



AdaMap: High-Scalable Real-Time Cooperative Perception at the Edge

Qiang Liu, Yongjie Xue, Yuru Zhang
University of Nebraska-Lincoln
qiang.liu@unl.edu

Dawei Chen, Kyungtae Han
Toyota InfoTech Labs
{dawei.chen1, kt.han}@toyota.com

ABSTRACT

Cooperative perception is the key approach to augment the perception of connected and automated vehicles (CAVs) toward safe autonomous driving. However, it is challenging to achieve real-time perception sharing for hundreds of CAVs in large-scale deployment scenarios. In this paper, we propose AdaMap, a new high-scalable real-time cooperative perception system, which achieves assured percentile end-to-end latency under time-varying network dynamics. To achieve AdaMap, we design a tightly coupled data plane and control plane. In the data plane, we design a new hybrid localization module to dynamically switch between object detection and tracking, and a novel point cloud representation module to adaptively compress and reconstruct the point cloud of detected objects. In the control plane, we design a new graph-based object selection method to un-select excessive multi-viewed point clouds of objects, and a novel approximated gradient descent algorithm to optimize the representation of point clouds. We implement AdaMap on an emulation platform, including realistic vehicle and server computation and a simulated 5G network, under a 150-CAV trace collected from the CARLA simulator. The evaluation results show that, AdaMap reduces up to 49x average transmission data size at the cost of 0.37 reconstruction loss, as compared to state-of-the-art solutions, which verifies its high scalability, adaptability, and computation efficiency.

CCS CONCEPTS

• **Networks** → **Network algorithms**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

Cooperative Perception, Point Cloud Compression, Automotive Edge Computing

ACM Reference Format:

Qiang Liu, Yongjie Xue, Yuru Zhang and Dawei Chen, Kyungtae Han. 2023. *AdaMap: High-Scalable Real-Time Cooperative Perception at the Edge*. In *The Eighth ACM/IEEE Symposium on Edge Computing (SEC '23)*, December 6–9, 2023, Wilmington, DE, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3583740.3626612>

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 the author(s) 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.

SEC '23, December 6–9, 2023, Wilmington, DE, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0123-8/23/12...\$15.00

<https://doi.org/10.1145/3583740.3626612>

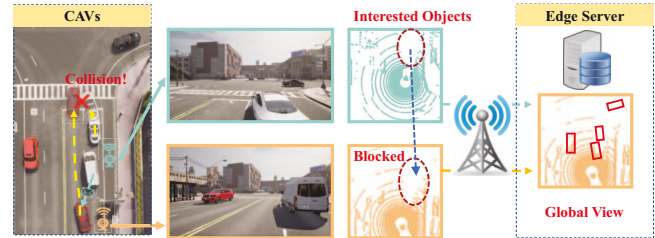


Figure 1: An illustrative example of cooperative perception.

1 INTRODUCTION

Autonomous driving and advanced driving assistance system [20, 39, 45, 48] generally control the vehicle and react to diversified roadway environments based on the local perception obtained from the perspective of the ego vehicle. To achieve robust perception, connected and automated vehicles (CAVs) [12, 37] are equipped with a variety of onboard sensors, such as cameras, LiDAR, and radar. However, these line-of-sight sensors can be contaminated under diverse roadway conditions [1], e.g., vision occlusion, extreme weather, and sensor failures, which may lead to incomplete and inaccurate information in the local perception [19, 25]. In addition, the limited view angle of the ego vehicle might confuse and even mislead the local perception. An example in [27] illustrates that a horse-drawn carriage is recognized as a truck or sedan intermittently by Tesla Autopilot. As a result, recent controversial accidents [35, 38] of pilot commercial deployment raises widespread concerns about autonomous driving techniques, in terms of safety, reliability, and availability.

Cooperative perception [9, 43, 47] has been extensively investigated to overcome the limitations of local perception in CAVs. As illustrated in Fig. 1, cooperative perception allows the sharing of perception data (e.g., point clouds, images and locations) among vehicles (i.e., V2V [13]) and infrastructures (i.e., V2I [36]), such as lampposts and edge servers. Thus, the ego vehicle can utilize the shared perception data to augment its local perception (e.g., reducing occluded areas and extending sensing ranges) for achieving high-safety driving [1, 25]. In particular, LiDAR point clouds are widely adopted in cooperative perception [9, 30], which attributes to the unique property of privacy-preserving and weather-proof. In cooperative perception, the key goal is to achieve *real time sharing* for assuring the freshness of shared information [1, 15, 34]. Given that commercial off-the-shelf LiDARs are with 10 Hz sampling rates, the end-to-end latency of perception sharing should be below 100 milliseconds, if not less. The shared perception data with unsatisfied latency provides trivial information regarding roadway environments [25], e.g., the location shift of 50mph vehicles can be more than 2 meters under 100ms delay.

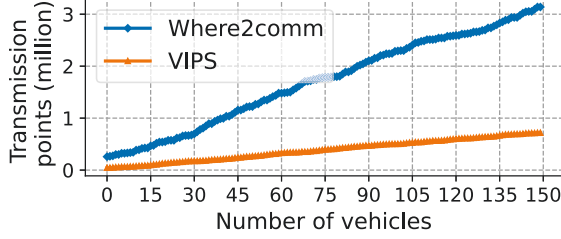


Figure 2: Number of sharing point clouds under different vehicle traffic.

It is, however, challenging to achieve real-time and large-scale cooperative perception, especially under time-varying network dynamics. For example, given the 1+ million/s point clouds of commercial 32-thread LiDARs, the required wireless data rate to support 10 CAVs could reach up to 400 Mbps [34], which would lead to unsustainable network subscription fees for service providers. Recent works [15, 31, 34] investigated transmitting only interested areas/objects (rather than the whole raw point cloud) to selective CAVs, and obtained very promising reductions in the over-the-air transmission data size. However, existing works are focused on supporting a limited number of CAVs (e.g., less than ten), which still require considerable networking resources in large-scale deployment scenarios (e.g., 100+ CAVs). According to the forecast of Automotive Edge Computing Consortium (AECC), more than 50% of cars on the road in the United States will have connected features by 2025 [11]. As a result, the cooperative perception system requires to support a massive number of CAVs, e.g., a 5G base station (1-mile radius coverage) may include hundreds of CAVs in metropolitan areas. In Fig. 2, we show the total number of points to be transmitted in existing works (i.e., VIPS [34] and Where2comm [15]) under different number of CAVs, based on our collected CARLA traces (see Sec. 6). To support the cooperative perception for 150 CAVs, Where2comm generates more than 3 millions points per frame, which needs to be transmitted over-the-air via wireless networks. Moreover, existing works cannot adapt to time-varying network dynamics in automotive edge computing (e.g., radio channel and traffic variation), which thus fail to assure consistent end-to-end latency for real-time cooperative perception [15, 34]. Therefore, it is dispensable to investigate high-scalable cooperative perception with percentile latency assurance toward future large-scale deployment scenarios.

In this paper, we propose AdaMap, a new real-time cooperative perception system, that achieves high-scalable perception sharing with assured percentile latency under time-varying network dynamics. The fundamental idea is two-fold. 1) We adaptively represent the point cloud of objects to achieve higher compression ratios by trading off reconstruction losses. This is based on the observation that, although reconstructed point clouds have perceivable reconstruction losses, they maintain sufficient structural details of point clouds regarding original objects. 2) We dynamically select objects detected by individual CAVs to reduce the total number of interested objects to be transmitted and processed. This is based on the observation that, removing duplicated multi-viewed point clouds have only trivial degradation, regarding the aggregated fidelity of objects. In AdaMap, we build an edge-based global map, which

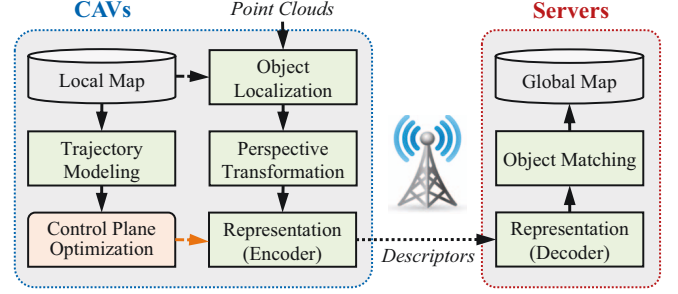


Figure 3: Overview of AdaMap.

aggregates the uploaded objects from individual CAVs, monitors and tracks aggregated objects, and broadcasts periodical updates to update the local map in all CAVs. Besides, by reusing the historical point clouds of objects, AdaMap could further reduce the transmission data size of all CAVs over time substantially. In particular, we achieve AdaMap by designing a tightly coupled data plane and control plane.

The data plane of AdaMap is to process raw point clouds in individual CAVs into sharable object descriptors (e.g., location and point cloud of vehicles). We design the data plane with two key modules. First, we design a new hybrid object localization module to adaptively switch between object detection and tracking, which reduces 26x inference time at the cost of a minimal 0.02m localization error on average. Second, we design a new flexible representation module to compress point clouds into low-dim latents based on variational autoencoder (VAE), which can achieve up to 64x compression ratio at the cost of 0.48 reconstruction loss on average.

