VIPS: Real-Time Perception Fusion for Infrastructure-Assisted Autonomous Driving

Shuyao Shi[†], Jiahe Cui[§,†], Zhehao Jiang[†], Zhenyu Yan[†], Guoliang Xing[†,*], Jianwei Niu[§,‡], and Zhenchao Ouyang[‡]
[†]The Chinese University of Hong Kong, Hong Kong SAR, China
[§]Beihang University, Beijing, China
[‡]Beihang Hangzhou Innovation Institute, Hangzhou, China

ABSTRACT

Infrastructure-assisted autonomous driving is an emerging paradigm that expects to significantly improve the driving safety of autonomous vehicles. The key enabling technology for this vision is to fuse LiDAR results from the roadside infrastructure and the vehicle to improve the vehicle's perception in real time. In this work, we propose VIPS, a novel lightweight system that can achieve decimeter-level and real-time (up to 100 ms) perception fusion between driving vehicles and roadside infrastructure. The key idea of VIPS is to exploit highly efficient matching of graph structures that encode objects' lean representations as well as their relationships, such as locations, semantics, sizes, and spatial distribution. Moreover, by leveraging the tracked motion trajectories, VIPS can maintain the spatial and temporal consistency of the scene, which effectively mitigates the impact of asynchronous data frames and unpredictable communication/compute delays. We implement VIPS end-to-end based on a campus smart lamppost testbed. To evaluate the performance of VIPS under diverse situations, we also collect two new multi-view point cloud datasets using the smart lamppost testbed and an autonomous driving simulator, respectively. Experiment results show that VIPS can extend the vehicle's perception range by 140% within 58 ms on average, and delivers a 4× improvement in perception fusion accuracy and 47× data transmission saving over existing approaches. A video demo of VIPS based on the lamppost dataset is available at https://youtu.be/zW4oi_EWOu0.

CCS CONCEPTS

• **Computing methodologies** → **Cooperation and coordination**; **Vision for robotics**.

KEYWORDS

Perception Fusion, Vehicle Mobility, Infrastructure-Assisted Autonomous Driving, Vehicle-Infrastructure Information Fusion

*Corresponding author.

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9181-8/22/10…$15.00
https://doi.org/10.1145/3495243.3560539

ACM Reference Format:
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 *The 28th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '22), October 17–21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3495243.3560539

1 INTRODUCTION

Autonomous driving is expected to revolutionize the transportation system. However, recent pilot commercial deployments have caused widespread concerns about the reliability and safety of existing autonomous driving systems [13, 50]. In particular, many recent accidents [51, 55] are caused by the delayed or erroneous detection of victims by autonomous vehicles. The root cause of these accidents lies in the limited visual perception range of a single car, even with the most advanced vehicular sensors available today installed. It is shown in [23] that physical barriers, such as obstacle occlusions and limited sensing range in challenging weather conditions, can lead to a 50% longer reaction time or even an immediate disengagement of the autonomous driving system.

An emerging technical paradigm to address such challenges is to leverage intelligent roadside infrastructures such as lampposts equipped with sensors and compute units to improve the safety of autonomous vehicles [22, 54]. The key enabling technology of such an *infrastructure-assisted autonomous driving* paradigm is *real-time perception fusion*, where the scene perception results of infrastructures are fused into the vehicle's view in real time. As shown in Fig. 1, the sensors (e.g., LiDARs and 2D/3D cameras) installed on the roadside infrastructure can help the vehicle extend its perception range, for example, by sharing the detected objects, including the vehicle (in the green box) invisible in the vehicle's view. Such real-time perception fusion capability greatly boots the accuracy and reliability of a number of autonomous driving tasks, such as path planning [41], localization [37], and navigation [43].

In this work, we focus on fusing LiDAR results from the roadside infrastructure and the vehicle to extend the vehicle's perception in real time. Thanks to its high-resolution yet privacy-preserving 3D data, LiDAR has been widely adopted by commercial autonomous driving platforms [1, 6, 7] and intelligent roadside infrastructures [4, 9]. However, current point cloud fusion techniques [14, 17, 24, 26, 29, 32, 47] either require highly accurate locations of vehicles or the transmission of raw point clouds from the infrastructure to vehicles, which incurs significant overhead.

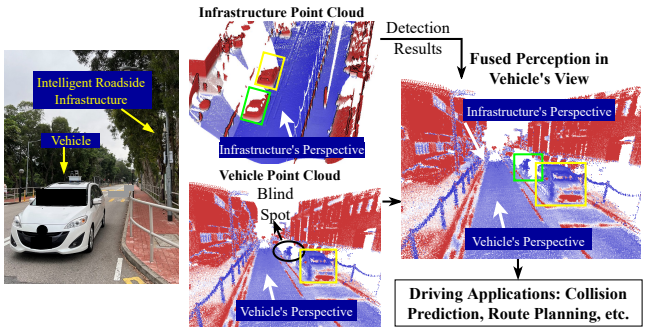


Figure 1: Infrastructure-assisted perception fusion.

