions. EMP's design of sharing raw sensor data performs better than sharing processed data. To showcase such benefits, we conduct case studies and assess how EMP avoids potential road hazards in different scenarios. We customize vehicle locations in GTA V to construct the three scenarios mentioned earlier in Figure 1. Due to the limited performance of existing 3D object detection frameworks on pedestrians and cyclists [34], we simplify the scenarios to only involve vehicles and the benefits can still be shown. We measure in which frame the vehicles can detect the hazards with EMP at the earliest versus in a single-CAV setup.

**Blind Spots.** In this scenario (Figure 16 (a, b)), from the view of the ego-vehicle (the vehicle with a first-person view), a sedan is blocked by a big delivery truck behind it. The sedan is changing to the center lane. As illustrated in the camera images, without EMP, the ego-vehicle cannot detect the sedan until Frame X+8 due to occlusion. However, with EMP, the ego-vehicle can detect the sedan in Frame X, 0.8s earlier than the single-CAV setup.

**Unprotected Left Turn.** As shown in Figure 16 (c, d), the ego-vehicle is trying to make a left turn, and has to judge on its own whether there are vehicles going straight from the opposite direction. An SUV is blocking the ego-vehicle view of a sedan behind the SUV. With EMP, the ego-vehicle can detect the sedan in Frame Y instead of Frame Y+12 at which point it may have already started turning. Figure 17 visualizes the point cloud of the ego-vehicle (CAV) for Frame Y, the point cloud merged from two full frames (EMP-Naïve), and the point cloud merged from partitioned frames (EMP). The sedan is successfully detected in the last two setups.

We also analyze how EMP can avoid the potential collision in both scenarios. As shown in Table 2, the initial distances between the target vehicle and the ego-vehicle are 28m and 25m, respectively. In city areas, vehicles drive on average at 9.2m/s [9] and the braking distance is around 20m correspondingly [2]. For a single CAV, there will be a 0.8s/1.2s delay from frame differences. Together with the earlier experimental data on processing latency, we derive the remaining distance when the ego-vehicle detects the target vehicle. We can learn from the results in Table 2 that, without EMP, the CAV only has a distance of less than 20m when it is aware of the occluded vehicle, while with EMP, the CAV system or the drivers have enough time to react. We also evaluate EMP cloud mode and calculate the remaining distance, and it is still above 20m.

**Table 2: Distance when detecting the target vehicle (m).**

| Scenarios | Init. distance | EMP (EMP-cloud) | CAV |
|---|---|---|---|
| Blind spots | 28 | 26.3 (25.4) | 19.8 |
| Unp. left turn | 25 | 23.5 (22.5) | 13.2 |

**Distant Broken Down Vehicle.** In this case study (Figure 1c), we show that sharing processed data can fail to detect the distant vehicle earlier. Three vehicles are approaching a car broken down in the middle of the road. With EMP, the broken down vehicle is detected in Frame Z, while the earliest frames where each single vehicle (left, middle, and right) can detect the broken down vehicle are Z+10, Z+6, and Z+14, respectively. That means, by combining their processed data on the edge the detection success frame is Frame Z+6. Further taking into account the processing latency, sharing raw sensor data with EMP can detect the hazards 0.5s earlier than sharing processed data. Additionally, we repeat the test over EMP-Naïve in which the vehicles share full frames and the broken down vehicle is detected 7 frames earlier. Without partitioning, all data were uploaded to the edge so the details of the object build up even faster as the three vehicles approach the distant one, showing a trade-off between transmission overhead perception performance.

## 5.5 Overhead Breakdown

We break down the latency of each system component and compare them in EMP and EMP-Naïve (no REAP partitioning and scheduling) to highlight the benefit of our design decisions. We then show the throughput (FPS) improvement brought by EMP edge-side parallelization and pipelining.

Figure 18 presents the overhead of each component. Thanks to the REAP partitioning and scheduling, the uploading can be finished before all the vehicles share their full frames, as soon as the edge receives enough chunks to build the global view. Thus, the encoding, uploading, and decoding times of EMP are significantly reduced, compared to those of EMP-Naïve. There is also a small saving on the merging time. The saving on these components is much more than the additional latency introduced by REAP partitioning (6.57 ms). Besides, we also measure the overhead of inference which is the time to run 3D object detection on merged frames generated from both system schemes (Figure 19). Corresponding to the preliminary results discussed in §2.2, inference overhead increases as the number of vehicles increases. This is because more vehicles lead to a larger amount of points in the merged global view. EMP saves 21-33 ms for this step.