The control plane of AdaMap is to adapt the data plane by optimizing the selection of detected objects and representation factor (RF) of selected objects in individual CAVs. To address the unique optimization problem in AdaMap, we design the control plane to be compute-efficient with three key steps. First, we decompose the problem into fully-distributed subproblems in individual CAVs, which mitigates communication overheads incurred in centralized optimization. Second, we design a new graph-based object selection method to selectively remove duplicated multi-viewed point clouds, by evaluating the metric of space density. Third, we design a new approximated gradient descent algorithm to optimize the RF of selected objects, which assures the percentile end-to-end latency under network dynamics.

We implement AdaMap on an emulation platform, which includes realistic vehicle and server computation and a simulated 5G network. We finish more than 4000 lines of codes to implement AdaMap, which is ready-to-deploy in real-world vehicles and servers. We extensively evaluate AdaMap in terms of efficacy, scalability, and adaptability, under a CARLA dataset with 150 CAVs at 100 time frames. The results show that, AdaMap reduces up to 49x over-the-air transmission data size at the cost of only 0.37 reconstruction loss, as compared to state-of-the-art solutions. Notably, AdaMap can achieve 89.4% end-to-end latency within 100ms for 150 CAVs, given only 0.2MHz wireless bandwidth in 5G networks, which sheds light on large-scale cooperative perception deployment in the future.

2 SYSTEM OVERVIEW

The goal of AdaMap is to achieve high-scalable perception sharing among CAVs with assured percentile latency under time-varying network dynamics. Besides, AdaMap needs to maintain high-fidelity objects regarding their aggregated point clouds, for further usage of autonomous driving pipelines, such as 3D scene understanding. To achieve the goal, we design AdaMap with a tightly coupled data plane and control plane in a fully distributed manner, shown in Fig. 3. 1) The object localization module localizes interested objects on the LiDAR point clouds, including 3D bounding boxes, orientations, and labels. 2) The perspective transformation module transforms detected objects from the local coordinate of CAVs into the global coordinate. 3) Then, the control plane applies to optimize the object selection and representation factor (RF) of selected objects in individual CAVs. 4) The representation (encoder) module encodes the point cloud of only selected objects into low-dim latents, according to the optimization results from the control plane. 5) An object descriptor will be built for each selected object, where CAVs will send all object descriptors to their associated edge servers. 6) The representation (decoder) module at the edge server decodes received latents and reconstructs the point cloud of objects. 7) The object matching module matches and updates received objects within historical objects in the global map. New updates of AdaMap will be periodically broadcasted to all CAVs.

3 DATA PLANE DESIGN

In this section, we describe the data plane, including object localization, point cloud representation, perspective transformation, trajectory modeling, and object matching.

3.1 Hybrid Object Localization

Object localization module is to localize and extract interested objects (e.g., pedestrians and cars) from 3D point clouds perceived by the ego vehicle. In AdaMap, we focus on only dynamic objects, e.g., moving cars and scooters, rather than stationary backgrounds (e.g., buildings and trees), which could be provided by high-definition (HD) maps. Extracting only dynamic objects will substantially reduce the total data size of point clouds to the edge server [31]. Given a point cloud, the output of object localization includes the 3D bounding boxes and the extracted point cloud of each detected object.

To localize objects in point clouds, there are generally two approaches, i.e., detection and tracking, can be used in AdaMap. On the one hand, existing deep neural network (DNN) based object detection solutions [14, 18] are generally compute-intensive, which would greatly squeeze the end-to-end latency allowance for following data plane modules. The state-of-the-art PointPillars [18] consumes 26.4 ± 4.7 ms to process point clouds in our CARLA dataset under NVIDIA RTX 3090 and Intel i7 CPU, which already occupies more than a quarter of the 100ms end-to-end latency allowance in AdaMap. On the other hand, existing object tracking solutions can achieve real time inference, while their tracking accuracy varies under different roadway situations, such as acceleration and braking (see Table 1 and Fig. 21). For example, Kalman filter [40] has been widely used for tracking in various application domains, where its inference time can be as low as sub-milliseconds.

Table 1: Comparison among object localization methods

Method	localization error (m)	inference time (ms)
Tracking	0.12 ± 0.16	0.73 ± 0.71
Detection	0.07 ± 0.04	26.41 ± 4.73
Ours	0.09 ± 0.05	1.02 ± 0.98

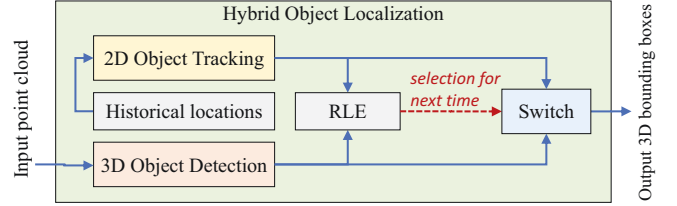


Figure 4: The hybrid object localization.

To this end, we propose a new hybrid object localization module, as shown in Fig. 4, to reduce the inference time at the cost of a minimal degradation of localization error. The basic idea is to adaptively switch between object detection and tracking, where the inference time can be considerably reduced by invoking object tracking at runtime. This is based on the observation that, object tracking can achieve considerable accurate localization in most cases in AdaMap. On the one hand, the location of objects are aggregated in the object matching module (see Sec. 3.5), which helps to correct the localization error in individual CAVs. On the other hand, the location of objects are updated frequently (e.g., maximum 100ms latency), where fine-grained historical locations help to improve the accuracy of trajectory modeling (see Sec. 3.4). However, it is challenging to determine when to switch between object tracking and detection. Because we do not have the ground-truth of object locations in real-world deployment, and thus it is difficult to evaluate the localization accuracy of object tracking.

To solve this issue, we define a new metric, named relative localization error (RLE), as the Euclidean distance between the location derived by the object tracking and detection. In other words, we consider the location generated by object detection as the ground-truth at runtime. In each time slot, we calculate the relative localization error (RLE) based on the derived locations of detection and tracking. The input of object detection and tracking are the whole raw point cloud and historical locations of previous detected objects, respectively. Then, we determine which localization method will be used, for the next time slot. For example, we select to use object tracking if the RLE is below a given threshold (empirically set as 0.5m in AdaMap). In the next time slot, the object tracking will be invoked to derive the output of 3D bounding boxes, which will hide the inference time of object detection. Otherwise, the object detection will be invoked at runtime, where its inference time will be included in the data plane of AdaMap. Note that, object detection and tracking will be performed in parallel at any time slots. For example, if we decide to use object tracking in the next time slot, the object detection will still be needed to derive the ground-truth location and calculate the RLE.

In Table. 1, we show the localization error and inference time under different object localization methods. As compared to object

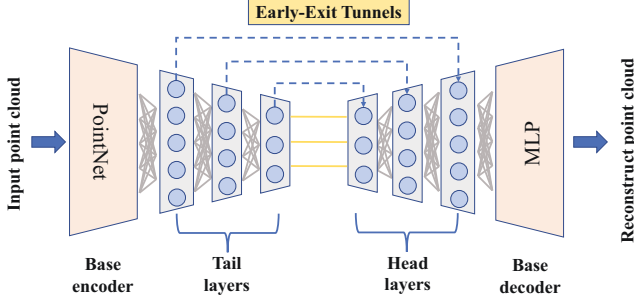


Figure 5: The representation module with early-exit tunnels.

detection (i.e., PointPillars), our method reduces 96.1% inference time at the cost of only additional 0.02m localization error. As compared to the object tracking (i.e., Kalman Filter), our method reduces 25% localization error, while the inference time is only increased 0.29ms on average. These results show that, our proposed hybrid localization methods can substantially accelerate the object localization at the cost of negligible degradation on localization losses.

3.2 Flexible Point Cloud Representation

The representation module is to represent point clouds by using low-dim vectors, where the vectors should be able to accurately recover the original point clouds afterward. In AdaMap, the information of objects (e.g., locations and point clouds) will be uploaded to edge servers and shared with all CAVs, where point clouds dominate the overall data size of object descriptors (see Sec. 3.6). Although the data size of point clouds of individual objects are relatively small (e.g., a few Kbits for 1000 points), the aggregated data size of thousands of objects (if not more) will require considerable networking resources. General-purpose compression solutions (e.g., Zstandard [10]) can be used to compress and decompress point clouds at runtime. However, the achieved compression ratios are typically limited for detected objects in AdaMap (e.g., 1.91 ± 0.06 for Zstandard), which mainly attributes to the sparse point clouds (e.g., 992 ± 1206 points in our CARLA dataset) of detected objects.

To this end, we propose a new flexible point cloud representation module based on variational autoencoder (VAE) [17]. As an unsupervised learning technique, an autoencoder generally includes an encoder that compresses the input data (e.g., point cloud) into low-dim latents with a fixed length, and a decoder that reconstructs the original data from the latents. Different from general-purpose compression solutions, autoencoders show great potentials to achieve high compression ratios under low reconstruction losses [25], for seen point clouds in the training dataset. However, the reconstruction loss might be very high for unseen point clouds (e.g., new car models), as they are not in the training dataset.

In AdaMap, we design the representation module based on VAE for the following reasons. First, we can obtain the structure model of almost all vehicles on the market and collect their point clouds in advance to be a complete training dataset. Because the majority of vehicles on the road are with specific models defined by their manufacturers. With the complete training dataset, the VAE will be trained to obtain minimal reconstruction losses, which assures the

Table 2: Reconstruction loss under different RFs

RF	4	8	16	32	64
Loss	0.26 ± 0.1	0.32 ± 0.1	0.38 ± 0.3	0.46 ± 0.3	0.48 ± 0.4

accurate reconstruction of point clouds in edge servers¹. Second, we adopt VAE rather than conventional autoencoder, where the generated latents of VAE serve as not only the low-dim representation but also the feature of point clouds in the latent space [17, 25]. By introducing a regularization term in the loss function, VAE regularizes the latent space, where similar input point clouds would generate similar latents. These unique features would be useful for further improving AdaMap, such as increasing matching accuracy via feature-aware object matching and enhancing the generalization of representation via adversarial training (see Sec. 8). However, we found that using VAE with fixed-length latents (i.e., fixed compression ratios) cannot adapt to time-varying network dynamics in AdaMap. In particular, CAVs usually experience varying radio channel qualities under high-speed mobility, where their non-stationary volatile wireless data rates would lead to fluctuating radio transmission latency.

To address this issue, we design an flexible representation module with multiple early-exit tunnels between the encoder and decoder. As shown in Fig. 5, the representation module consists of a base encoder followed by multiple tail layers, and multiple head layers followed by a base decoder. We denote an *early-exit tunnel* as the pair of a tail and a head layer with the corresponding dimension of latent space. For example, we may create $256 \times 128 \times 64$ tail layers and $64 \times 128 \times 256$ head layers, where three early-exit tunnels will be enabled, including the outermost (256 dim), middle (128 dim), and innermost (64 dim). The innermost early-exit tunnel generates latents with the lowest dimension (i.e., the highest compression ratio), and obtains the highest reconstruction loss. In contrast, the outermost early-exit tunnel obtains the lowest compression ratio and the reconstruction loss in the meantime. With this unique design, our proposed flexible representation module allows dynamic configuration of representation factor (RF) at runtime, without the need for retraining. Here, we define *representation factor (RF)* as the ratio between the dimension of input point clouds and the generated latents under individual early-exit tunnels in the representation module.

Next, we design the process for training the proposed representation module. First, we create the VAE to include only the base encoder, the outermost early-exit tunnel (e.g., 256-dim tail and head layer), and the base decoder. Second, we train the VAE extensively and freeze all of its parameters. Third, we add an additional early-exit tunnel, and train the VAE by updating only the parameters of the newly added tunnel, such as the 128-dim tail and head layer. We repeat the second and third steps to add more early-exit tunnels toward the final representation module. Note that, the representation module cannot be trained as a whole (e.g., in one training process), because it will break the early-exit mechanism, where

¹With a complete training dataset, partial-view point clouds can be represented and even completed with a high quality and accuracy [21, 44].

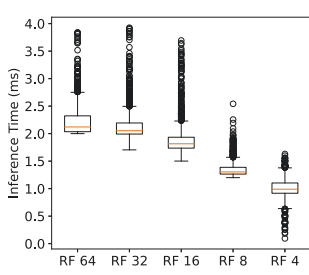


Figure 6: Inference time of representation module.

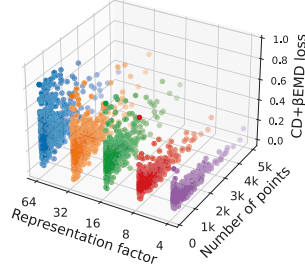


Figure 7: Reconstruction loss of representation module.

the generated latents will not be able to recover the original point clouds in the decoder, except the innermost tunnel.

In addition, we design the loss function of the representation module to include the regularization loss and the reconstruction loss. First, the regularization loss is defined as $D_{KL} [q(z|x)|p(z)]$, where x are the input point clouds and z are the sampled latents. Here, D_{KL} is the KL-divergence to evaluate the difference between the probability distribution of latents and a prior distribution (Gaussian $p(z) \sim \mathcal{N}(0, 1)$). Second, the reconstruction loss is defined as the weighted sum of Chamfer Distance (CD) [42] and EarthMover’s Distance (EMD) [41], which are the widely used metrics to evaluate the difference between point clouds. Specifically, the metric of CD and EMD are mainly focused on the local and global scale, respectively. This is based on our observation that, using merely CD or EMD will result in significant loss on global and local details in reconstructed point clouds, respectively. Hence, the reconstruction loss is expressed as

$$loss = Loss_{CD} + \beta \cdot Loss_{EMD}, \quad (1)$$

where β is the non-negative weight factor.

Table 2 shows the reconstruction loss achieved by the representation module under different RFs, where all input point clouds are re-sampled into 1024 points. We can see the tradeoff between compression ratios and reconstruction losses, where a higher RF generally leads to a higher reconstruction loss. As compared to the average 0.26 reconstruction loss under RF 4, an additional 0.22 loss can trade for 16x more compression ratio under RF 64. Fig. 6 shows the inference time of the representation module under different RFs. As more tail and head layers need to be computed for larger RFs, their inference times increase moderately. Note that, although one-time inference of the representation module is minimal, the aggregated computation time for all objects in individual CAVs is non-negligible.

3.3 Perspective Transformation

The perspective transformation module is to transform the objects in the local coordinate of individual CAVs to the global coordinate system in the edge server. We denote the 3D location of an object $P_l = [X, Y, Z, 1]$ in the local coordinate. Hence, the object’s location in the global coordinate P_g can be calculated as $P_g = P_l \times T_{l2g}$,

where T_{l2g} is the 4x4 transformation matrix defined as

$$T_{l2g} = \begin{bmatrix} c_p c_y & -c_r s_y + c_y s_p s_r & c_r c_y s_p + s_r s_y & X \\ c_p s_y & c_r c_y + s_p s_r s_y & c_r s_p s_y - c_y s_r & Y \\ -s_p & c_p s_r & c_p c_r & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

where c and s denote sine and cosine operations, and p, r, y are the pitch, roll, and yaw, respectively. At runtime, the transformation matrix could be obtained by using high-precision GPS and simultaneous localization and mapping (SLAM) techniques [26].

3.4 Trajectory Modeling

The trajectory modeling module is to model and predict the future path of dynamic objects (e.g., vehicles, scooters and pedestrians), based on their historical locations. In AdaMap, the trajectory modeling will be utilized to track all identified objects in both the object localization module (see Sec. 3.1) and object matching module (see Sec. 3.5). Although there are extensive solutions developed for trajectory modeling, e.g., deep learning-based [2, 3], they generally have high computation complexity and thus lead to extra computation latency at runtime. In AdaMap, we achieve trajectory modeling by using Kalman filter, which has proven robustness and computational efficiency under various application scenarios, such as robotics, UAVs, and aerospace.

The Kalman filter comprises two primary stages, namely the prediction stage and the correction stage. In the prediction stage, it uses the previous state estimation as the input to generate a forecast of the current state. In the correction stage, the predicted state obtained in the prediction stage is adjusted based on the current observed state. Without loss of generality, we define the state as $[X, Y, d_X, d_Y]$, where X and Y are the current location of the CAV, and Z is omitted. The d_X and d_Y are the velocity of the CAV, which are initialized as zeros.

3.5 Predictive Object Matching

The object matching module is to match multi-viewed objects received from individual CAVs, combine and update them into individual objects in the global map. In AdaMap, objects might be detected intermittently under complex roadway environments, such as miss-detection in object detection and vision blocking. Besides, all modules in the data plane might introduce noises and errors in real-world deployment [34], such as inaccurate detection and tracking, and transformation matrix error. Hence, it is non-trivial to accurately match multi-viewed dynamic objects under large-scale deployment scenarios.

To this end, we develop a predictive-based matching method by evaluating the predictive location, rather than last-observed location, of objects at runtime. Specifically, we predict the location of all objects in the global map by using the trajectory modeling module at each time frame. Denote g as the observed geo-location of an object in the world coordinate, we match it with the object $i = \arg \min_{k \in \mathcal{K}} \|g - \hat{g}_k\|^2$, where $\hat{g}_k$ is the predicted location of the k th object in the global map.

3.6 Object Descriptor

In AdaMap, we build the structure of object descriptor to include all the sharable information regarding an individual object to be transmitted. To describe an object, an object descriptor incorporates the following attributes, i.e., location, orientation, 3D bounding box, label, confidence, speed, trajectory model, latents, and certain meta-data, such as the number of raw point cloud and time stamp. The global map is mainly composed of object descriptors for all objects. For the periodical updates, only changed attributes will be included in the object descriptor and broadcasted to all CAVs.

4 CONTROL PLANE DESIGN

In this section, we describe the control plane, including system model, problem formulation, and solution design.

4.1 System Model

We consider a generic mobile network, including multiple connected and automated vehicles (CAVs), wireless base stations (BSs), and edge computing servers. As described in the data plane in Sec. 3, CAVs periodically (e.g., 10Hz) upload the descriptor of detected objects to their associated edge servers in a time-slotted manner. Edge servers will be responsible for further post-processing of object descriptors (e.g., the decoder representation module), in a first-come-first-serve manner. Note that, we design the control plane of AdaMap from the perspective of service providers, which means that network transmission generally becomes a blackbox². Without loss of generality, we consider the association between CAVs and BSs and edge servers are pre-determined, and focus on adapting the data plane towards diversified networking and computation dynamics in AdaMap.