As mentioned in §3.5, we optimize the system workflow to increase throughput of EMP. Figure 20 illustrates the pipeline with the average latency of each part. The throughput is determined by the vehicle side processing which takes the longest time (41.39 ms), which means the system can process at 24 frames per second. Note that we directly apply Draco [14] for point cloud compression. More advanced compression approaches [44] can further reduce the vehicle-side latency.

[5]To our knowledge, no existing work on point cloud-based object detection has investigated training using views merged from multiple point clouds.

(a) Scenario 1 - Frame X (b) Scenario 1 - Frame X+8 (c) Scenario 2 - Frame Y (d) Scenario 2 - Frame Y+12

**Figure 16: Image data collected by the ego-vehicle in two scenarios where EMP detect road hazards earlier.**

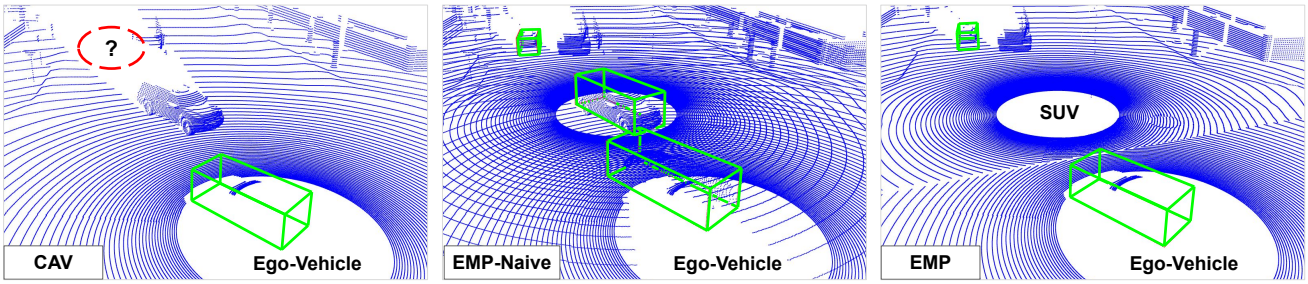



**Figure 17: LiDAR point clouds in Scenario 2. The occluded sedan can be detected in both EMP setups.**

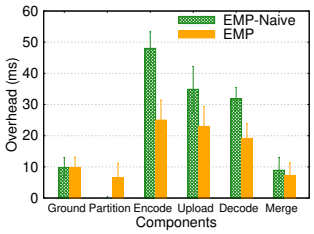



**Figure 18: Overhead of each system component.**

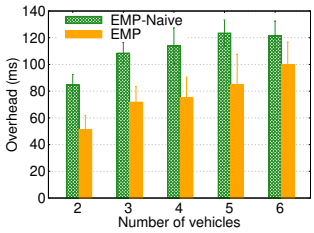



**Figure 19: Inference overhead on merged frames.**

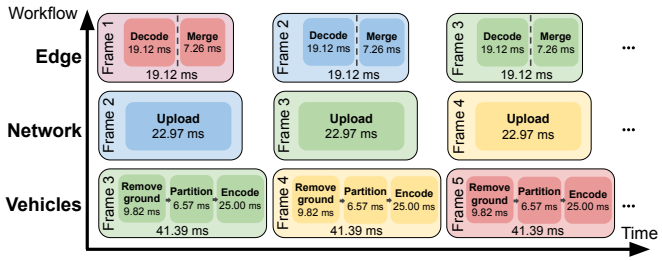



**Figure 20: Edge-side parallelization and pipelining enable EMP to process at 24 FPS.**

## 5.6 Large-scale Simulation

We next study the scalability of REAP partitioning algorithm on reducing uploaded data size and coping with network fluctuations under a large number of vehicles. We conduct large-scale simulation on the point cloud transmission with LTE uplink network traces and compare the performances of 3 settings: (1) vehicles upload full point cloud frames (Baseline); (2) vehicles partition data based on their locations following the Voronoi diagram (Voronoi); (3) vehicles partition data with REAP (REAP). In detail, we measure maximum frame uploading time ($t_{max}$) among all vehicles. For REAP, it is the time between when the first byte from any vehicle is sent and when the data from all vehicles can cover the entire area, which means it is ready for processing based on the conditions defined in §3.3. Figure 21 (a) plots $t_{max}$ averaged over 1000