For the i th CAV, we denote $O_i = \{o_1^i, o_2^i, \dots, o_{K_i}^i\}$ as the set of detected objects, where K_i is the number of total detected objects of the i th CAV. Denote $\mathcal{A}_i = \{a_1^i, a_2^i, \dots, a_{K_i}^i\}$ as the set of selection indicator. Note that, $a_k^i = 1$ means the k th detected object in the i th CAV is selected to be transmitted to edge servers, otherwise the object will not be transmitted. Furthermore, we denote the set of representation factor (RF) of the i th CAV as $\mathcal{R}_i = \{r_1^i, r_2^i, \dots, r_{K_i}^i\}$, where r_k^i is the RF for the k th object. The action space of RFs are constrained by the minimum $r_{\min}$ and maximum $r_{\max}$, defined by the representation module in the data plane. For example, a point cloud will go through the encoder and all the followed tail layers, if its RF is the maximum $r_{\max}$ to generate the most condensed latents.

Fidelity Model. In AdaMap, we define the fidelity of an object as the difference between its input and reconstructed point cloud. Without loss of generality, we measure the difference by mostly adopting the loss function of the representation module in the data plane. Specifically, given the discrete early-exit tunnels in the representation module, the fidelity is defined as

$$f(r|s) = -(Loss_{CD} + \beta \cdot Loss_{EMD}), \quad (3)$$

where r is the RF of the object and s is the auxiliary state. The $Loss_{CD}$ and $Loss_{EMD}$ are the loss of CD [42] and EMD [41], respectively. The rationale of introducing the auxiliary state is based

²Although network slicing techniques [24] may create a network slice with isolated resources to support AdaMap, it is impractical to assign exact radio resources to individual CAV transmissions in the time scale of subseconds.

on the following observation. In Fig. 7, we observe that the reconstruction loss of individual objects relate to not only their RFs but also the total number of their raw point clouds. Given a fixed RF, the resulted fidelity are largely distributed, but can be generally determined by the total number of raw point cloud. In other words, the averaged fidelity regardless the auxiliary state s cannot provide the effective guidance when optimizing individual objects in CAVs.

Therefore, the overall fidelity of AdaMap is defined as

$$F = \sum_{i=1}^N \sum_{k=1}^{K_i} \alpha_k^i \cdot f(r_k^i|s). \quad (4)$$

Latency Model. The end-to-end latency is defined as the elapsed time of completing the whole data plane for individual CAVs, including vehicle-side processing, radio transmission, and server-side processing. Under the complex and unpredictable network and computing dynamics, the data plane is with probabilistic transmission and computation time. In particular, we conduct an experiment to measure the computation time of the representation module under more than 10K objects. As shown in Fig. 6, the computation time of the representation module under all RFs is 1.72 ± 0.53 ms. Based on the above observation, it is difficult to derive a deterministic formulation for end-to-end latency in AdaMap.

To accurately represent the uncertainty in AdaMap, we build the latency model by incorporating multiple probabilistic parts as

$$L_i = \sum_{k=1}^{K_i} \alpha_k^i \cdot \left(\frac{c_v(r_k^i)}{R_v^i} + \frac{d(r_k^i)}{R_w^i} + \frac{c_e(r_k^i)}{R_e^i} + B_i \right), \quad (5)$$

which includes four parts, i.e., vehicle-side processing, radio transmission, server-side post-processing, and aggregated latency B_i . Here, we focus on the optimization of RF and object selection in the control plane, so the incurred latency of all other modules are aggregated, such as detection and tracking in the CAV, and object matching in the edge server. The first vehicle-side processing latency is defined as the ratio between the computation complexity of the encoder representation module $c_v(r_k^i)$ and the computation capacity of the CAV R_v^i . The second radio transmission latency is defined as the ratio between the size of generated latents $d(r_k^i)$ and the experienced wireless data rate R_w^i . Note that, we design an approximated model for this radio transmission latency. This is because, we are optimizing AdaMap from the perspective of service providers who cannot control network transmission (e.g., radio resource allocation [22, 23]) and server computation. The third server-side processing latency is defined as the ratio between the computation complexity of the decoder representation module $c_e(r_k^i)$ and the experienced computation capacity of the CAV R_e^i . Note that, 1) non-optimization variables (i.e., R_v^i, R_e^i) and static functions (i.e., $c_v(r_k^i), d(r_k^i), c_e(r_k^i)$) can be profiled under frozen implementation of the representation module; 2) the end-to-end latency is probabilistic, even under the given RF r_k^i .

4.2 Problem Formulation

In the control plane design, the objective is to maximize the expectation of overall fidelity while meeting the percentile requirement of end-to-end latency. To achieve the goal, we focus on optimizing the selection and RF of detected objects in all CAVs. Therefore, we

formulate the control plane problem as

$$\mathbb{P}_0 : \max_{\mathcal{A}_i, \mathcal{R}_i, \forall i} \mathbb{E}[F] \quad (6)$$

$$s.t. \quad Prob(L_i \leq H) \geq p, \forall i \quad (7)$$

$$r_k^i \in [r_{\min}, \dots, r_{\max}], \forall i, k \quad (8)$$

$$\alpha_k^i \in [0, 1], \forall i, k, \quad (9)$$

where $\mathbb{E}[\cdot]$ is the operator of expectation. Besides, H is the given latency requirement (e.g., 100ms) and p is the percentage threshold, such as 90%. The constraint in Eq. 7 assures that there are at least p percentage CAVs have less than H end-to-end latency.

Challenges. The challenges of solving the above problem is multi-fold. First, the problem has to be solved in a fully distributed approach. In existing centralized approaches, the optimization is completed in a central server, where non-trivial communication delay and bandwidth overheads are incurred, especially when supporting large-scale geo-distributed users. In the distributed scenario, global information (e.g., locations and wireless data rate) is generally absent when conducting the optimization in individual CAVs. Second, the problem involves multiple probabilistic parts in both the objective function and constraints, which falls into the field of stochastic programming. Due to the uncertainty in the problem, existing optimization methods (e.g., gradient descents [5]) cannot be directly applied. Third, the distributed optimization needs to be compute-efficient for runtime decisions. The transmission and computation in the data plane can only be proceeded until the optimization decision is derived. Given the stringent end-to-end latency requirement, the optimization time itself has to be minimal, such as less than 10ms.

4.3 Algorithm Design

In this section, we design a new control algorithm to effectively solve the problem $\mathbb{P}_0$ as follows. First, we decompose the problem into multiple fully-distributed subproblems, where each subproblem is focused on optimizing the selection and RF of objects in individual CAVs. The rationale of problem decomposition lies in the observation that, only two non-optimization variables (i.e., wireless data rate R_w^i and server computation capacity R_e^i) couple the problem $\mathbb{P}_0$. These subproblems will be addressed independently in corresponding CAVs at runtime. Second, we optimize the object selection to minimize the total number of selected objects without compromising the overall fidelity. The rationale behind is that, objects are generally viewed by multiple CAVs, where removing excessive point clouds will not decrease its aggregated fidelity perceivably. Third, we optimize the RF for only selected objects to meet the percentile latency constraint based on Lagrangian primal-dual method. To deal with the uncertainty in the latency and fidelity models, we design an approximated gradient descent method in solving the inner primal subproblem, inspired by the sample average approximation (SAA) [16].

4.3.1 Problem Decomposition. We observe that the problem $\mathbb{P}_0$ is mainly coupled by the experienced wireless data rate $R_w^i, \forall i$ and server computation capacity $R_e^i, \forall i$, which are not optimization variables. Given the blackbox property of radio transmission from the perspective of service providers, the future wireless data rate

needs to be predicted no matter in the server (centralized) or vehicle (distributed) sides. Besides, the first-come-first-serve edge servers can hardly be accurately managed and modeled in terms of the experienced computation capacity of individual CAVs, which generally leads to uncontrollable blackbox. Hence, we decompose the problem $\mathbb{P}_0$ into fully distributed subproblems in individual CAVs. Without loss of generality, the subproblem in the i th CAV is expressed as

$$\mathbb{P}_1 : \max_{\mathcal{A}_i, \mathcal{R}_i} \mathbb{E}[F_i] \quad (10)$$

$$s.t. \quad Prob(L_i \leq H) \geq p, \quad (11)$$

(8), (9),

where the experienced wireless data rate R_w^i and server computation capacity R_e^i will be predicted at runtime in CAVs.

4.3.2 Graph-Based Object Selection. Here, we focus on optimizing the selection of detected objects in individual CAVs. The goal is to un-select excessive multi-viewed point clouds of CAVs as many as possible, which corresponds to reducing the total number of object descriptors to be transmitted over-the-air and processed in edge servers. The challenge of object selection lies in the complex aggregated fidelity, i.e., the fidelity after post-processing (e.g., registration, fusion) of multi-viewed point clouds of CAVs. In other words, it is difficult to evaluate the impact of aggregated fidelity if any multi-viewed point clouds are removed.

To tackle this issue, we design a new graph-based selection method to selectively remove multi-viewed point clouds of objects. In particular, we introduce a metric (named *space density*), which is defined as the density of point clouds in a given 3D space. The proposed selection method iterates to remove objects, as long as the space density of objects are larger than the given threshold. The rationale behind is that, excessive point clouds on the limited surface contributes negligible improvement on the fidelity of objects. Because non-penetration LiDAR sensors can only scan the surface of objects, excessive point clouds will densely overlay on the surface without providing new structural information. In AdaMap, the multi-viewed point clouds of the identical object will be registered and stitched together, where overlaid points will be mostly combined at the end.