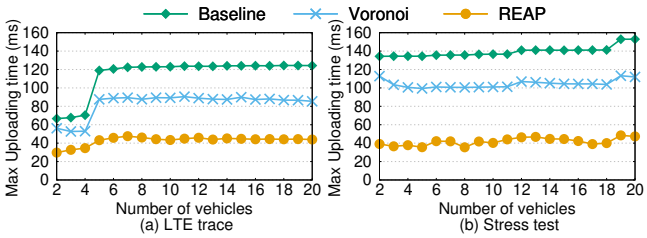



**Figure 21: Max uploading time of EMP remains stable when there are different numbers of vehicles in the system.**

runs. By removing redundant data based on vehicle locations, the Voronoi partitioning outperforms the Baseline. Further considering estimated bandwidths of vehicles and dividing the frames into several parts so that vehicles can help each other by opportunistically uploading, REAP provides the best transmission performance. Noticeably, when the number of vehicles increases from 4 to 5, $t_{max}$ increases dramatically for Baseline and Voronoi schemes. The reason for such a sharp increase is that the bandwidths in the fifth randomly selected piece of LTE traces are mostly very low, making the fifth vehicle send the slowest. However, the results of REAP remain stable, thanks to its bandwidth awareness and adaptation. Besides, the standard deviation of $t_{max}$ across different runs becomes better as we enable partitioning, and add bandwidth-aware and adaptation (Baseline: 50.30, Voronoi: 40.07, REAP: 11.68).

To evaluate the system scalability under extreme network conditions, we conduct a stress test. We randomly generate bandwidth profiles for different vehicles following normal distributions. The average bandwidths vary greatly across different vehicles (4-18Mbps) and the standard deviation is 1/4 of the mean. We also impose up to ±40% estimation errors for REAP. As shown in Figure 21 (b), the performance of different algorithms aligns with the first simulation results, proving the robustness of REAP.

# 6 RELATED WORK

**Cooperative Perception.** Various efforts have been made on cooperative vehicular perception. AVR [56] extends two vehicles'

vision by wirelessly sharing stereo camera data among each other. The See-Through System [54] streams video data directly from a leader vehicle to a follower to enhance the visibility of the follower's driver. Cooper [29] improves 3D object detection algorithms for sensor sharing but they rely on a single-vehicle dataset and simulate cooperative perception by merging data collected by the same vehicle at different timestamps. Arnold *et al.* [22] instead give attention to perception using stationary infrastructure sensors. There are other studies that target vehicle-to-vehicle communications [31, 40, 66, 72]. However, existing wireless techniques cannot support sharing of raw sensor data at a high frame rate especially as the system scales up. Prior works mostly focus on data exchange between two vehicles (*e.g.,* a leader and a follower) and do not evaluate vehicle-to-vehicle sharing at scale. EMP employs edge servers to aggregate data from vehicles, which allows each vehicle to only upload the data once instead of sharing multiple times to different peer vehicles. We further evaluate its performance with various numbers of vehicles in the system.

Beyond sharing raw sensor data, feature-level and object-level sharing approaches are explored in order to save bandwidth and reduce processing complexity. F-Cooper [28] designs a cooperative perception framework based on features extracted from point cloud data. This solution may not save much bandwidth (feature extraction may increase the data dimension and consequently the size) while sacrificing part of the original information. FusionEye [47] leverages Bipartite Graph to merge objects detected from image data. Rauch *et al.* [57] discuss sharing locally perceived object data and investigate the temporal and spatial alignment for the shared data. However, object-level sharing can fail when there are missed objects in the single-view detection since the missed ones will never appear in the combined data. In this work, to retain important sensor details while reducing bandwidth requirements, we enable efficient raw sensor data sharing by carefully partitioning the data and prioritizing different portions to be uploaded.

**Edge Computing.** Offloading heavy computational tasks from devices with limited resources such as smartphones and vehicles is a promising option to reduce on-board processing overhead. EAVVE [75] leverages edge computing to augment vision for vehicles without or with insufficient data processing capabilities. Liu *et al.* [48] propose to offload object detection tasks for mobile AR to an edge to achieve high detection accuracy and low end-to-end latency. GRACE [69] is a compression algorithm which leverages edge nodes for image inference while reducing network bandwidth consumption. Wang *et al.* [65] build a real-time video analytics system on autonomous drones involving edge-assisted processing. EMP's sensor sharing design can also be extended for other mobile entities such as drones whose fields of view can be limited, to facilitate video analytics on tasks such as search-and-rescue and wildlife protection.

**Vehicular Applications.** There are many CAV applications that can make use of on-board sensor data such as 2D images and 3D point clouds and benefit from EMP's sharing framework. For example, connected and autonomous driving systems rely on sensor data for object detection [41, 61, 73], object tracking [23, 59, 74], motion prediction [68], and path planning [45]. Sharing sensor information can provide CAVs with a broader understanding of the surroundings and eventually lead to a safer driving environment.

# 7 DISCUSSION

**Security and Privacy Considerations.** It is possible in real world that a malicious vehicle sends incorrect data misleading edge's perception [27, 63, 64]. While beyond the scope of this work, there are several possible solutions: (1) The edge distributes certificates to vehicles using protocols like P2PCD [67] at the beginning to ensure trusted communication and cross-validate the data from different vehicles. It will revoke certificates from specific vehicles upon anomaly detected. (2) Trusted execution environment (TEE) [19, 36] can be deployed at the vehicle side to prevent the attacker from sending fake data. Besides, while raw data may give less privacy, the edge can be configured to only use the shared data for enhancing perception without other purposes (*e.g.,* traffic surveillance).

**Multi-edge Support and Handover across Edges.** In this work, we mainly focus on the assistance on CAV perception from a single edge but the EMP's core design can be flexibly extended to support multiple edge nodes. In order to enable handover across edges, the serving edge (the edge that a vehicle is communicating with) needs to forward data uploaded by vehicles near the boundary of two edge nodes which can be identified from vehicles' locations. Then the serving edge signals those vehicles so that they will share future data with the target edge (the serving edge after handover).

**Adoption of the EMP System.** Many multi-agent collaboration systems [66, 76] face a cold start issue where the system cannot work (well) without enough agents to have been equipped with necessary system functions and thus new users may not be motivated to join the system. However, this issue is not critical in EMP since it is incrementally deployable. EMP does not require all vehicles to install it together. EMP cannot enhance the perception with only one vehicle, but as the number of vehicles ramps up, the effectiveness will significantly improve, as demonstrated in the experiments.

Moreover, future work can explore improvements in sensor sharing and cooperative perception. EMP's REAP algorithm partitions the sensor data based on vehicle locations and network resources. Further accounting for occlusions to adjust the region boundaries may lead to better performance. We are also planning to boost the edge-side perception by innovating the multi-vehicle data merging and the object detection framework on multi-vehicle data. PERCEIVE [43] and Lumos5G [52] provide potential solutions for optimizing the EWMA-based uplink bandwidth estimation.

# 8 CONCLUDING REMARKS

Through edge assistance and adaptive spatial partitioning, EMP makes multi-vehicle perception scalable, robust, and efficient. We believe that EMP can enable or boost a wide range of cooperative sensing applications that require multiple participating vehicles, in particular given the fast deployment of mobile networks such as 5G that offers high bandwidth and low latency. In addition to ground transportation, the underlying concept of EMP can be potentially generalized to other domains such as cooperative UAVs (drones).

# 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 Award CMMI-2038215, CMMI-2038559, CNS-1930041, CNS-1915122, and CCF-1628991.

# REFERENCES


[1] 2001. Linux Traffic Control. https://man7.org/linux/man-pages/man8/tc.8.html.

[2] 2013. Vehicle Stopping Distance and Time. https://nacto.org/docs/usdg/vehicle_stopping_distance_and_time_upenn.pdf.