The proposed object selection method is achieved as follows. First, we partition the 3D bounding box of detected objects into multiple 3D sub-spaces, such as $1m^3$ cubes. Because LiDARs in CAVs can only perceive partial of an object (i.e., less than 180 degrees). For example, we can consider that two multi-viewed point clouds of an object are both on the same side. If we use the 3D bounding box of the object as a whole space, the object selection method would un-select all other multi-viewed point clouds, which will lead to imbalance space density. In AdaMap, we partition the 3D bounding box of objects into 4 sub-spaces. Second, for each 3D sub-space, we need to obtain the number of point cloud of objects for comparing with the given threshold. However, we do not have the exact numbers, because the ego CAV cannot communication with other CAVs under the fully-distributed control plane. To this end, we predict the number of point clouds by leveraging the LiDAR scanning mechanism, which generally perceives 360-degree surroundings with even distributed points.

Third, we iteratively remove multi-viewed point clouds for each sub-space of the object. We build a directional star-like graph (V, E) , where the center vertex is the current detected object. Besides, vertices V also include other potential multi-viewed CAVs, and edges E denote the predicted number of point clouds. In each iteration, we sum up the total number of point clouds of existing edges E , and compared with the given threshold of space density. The edge e with the least edge value (i.e., the number of point cloud) will be removed, if the sum of remaining edges is still larger than the threshold. The above process iterates until the threshold is met.

Note that, although the object selection is optimized in each CAV independently, all CAVs share the same global information (e.g., trajectory modeling and point cloud prediction), from the periodical updates of AdaMap. As a result, the optimization result of object selection in all CAVs would be nearly the same, even if under the fully distributed scenarios in AdaMap.

4.3.3 Approximated Gradient Descent. Here, we focus on optimizing the RF for only selected objects in individual CAVs. The challenges of RF optimization lie in 1) the percentile constraint of latency, and 2) the probabilistic fidelity and latency models.

To tackle the percentile constraint Eq. 11 in the subproblem, we design a two-layer iterative approach based on the Lagrangian primal-dual method [5]. The basic idea is to adaptively weight and incorporate the constraint into the objective function Eq. 10, which converts the constrained subproblem $\mathbb{P}_1$ into a series of unconstrained subproblems. First, we relax the discrete variables $\mathcal{R}_i$ into continuous variables denoted by $\hat{\mathcal{R}}_i$. Second, we build the Lagrangian as

$$\mathcal{G}(\hat{\mathcal{R}}_i, \lambda) = \mathbb{E}[F_i] + \lambda (Prob(L_i \leq H) - p), \quad (12)$$

where a non-negative multiplier λ is introduced to weight the constraint. Then, the subproblem $\mathbb{P}_1$ can be addressed by alternatively addressing the primal and dual subproblem. Specifically, the dual subproblem is expressed as

$$\lambda^* = \arg \max_{\lambda \geq 0} \mathcal{G}(\hat{\mathcal{R}}_i, \lambda). \quad (13)$$

In contrast, the primal subproblem is expressed as

$$\hat{\mathcal{R}}_i^* = \arg \max_{\hat{\mathcal{R}}_i \in (8)} \mathcal{G}(\hat{\mathcal{R}}_i, \lambda), \quad (14)$$

where $\hat{\mathcal{R}}_i^*$ denotes the optimal continuous RFs. In the outer layer, the dual subproblem is solved under the given RFs, by using the sub-gradient descent [5]. In the inner layer, the primal subproblem is solved under the given the multiplier λ . As the convergence of the two-layer iteration, the obtained continuous RFs will be discretized into the discrete values in $[r_{\min}, \dots, r_{\max}]$.

Due to the probabilistic parts in the above primal subproblem, its falls into the field of stochastic programming [33]. We observe that these probabilistic parts are dependent on only the representation module, which is generally offline trained before its real-world deployment in AdaMap. Hence, we can offline evaluate the representation module extensively to peek at the uncertainty of the primal subproblem. Specifically, we vary the RFs, measure the resulted probabilistic fidelity and latency, and collect the measurement datasets, as shown in Fig. 7. Given the collected datasets, we solve the primal subproblem by using gradient descent methods, which is described as follows. First, we select arbitrary initial RFs.

Algorithm 1: Approximated Gradient Descent

Input: $\hat{\mathcal{A}}_i, r_{\min}, r_{\max}$

Output: $\hat{\mathcal{R}}_i$

```

1 Offline collect measurement datasets  $\mathcal{D}$ ;
2 Initialize multiplier  $\lambda$  and start point  $\hat{\mathcal{R}}_i$ ;
3 for  $m = 1, 2, \dots$  do
4   /** solve primal subproblem ** /;
5   for  $n = 1, 2, \dots$  do
6     Deviate  $\hat{\mathcal{R}}_i$  via adding random noises  $N_o$ ;
7     Sample fidelity and latency from datasets  $\mathcal{D}$ , under
        $\hat{\mathcal{A}}_i$  and  $\hat{\mathcal{R}}_i + N_o$ ;
8     Build linear regression models for fidelity  $F_i$  and
       latency  $L_i$ ;
9     Calculate gradients based on regressed models;
10    Update  $\hat{\mathcal{R}}_i$  via one-step gradient descent;
11  /** solve dual subproblem ** /;
12  Calculate the achieved percentile latency;
13  Update multiplier  $\lambda$  via sub-gradient descent;
14 Discretize  $\hat{\mathcal{R}}_i$  to  $\mathcal{R}_i$  in  $[r_{\min}, r_{\max}]$ ;
15 return  $\mathcal{R}_i$ ;
```

Second, we deviate the current RFs and sample the corresponding probabilistic parts (i.e., fidelity F_i and latency L_i) from the measurement datasets. Third, we build an approximated local model for both fidelity and latency functions (i.e., F_i and L_i) with respect to RFs (i.e., $\mathcal{R}_i$) by using compute-efficient linear regression techniques. Fourth, we calculate the gradients based on the approximated models and use gradient descent to conduct a one-step update to the current RFs. The above process will be iterated toward its convergence. The pseudo code of the approximated gradient descent algorithm is shown in Alg. 1.

5 SYSTEM IMPLEMENTATION

We implement AdaMap on a desktop computer with Intel i7 CPU and RTX 3090 GPU under Ubuntu 20.04. The AdaMap system is implemented with mainly three parts, i.e., vehicles, the simulated network, and servers, where only object descriptors will be transmitted. Due to the intractable complexity of conducting real-world experiments for hundreds of realistic vehicles, we choose to run all the data plane on the same desktop computer. Given the limited computation capacity of the desktop computer (e.g., CPU cores and GPU memory), we execute the data plane of all CAVs sequentially and collect their vehicle-part computation latencies and transmission data sizes individually. At runtime, the simulated network takes the necessary information (e.g., vehicle locations and transmission data sizes) as the input and simulates the transmission latency of individual CAVs. Specifically, the simulated network is achieved by using a 5G system-level simulator [28], where the urban micro (UMi, Street Canyon) radio channel is adopted and the base station (BS) is located in the center of all CAVs by default³. Then, the servers continue the post-processing of the data plane

³The evaluation results of different vehicle-BS distance can be found in Fig. 18 and Fig. 19 in Sec. 6.

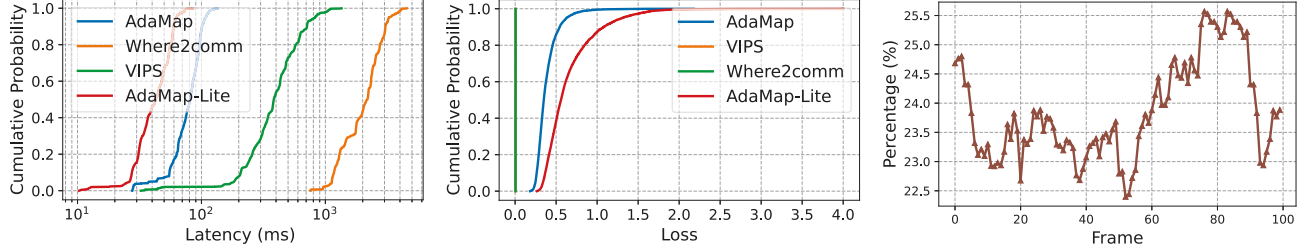


Figure 8: Cumulative probability of latency. Figure 9: Cumulative probability of losses. Figure 10: Detailed object selection under AdaMap.

accordingly. Finally, the latency of vehicle-part computation, network transmission, and server-part computation are aggregated to calculate the end-to-end latency of individual CAVs. To implement AdaMap, we have finished more 4000 lines of codes in Python. Note that, the AdaMap implementation is ready for real-world deployments, where only trivial networking configurations (e.g., pairing IP addresses) are needed to deploy on realistic cars and servers.

In the data plane, we use state-of-the-art PointPillars [18] as the 3D object detector. The threshold in our hybrid localization module is empirically tested and set to be 0.5m. To achieve the representation module, we adopt the PointNet [29] as the base encoder, and multiple fully-connected layers for the base decoder. The early-exit tunnels are implemented with 256x128x64x32 and 32x64x128x256, where all input point clouds are resampled to 1024 points. In other words, four discrete early-exit tunnels are available and thus the action space of RFs are [4, 8, 16, 32, 64]. In the loss function of the representation module, the weight $\beta = 0.0001$.

In the control plane, the percentile $p = 0.99$ and the latency threshold $H = 100ms$ for meeting the requirement of 10Hz LiDARs. We use 4 partitions for individual CAVs, where the threshold for the space density is 1024 per sub-space. This is based on our experimental measurement in CARLA, where 4096 points are generally sufficient to represent vehicles.

To collect the dataset for AdaMap evaluation, we use CARLA simulator under the *town5* with 150 vehicles and automatic navigation. We collect more than 100 time frames at 10Hz interval. The dataset includes the raw data of LiDAR and the ground-truth transformation matrix T_{l2g} for each vehicle at all time frames, at the total size of 82GB. The specification of LiDARs are 64-thread, 2.4M/s data rate, and 50m sensing range, where LiDARs are mounted on the 1 meter above the vehicle’s roof.

We compare *AdaMap* with the following state-of-the-art cooperative perception systems:

- VIPS [34]: VIPS is a state-of-the-art vehicle-to-infrastructure (V2I) solution for cooperative perception. Its basic idea is to transmit only the interested objects to the ego vehicle, without point cloud compression. For a fair comparison, we implement VIPS to use off-the-shelf compression library (i.e., Zstandard) to transmit only detected objects (rather than the whole raw point cloud) of individual vehicles.
- Where2comm [15]: Where2comm is a representative vehicle-to-vehicle (V2V) solution for extending the perception range of vehicles. Its basic idea is to share only interested areas (e.g., vision occlusion) between vehicles. For a fair comparison, we

implement Where2comm to use Zstandard compression library to transmit point clouds of the whole blind spot among individual vehicles.

- AdaMap-Lite: The AdaMap-Lite adopts the data and control plane of AdaMap, but it selects all the detected objects and uses the maximum RF at all time frames.

6 PERFORMANCE EVALUATION

In this section, we evaluate AdaMap to answer the following questions. 1) How AdaMap performs compared to state-of-the-art solutions? 2) How AdaMap scales under different roadway traffic and network conditions? 3) How AdaMap adapts to network dynamics? 4) How individual components in the data and control plane impact the overall performance of AdaMap? In particular, we evaluate AdaMap with two key metrics, including percentile end-to-end latency and average reconstruction loss. Note that, as existing solutions adopt the lossless point cloud compression library, their reconstruction losses are always zeros.

6.1 Overall Performance

We evaluate the overall latency and loss performance, which indicates the efficacy of cooperative perception systems. Fig. 8 and Fig. 9 show the cumulative probability of end-to-end latency and reconstruction loss achieved by different solutions under 150 CAVs. For the latency performance, AdaMap reduces 9.8x and 49x on average as compared to VIPS and Where2comm, respectively. Under limited networking resources (i.e., 200KHz radio bandwidth), existing solutions consume $0.42 \pm 1.5s$ and $2.12 \pm 2.4s$ to complete the perception sharing for all CAVs, which cannot be tolerated in real world deployment. In contrast, AdaMap achieves 70ms latency on average, where 89.4% CAVs have less than 100ms end-to-end latency. Note that, we evaluate AdaMap with $p = 0.99$ by default, which indicates that the percentile latency constraint in Eq. 7 is not fully satisfied here. We found the key reason is the limited number of discrete RFs in AdaMap, where the data plane supports only a total of 5 RFs. When the continuous RFs are discretized, the achieved optimality is compromised to a certain extent. In other words, this gap would be bridged by developing more fine-grained RFs in real-world deployment. For the loss performance, AdaMap achieves 0.37 reconstruction loss on average, where 95% CAVs are with less than 0.63 loss. Note that, Table 2 shows the reconstruction loss under RF 64 is 0.48 on average, where the resulted visual effects are generally acceptable for autonomous driving pipeline as illustrated in Fig. 22. These results verify that AdaMap can achieve high-scalable

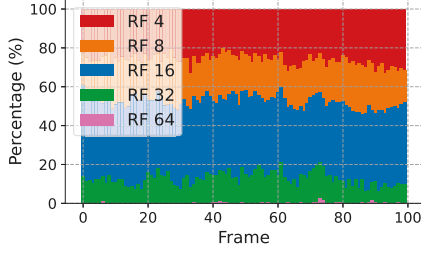


Figure 11: Detailed RF optimization under AdaMap.

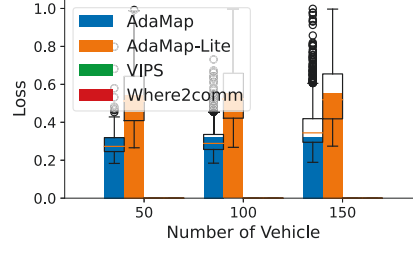


Figure 12: Loss performance under different number of CAVs.

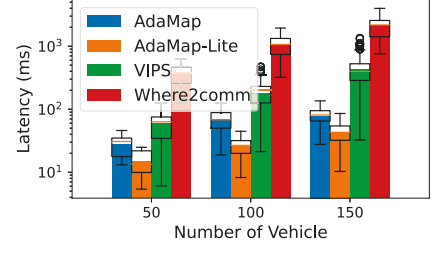


Figure 13: Latency performance under different number of CAVs.

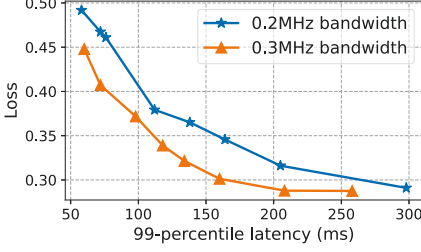


Figure 14: Pareto boundary of AdaMap.

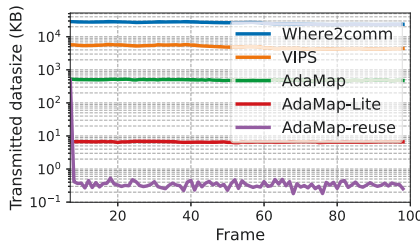


Figure 15: Transmission size under different solutions.

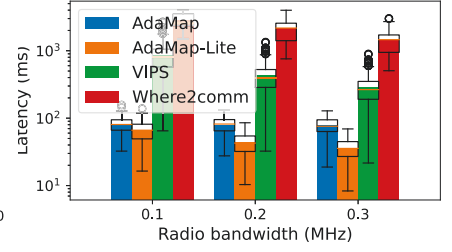


Figure 16: Latency performance under different radio bandwidth.

real-time cooperative perception by using very limited networking resources.

Besides, we dissect AdaMap in details regarding how the control plane optimizes the object selections and representation factors (RFs). Fig. 10 and Fig. 11 show the percent of selected objects and optimized RFs over 100 time frames. It can be seen that, AdaMap adaptively selects 23.6% objects on average, which indicates that a large portion of multi-viewed point clouds of objects are duplicated in terms of contributing to the overall fidelity. Hence, we observe that the transmission size of all CAVs are significantly reduced, which generally improves the latency performance under fixed networking resources. In addition, we observe that optimized RFs in AdaMap are distributed from $r_{\min} = 4$ to $r_{\max} = 64$ in Fig. 11. Generally, AdaMap would choose the RF with the highest compression ratio for trading off the latency performance under heavy traffic scenarios, such as traffic jam. However, AdaMap rarely uses RF 64 (only 0.55%) but uses RF 32 at the probability of 20.9%. This might be attributed to 1) the mechanism of evenly vehicle dispatch in CARLA simulator generally leads to non-heavy traffic scenarios; 2) the object selections in AdaMap reduce the number of objects to be transmitted before the RF optimization.

6.2 Scalability

We evaluate the scalability of AdaMap by varying the number of CAVs and percentile latency requirement. Fig. 12 and Fig. 13 show the reconstruction loss and end-to-end latency achieved by different solutions under different number of CAVs. It can be seen that, AdaMap can always maintain the percentile latency, while existing solutions experience exponential increment on their latency performances. In particular, AdaMap achieves 27.1ms latency on average when there are 50 CAVs, where the optimized RFs are mostly focused on RF 4 (98.6%). Moreover, AdaMap reduces the

average reconstruction loss by 23.8% as compared to that in high traffic (150 CAVs).

In Fig. 14, we obtain the Pareto boundary of AdaMap by varying the 99-percentile latency requirement under 150 CAVs. We observe that AdaMap can achieve the lowest 99-percentile latency of 57.7ms at the cost of 0.49 reconstruction loss, and the lowest loss of 0.29 when the latency requirement is loosen to 298ms. To push the boundary towards even lower loss and latency, one of feasible approaches is to increase available networking resources, such as radio bandwidth. In particular, the achieved 99-percentile latency of 138ms under 200KHz can be further reduced to 102ms with the same average loss of 0.36, when the radio bandwidth is expanded to 300KHz. Fig. 15 shows the total transmission size of all CAVs under different solutions in 100 frames. The transmission size is determined by not only the number of total selected objects but also the average size of each objects. In AdaMap, we transmit only selected objects, where each object is compressed by using the representation module with high compression ratios. Hence, AdaMap reduces on average 89.7% and 98.12% transmission size as compared to VIPS and Where2comm, respectively. In particular, benefit from the object matching in AdaMap, we can track all the objects in the global map. Hence, considering the reusing of historical point cloud of objects, we observe that AdaMap-Reuse further reduces 98.1% as compared to original AdaMap. Besides, AdaMap-Reuse obtains 97.8% and 99.95% reduction on the transmission size as compared to VIPS and Where2comm, respectively. These results demonstrate that AdaMap is highly scalable to tackle large-scale traffic scenarios.