[3] 2016. Annual Traffic Report - WSDOT. https://www.wsdot.wa.gov/mapsdata/travel/pdf/Annual_Traffic_Report_2016.pdf.

[4] 2018. How autonomous vehicles could save over 350K lives in the US and millions worldwide. https://zdnet.com/article/how-autonomous-vehicles-could-save-over-350k-lives-in-the-us-and-millions-worldwide/.

[5] 2019. AT&T integrating 5G with Microsoft cloud to enable next-generation solutions on the edge. https://news.microsoft.com/2019/11/26/att-integrating-5g-with-microsoft-cloud-to-enable-next-generation-solutions-on-the-edge/.

[6] 2019. Study shows autonomous vehicles can help improve traffic flow. https://phys.org/news/2018-02-autonomous-vehicles-traffic.html.

[7] 2019. Verizon and AWS announce 5G Edge computing partnership. https://techcrunch.com/2019/12/03/verizon-and-aws-announce-5g-edge-computing-partnership/.

[8] 2020. Apollo. https://apollo.auto/.

[9] 2020. INRIX Global Traffic Scorecard - Last-Mile Speed. https://inrix.com/scorecard/.

[10] 2020. Microsoft partners with the industry to unlock new 5G scenarios with Azure Edge Zones. https://azure.microsoft.com/en-us/blog/microsoft-partners-with-the-industry-to-unlock-new-5g-scenarios-with-azure-edge-zones/.

[11] 2020. Snow and Ice Pose a Vexing Obstacle for Self-Driving Cars. https://www.wired.com/story/snow-ice-pose-vexing-obstacle-self-driving-cars/.

[12] 2020. Unprotected Turns - The Right Way To Navigate Complex Intersections. https://www.epermittest.com/drivers-education/unprotected-turns.

[13] 2021. Autoware. https://www.autoware.org/.

[14] 2021. Draco 3D Graphics Compression. https://google.github.io/draco/.

[15] 2021. EMP GitHub repository. https://github.com/Shawnxm/EMP.

[16] 2021. High performance INS for ADAS and autonomous vehicle testing. https://www.oxts.com/products/rt3000/.

[17] 2021. Lidar — Wikipedia. https://en.wikipedia.org/wiki/Lidar.

[18] 2021. Netgear nighthawk x10 ad7200 smart wifi router (r9000). https://www.netgear.com/home/wifi/routers/ad7200-fastest-router/.

[19] 2021. TrustZone – Arm Developer. https://developer.arm.com/ip-products/security-ip/trustzone.

[20] 2021. Velodyne LiDAR HDL-64E. https://www.velodynelidar.com/hdl-64e.html.

[21] 2021. Verizon Hyper Precise Location. https://thingspace.verizon.com/services/hyper-precise-location/.

[22] 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).

[23] Alireza Asvadi, Pedro Girao, Paulo Peixoto, and Urbano Nunes. 2016. 3D object tracking using RGB and LIDAR data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1255–1260.

[24] Franz Aurenhammer. 1987. Power diagrams: properties, algorithms and applications. *SIAM J. Comput.* 16, 1 (1987), 78–96.

[25] Franz Aurenhammer. 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)* 23, 3 (1991), 345–405.

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

[27] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2267–2281.

[28] 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.

[29] 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.

[30] Junil Choi, Vutha Va, Nuria Gonzalez-Prelcic, Robert Daniels, Chandra R Bhat, and Robert W Heath. 2016. Millimeter-wave vehicular communication to support massive automotive sensing. *IEEE Communications Magazine* 54, 12 (2016), 160–167.

[31] Tanmoy Das, Lu Chen, Rupam Kundu, Arjun Bakshi, Prasun Sinha, Kannan Srinivasan, Gaurav Bansal, and Takayuki Shimizu. 2018. Corecast: Collision resilient broadcasting in vehicular networks. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 217–229.

[32] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 2018. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 195–205.

[33] 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.

[34] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[35] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. 2019. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5545–5554.

[36] Shengtuo Hu, Qi Alfred Chen, Jiwon Joung, Can Carlak, Yiheng Feng, Z Morley Mao, and Henry X Liu. 2020. Cvshield: Guarding sensor data in connected vehicle with trusted execution environment. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*. 1–4.