6.3 Adaptability

We evaluate the adaptability of AdaMap by varying radio bandwidth and channel qualities. Fig. 16 shows the impact of radio bandwidth on the latency of different solutions. With 100KHz more radio bandwidth, VIPS and Where2comm reduce 25.3% and 50.1% average

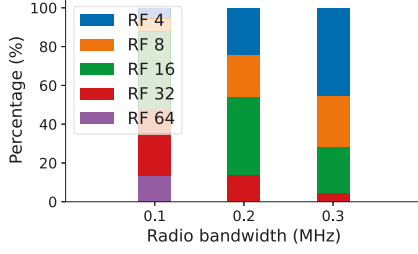


Figure 17: Detailed RF optimization under different radio bandwidth.

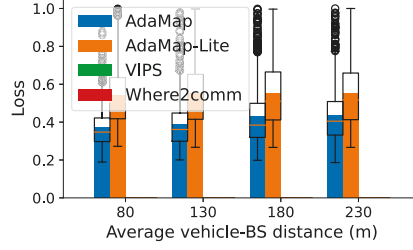


Figure 18: Loss performance under different average vehicle-BS distances.

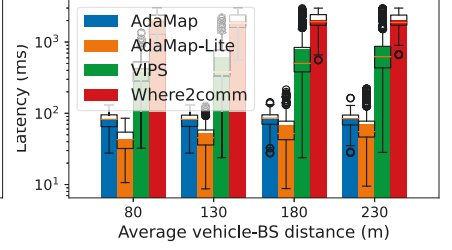


Figure 19: Latency performance under different average vehicle-BS distances.

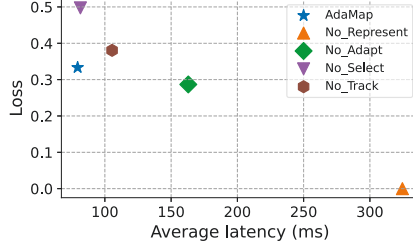


Figure 20: Impact of individual components in AdaMap.

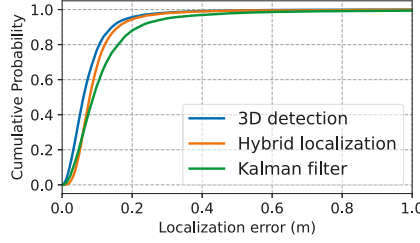


Figure 21: Localization errors under different methods.

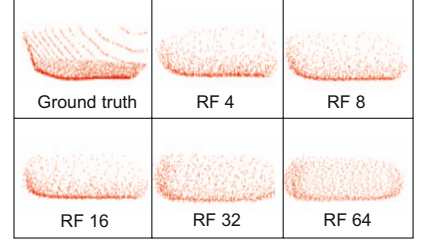


Figure 22: Example of reconstructed point clouds.

latency than that under 200KHz radio bandwidth, respectively. In contrast, AdaMap and AdaMap-Lite maintain the percentile latency by adaptively optimizing RFs, where the reconstruction loss is reduced by 23.2% on average under 100KHz more radio bandwidth. Fig. 17 shows the detailed RF optimization under different radio bandwidth. We see that, the average RF is increased from 13.5 when the bandwidth is 200KHz to 9.1 under 300KHz bandwidth. Under more radio bandwidth, AdaMap can intelligently adapt to available network resources to reduce the reconstruction loss and improve the overall fidelity.

Fig. 18 and Fig. 19 depict the impact of radio channel qualities on the performance of different solutions. Specifically, this result is collected by changing the location of the base station (BS) in the simulated network to vary the distance between vehicles and base stations. The larger vehicle-BS distance generally indicates lower achievable wireless data rates and higher fluctuations in the mean-time. We observe that, AdaMap can always satisfy the percentile latency requirements, at the cost of increasing the average reconstruction loss. These results verify that AdaMap can effectively adapt to time-varying network dynamics.

6.4 Design of Data Plane

We evaluate the data plane in terms of localization error and reconstruction loss. In Fig. 21, we show the cumulative probability of localization error, which is measured by the Euclidean distance between the predicted and ground-truth location. It can be seen that, our hybrid localization module achieves 0.19m localization error with the probability of 95%. As compared to purely object detection, our hybrid localization module obtains 96.1% reduction on the inference time at the cost of only 0.02m localization loss.

Besides, we illustrate the visual effects of the point cloud reconstructed by the representation module (see Sec. 3) under different

Table 3: Computation time of data plane modules

Modules	Inference time (ms)
Object Localization	0.061 ± 0.023
Representation	1.72 ± 0.53
Perspective Transformation	0.006 ± 0.012
Object Matching	0.014 ± 0.023

representation factors (RFs) in Fig. 22. It can be seen that, the reconstructed point cloud under RF 4 is very close to the original point cloud structurally. Although the reconstructed point cloud under RF 64 is relatively far from the original, we believe it still provides sufficient structural details for following autonomous driving pipeline, such as 3D scene understanding (more discussions in Sec. 8). Note that, we implemented the representation module with relatively small DNNs, which indicates that its reconstruction loss could be further reduced by using larger DNNs with efficient and sophisticated neural network architectures [21].

In Table 3, we show the computing time of individual data plane modules in AdaMap. With the unique design in the hybrid localization module, AdaMap only spends negligible delay on localizing interested objects. In particular, the most time-consuming module is the representation module (1.72ms on average), which involves the computation of DNNs. Note that, all these modules are very compute-efficient and can be completed in a few milliseconds, which justifies the computation efficiency of data plane in AdaMap.

6.5 Impact of Individual Components

We evaluate individual components in contributing to the overall performance of AdaMap. Overall, we observe that AdaMap achieves the best performance in terms of meeting the percentile latency while reducing the reconstruction loss. Without using the representation module in the data plane, *No_Represent* relies on existing off-the-shelf compression, which increases 4.63x average latency

with no reconstruction loss. Without using the object selection in the control plane, *No_Select* transmits all detected objects, which results in higher optimized RFs and higher reconstruction loss. Without using the hybrid detection and tracking in the data plane, *No_Track* relies on only object detection to localize interested objects, which squeezes the latency allowance for following data plane modules, and leads to worse performance on both latency and loss. Besides, *No_Adapt* uses only RF 4 all the time, which cannot adapt to network dynamics and thus fails the percentile latency requirement.

7 RELATED WORK

Cooperative perception has been increasingly studied to augment the perception for the ego vehicle towards safe autonomous driving. AVR [31] shares raw point clouds among vehicles to extend their sensing ranges and reduces the transmission size by selecting only dynamic objects and motion vectors. F-cooper [9] introduced the first feature-level fusion solution to combine both voxels and features of different LiDAR point clouds, which improves the accuracy of 3D object detection in the ego vehicle. Recently, infrastructure-assisted solutions emerge to tackle the exponential increment of transmission data size under vehicle-to-vehicle (V2V) approaches. EdgeSharing [19] leveraged edge computing servers to maintain a global map, that aggregates various crowdsourced data (e.g., raw point clouds and features), to assist the localization, detection, and matching for individual CAVs. To improve the scalability of cooperative perception, multiple works (e.g., AutoCast [30], VIPS [34], and Where2comm [15]) further exploited the spatial correlation and individual needs among CAVs to reduce the transmission data size. However, these works cannot scale in large-scale deployment with hundreds of CAVs (if not more), such as block and community scales, in terms of networking resource demand and end-to-end latency assurance.

Edge computing is expected to shape and revolutionize the future automotive industry with low-latency accessible resources, where computation offloading provides promising computation acceleration [6–8, 46]. DeepDecision [32] outlined an optimization framework to determine the offloading strategy for compute-intensive workloads, while balancing the detection accuracy, energy consumption, and network data usage. Edge-SLAM [4] is designed to split the computation modules of visual simultaneous localization and mapping (Visual-SLAM) between mobile devices and edge servers, to accelerate the overall computation by leveraging the high computation capacity at edge servers. LiveMap [25] introduced a new collaborative computing framework to intelligently and adaptively place partial computation workloads in vehicles and edge servers, during the construction of real-time dynamic map at the edge. However, existing works are mostly focused on centralized optimization with deterministic system models, which incurs significant communication delay and overhead under large-scale cooperative perception scenarios.

8 DISCUSSION

The main focus of this work lies in the fundamental trade-offs between the latency and fidelity in large-scale cooperative perception

systems. One of the key designs is the representation module for lossy compression of point clouds, which allows adaptive balancing between latent dimensions and reconstruction losses at runtime. From the visualization results in Fig. 22, we believe that our implementation of the representation module remains noticeable rooms for further improvement, in terms of reducing reconstruction losses. We envision two potential directions of improving its implementation as follows. First, the design of its loss function might be further enhanced to balance local detail and global similarity, e.g., linear searching of the weight factor β . Second, the design of its neural network architecture could be further optimized to improve the training performance, e.g., neural architecture search (NAS) and automatic hyper-parameter tuning.

In addition, one of the limitations of the representation module is the unpredictable (e.g., potentially catastrophic) reconstruction losses under unseen point clouds in the training dataset. This limitation may be alleviated by 1) building a complete training dataset as much as possible to cover diversified point clouds; 2) enhancing its generalization by adopting robust and adversarial training techniques; 3) predicting the reconstruction loss (refer to Fig. 7) and resort to lossless compression techniques if the predicted loss is too high.

Furthermore, the evaluation of this work is based on accurate local-to-global conversion matrix and non-contaminated sensory data, which might not be available in real-world deployment scenarios. The robustness of cooperative perception systems would need further extensive evaluation under real-world experiments, such as the error propagation under detection errors and localization shifts. Besides, the corresponding countermeasures demand for further exploratory study, experimental testing, and real-world validation.

9 CONCLUSION

In this paper, we presented AdaMap, a new high-scalable real-time cooperative perception system. We designed a tightly coupled data plane and control plane to achieve AdaMap. In the data plane, we designed the hybrid localization module to accelerate the object localization, and the flexible representation module to trade acceptable reconstruction losses for higher compression ratios. In the control plane, we designed the object selection method to mitigate the excessive multi-viewed point clouds, and the approximated gradient descent algorithm to online optimize the representation factor (RF) under time-varying network dynamics. We evaluated AdaMap in an emulation platform to show its scalability, adaptability and computation efficiency. The evaluation results showed that, AdaMap can assure percentile latency requirements and reduce up to 49x transmission data size at the cost of 0.37 reconstruction loss than state-of-the-art solutions. We believe that, AdaMap sheds the light on large-scale real-time cooperative perception deployment in future scenarios.

ACKNOWLEDGEMENT

This work is partially supported by the US National Science Foundation under Grant No. 2321699.

REFERENCES

- [1] Fawad Ahmad et al. 2020. Carmap: Fast 3d feature map updates for automobiles. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI})* 20. 1063–1081.
- [2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 961–971.
- [3] Florent Althé and Arnaud de La Fortelle. 2017. An LSTM network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 353–359.
- [4] Ali J Ben Ali, Marziye Kouroshli, Sofiya Semenova, Zakieh Sadat Hashemifar, Steven Y Ko, and Karthik Dantu. 2022. Edge-SLAM: Edge-assisted visual simultaneous localization and mapping. *ACM Transactions on Embedded Computing Systems* 22, 1 (2022), 1–31.
- [5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenbergh. 2004. *Convex optimization*. Cambridge university press.
- [6] Dawei Chen, Yin-Chen Liu, BaekGyu Kim, Jiang Xie, Choong Seon Hong, and Zhu Han. 2020. Edge Computing Resources Reservation in Vehicular Networks: A Meta-Learning Approach. *IEEE Transactions on Vehicular Technology* 69, 5 (2020), 5634–5646. <https://doi.org/10.1109/TVT.2020.2983445>
- [7] Dawei Chen, Haoxin Wang, and Kyungtae Han. 2023. Adaptive Delivery for High Definition Map Using A Multi-Arm Bandit Approach. In *2023 32nd Wireless and Optical Communications Conference (WOCC)*. 1–6. <https://doi.org/10.1109/WOCC58016.2023.10139782>
- [8] Juan Chen et al. 2021. A DRL Agent for Jointly Optimizing Computation Offloading and Resource Allocation in MEC. *IEEE Internet of Things Journal* (2021).
- [9] 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.
- [10] Yann Collet and Murray Kucherawy. 2018. *Zstandard Compression and the application/zstd Media Type*. Technical Report.
- [11] Automotive Edge Computing Consortium. June. 2021. *Break Down the Barriers to Automotive Edge Adoption*. White Paper. Automotive Edge Computing Consortium. <https://aecc.org/events/breaking-down-the-barriers-to-automotive-edge-adoption/>
- [12] David Elliott, Walter Keen, and Lei Miao. 2019. Recent advances in connected and automated vehicles. *journal of traffic and transportation engineering (English edition)* 6, 2 (2019), 109–131.
- [13] Ryuichi Fukatsu and Kei Sakaguchi. 2019. Millimeter-wave V2V communications with cooperative perception for automated driving. In *2019 IEEE 89th vehicular technology conference (VTC2019-Spring)*. IEEE, 1–5.
- [14] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bannamoun. 2020. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence* 43, 12 (2020), 4338–4364.
- [15] Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. 2022. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. *Advances in neural information processing systems* 35 (2022), 4874–4886.
- [16] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. 2015. A guide to sample average approximation. *Handbook of simulation optimization* (2015), 207–243.
- [17] Diederik P Kingma, Max Welling, et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392.
- [18] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12697–12705.
- [19] Luyang Liu and Marco Gruteser. 2021. EdgeSharing: Edge assisted real-time localization and object sharing in urban streets. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [20] Liangkai Liu, Sidi Lu, Ren Zhong, Baofu Wu, Yongtao Yao, Qingyang Zhang, and Weisong Shi. 2020. Computing systems for autonomous driving: State of the art and challenges. *IEEE Internet of Things Journal* 8, 8 (2020), 6469–6486.
- [21] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. 2020. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 11596–11603.
- [22] Qiang Liu, Nakjung Choi, and Tao Han. 2021. Constraint-Aware Deep Reinforcement Learning for End-to-End Resource Orchestration in Mobile Networks. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.
- [23] Qiang Liu, Nakjung Choi, and Tao Han. 2021. OnSlicing: online end-to-end network slicing with reinforcement learning. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 141–153.
- [24] Qiang Liu, Tao Han, and Ephraim Moges. 2020. EdgeSlice: Slicing wireless edge computing network with decentralized deep reinforcement learning. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 234–244.
- [25] Qiang Liu, Tao Han, Jiang Linda Xie, and BaekGyu Kim. 2021. LiveMap: Real-Time Dynamic Map in Automotive Edge Computing. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [26] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics* 31, 5 (2015), 1147–1163.
- [27] Online. 2022. Video appears to show a Tesla’s autopilot system confusing horse-drawn carriage for truck. <https://www.businessinsider.com/video-tesla-autopilot-appears-to-confuse-horse-drawn-carriage-truck-2022-8>
- [28] Edward J Oughton, Konstantinos Katsaros, Fariborz Entezami, Dritan Kaleshi, and Jon Crowcroft. 2019. An open-source techno-economic assessment framework for 5G deployment. *IEEE Access* 7 (2019), 155930–155940.
- [29] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.
- [30] Hang Qiu et al. 2022. AutoCast: scalable infrastructure-less cooperative perception for distributed collaborative driving. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications, and Services*. 128–141.
- [31] 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*. 81–95.
- [32] Xukan Ran et al. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE Conference on Computer Communications*. IEEE, 1421–1429.
- [33] Alexander Shapiro and Andy Philpott. 2007. A tutorial on stochastic programming. *Manuscript*. Available at www2.isye.gatech.edu/ashapiro/publications.html 17 (2007).
- [34] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. 2022. VIPs: Real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 133–146.
- [35] Jack Stilgoe and Jack Stilgoe. 2020. Who killed elaine herzburg? *Who’s Driving Innovation? New Technologies and the Collaborative State* (2020), 1–6.
- [36] Manabu Tsukada, Takaharu Oi, Akihide Ito, Mai Hirata, and Hiroshi Esaki. 2020. AutoC2X: Open-source software to realize V2X cooperative perception among autonomous vehicles. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. IEEE, 1–6.
- [37] Elisabeth Uhlemann. 2018. Time for autonomous vehicles to connect [connected vehicles]. *IEEE vehicular technology magazine* 13, 3 (2018), 10–13.
- [38] Bill Vlasic and Neal E Boudette. 2016. Self-driving Tesla was involved in fatal crash, US says. *New York Times* 302016 (2016).
- [39] Jiadai Wang, Jiajia Liu, and Nei Kato. 2018. Networking and communications in autonomous driving: A survey. *IEEE Communications Surveys & Tutorials* 21, 2 (2018), 1243–1274.
- [40] Greg Welch, Gary Bishop, et al. 1995. An introduction to the Kalman filter. (1995).
- [41] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. 2021. Pmp-net: Point cloud completion by learning multi-step point moving paths. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 7443–7452.
- [42] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. 2021. Balanced chamfer distance as a comprehensive metric for point cloud completion. *Advances in Neural Information Processing Systems* 34 (2021), 29088–29100.
- [43] Ruozhou Yu, Dejun Yang, and Hao Zhang. 2021. Edge-assisted collaborative perception in autonomous driving: A reflection on communication design. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 371–375.
- [44] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. 2021. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 12498–12507.
- [45] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access* 8 (2020), 58443–58469.
- [46] Jing Zhang et al. 2018. Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. *IEEE Access* 6 (2018), 19324–19337.
- [47] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2021. Emp: Edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 545–558.
- [48] Adam Ziebinski, Rafal Cupek, Damian Grzechca, and Lukas Chruszczyk. 2017. Review of advanced driver assistance systems (ADAS). In *AIP Conference Proceedings*, Vol. 1906. AIP Publishing.