[37] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile Edge Computing—A Key Technology towards 5G. *ETSI white paper* 11, 11 (2015), 1–16.

[38] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. 2019. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2522–2529.

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

[40] 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.

[41] 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 Conference on Computer Vision and Pattern Recognition*. 12697–12705.

[42] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*. 183–196.

[43] Jinsung Lee, Sungyong Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Hyoyoung Lim, Jihoon Lee, Xiaoqing Zhu, Sangeeta Ramakrishnan, Dirk Grunwald, Kyunghan Lee, et al. 2020. PERCEIVE: deep learning-based cellular uplink prediction using real-time scheduling patterns. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 377–390.

[44] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. 2020. GROOT: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[45] Jianqiang Li, Genqiang Deng, Chengwen Luo, Qiuzhen Lin, Qiao Yan, and Zhong Ming. 2016. A hybrid path planning method in unmanned air/ground vehicle (UAV/UGV) cooperative systems. *IEEE Transactions on Vehicular Technology* 65, 12 (2016), 9585–9596.

[46] 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.

[47] 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.

[48] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[49] Martin Magnusson, Achim Lilienthal, and Tom Duckett. 2007. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics* 24, 10 (2007), 803–827.

[50] Gaurang Naik, Biplav Choudhury, and Jung-Min Park. 2019. IEEE 802.11 bd & 5G NR V2X: Evolution of radio access technologies for V2X communications. *IEEE Access* 7 (2019), 70169–70184.

[51] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. 2020. A first look at commercial 5G performance on smartphones. In *Proceedings of The Web Conference 2020*. 894–905.

[52] 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.

[53] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Z Morley Mao, et al. 2021. A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*.

[54] Cristina Olaverri-Monreal, Pedro Gomes, Ricardo Fernandes, Fausto Vieira, and Michel Ferreira. 2010. The See-Through System: A VANET-enabled assistant for overtaking maneuvers. In *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 123–128.

[55] Apostolos Papathanassiou and Alexey Khoryaev. 2017. Cellular V2X as the essential enabler of superior global connected transportation services. *IEEE 5G Tech Focus* 1, 2 (2017), 1–2.

[56] 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.

[57] Andreas Rauch, Felix Klanner, Ralph Rasshofer, and Klaus Dietmayer. 2012. Car2x-based perception in a high-level fusion architecture for cooperative perception systems. In *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 270–275.

[58] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. 2009. Fast point feature histograms (FPFH) for 3D registration. In *2009 IEEE international conference on robotics and automation*. IEEE, 3212–3217.

[59] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. 2019. PCRNet: Point cloud registration network using PointNet encoding. *arXiv preprint arXiv:1908.07906* (2019).

[60] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (2017), 30–39.

[61] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 770–779.

[62] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE internet of things journal* 3, 5 (2016), 637–646.

[63] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. 2020. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 877–894.

[64] James Tu, Tsunhsuan Wang, Jingkang Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. 2021. Adversarial Attacks On Multi-Agent Communication. *arXiv preprint arXiv:2101.06560* (2021).

[65] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. 2018. Bandwidth-efficient live video analytics for drones via edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 159–173.

[66] 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.

[67] IEEE 1609 WG. 2016. IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages. *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)* (2016).

[68] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. 2020. MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11385–11395.

[69] Xiufeng Xie and Kyu-Han Kim. 2019. Source Compression with Bounded DNN Perception Loss for IoT Edge Computer Vision. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[70] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. 2020. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 479–494.

[71] 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).

[72] Huacheng Zeng, Hossein Pirayesh, Pedram Kheirkhah Sangdeh, and Adnan Quadri. 2021. VehCom: Delay-Guaranteed Message Broadcast for Large-Scale Vehicular Networks. *IEEE Transactions on Wireless Communications* (2021).

[73] Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun. 2018. Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *IEEE Robotics and Automation Letters* 3, 4 (2018), 3434–3440.

[74] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. 2019. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2365–2374.

[75] Pengyuan Zhou, Tristan Braud, Aleksandr Zavodovski, Zhi Liu, Xianfu Chen, Pan Hui, and Jussi Kangasharju. 2020. Edge-facilitated Augmented Vision in Vehicle-to-Everything Networks. (2020).

[76] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2020. MPBond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 364–376.