Moreover, the compute-intensive nature of 3D point cloud processing is a major barrier to the deployment on existing roadside infrastructures. For instance, existing metropolitan lampposts in the US have a fixed power supply of around 8.5 kW/mile [5], of which over 80% is used for lighting, especially during the night. Retrofitting such large-scale public utilities with more power supply or deploying new infrastructure is hence extremely costly. Therefore, there still remains a significant gap between the vision of infrastructure-assisted autonomous driving and the capabilities of current perception fusion technologies based on existing roadside infrastructures.

In this paper, we propose VIPS, a new **V**ehicle-**I**nfrastructure **P**erception fu**S**ion system that accurately aligns objects detected by the vehicle and the infrastructure to expand the vehicle's perception beyond its field of view in real time. VIPS can achieve high-precision decimeter-level and real-time (up to 100 ms) perception fusion between driving vehicles and roadside infrastructure. Rather than fusing bulky raw 3D point clouds, VIPS aims to match two graphs built from lean and heterogeneous representations of the objects detected by the vehicle and the infrastructure, respectively, such as locations, semantics, sizes, and the spatial distribution of objects. Moreover, VIPS tracks the object motion trajectories to maintain the spatial and temporal consistency of the scene, which effectively mitigates the impact of asynchronous data frames and unpredictable communication/compute delays between vehicles and roadside infrastructure.

Specifically, VIPS first detects vehicles and pedestrians from the point clouds on both the infrastructure and vehicle independently. Second, it tracks the identities and motion speeds of objects, with which the vehicle rectifies the frames from the infrastructure to time-align them with the vehicle's own frames. Then, VIPS builds two multi-affinity graphs and leverages an efficient graph matching algorithm to identify the co-visible objects in the two perspectives. Lastly, VIPS aligns all the objects into the vehicle's view by finding the transformation based on the co-visible objects.

Fundamentally different from the current wisdom of LiDAR perception fusion which is based on a costly and rigid alignment of raw 3D point clouds, VIPS offers several key advantages: (i) VIPS utilizes the (possibly limited) infrastructure's compute/communication resources efficiently since only a small amount of detection results is broadcast to all passing vehicles. This leads to a highly scalable infrastructure-assisted autonomous driving architecture. (ii) VIPS leverages the spatial and semantic consistency of dynamic objects from different perspectives, which achieves accurate real-time alignment without reliance on high-precision locations of vehicles. (iii) VIPS exploits motion characteristics of the objects and thus maintains continuous representation, which enables VIPS to be robust to communication dynamics and reduces the compute requirement of both infrastructure and vehicles.

We have implemented VIPS on a real testbed consisting of a modified passenger vehicle and 16 smart lamppost nodes we deployed on a university campus. We collect two new multi-view LiDAR point cloud datasets using this testbed and a leading simulation platform for autonomous driving [25], respectively. Our results show that VIPS extends the vehicle's perception range by 140%, with an average perception fusion error of 0.6 m, which is only 25% of the results of state-of-art baselines. Moreover, VIPS only transmits object detection results at 1.2 KB per frame, which reduces the data transmission volume by about 343× and 47× compared with transmitting raw point cloud data and feature points, respectively. Lastly, VIPS achieves an end-to-end system latency within 70 ms, which enables real-time perception fusion for autonomous vehicles.

2 RELATED WORK

3D Object Detection & Tracking. 3D object detection is a core component of autonomous driving systems. With the popularity of LiDAR and stereo cameras, extensive research has focused on 3D object detection with point clouds. Extending the pioneering work, including PointNet [46] and VoxelNet [67], many solutions have been proposed to estimate and classify 3D bounding boxes of objects in point clouds [38, 49, 59, 60]. However, 3D object detection based on a single LiDAR on the vehicle often results in unsatisfactory performance due to the limited perception range and occlusions. Multi-object tracking (MOT) enables autonomous route planning and navigation by localizing detected objects in 3D space and time. Based on early works on 2D image processing [16, 56, 64], LiDAR-based tracking-by-detection [15, 28, 34, 61] has gained popularity because of its high-ranging accuracy and robustness in low-light conditions. However, most of them are based on deep learning which incurs significant computational costs. In addition, designed for a single camera/LiDAR, they fail to address the timing misalignment and inconsistent frame rates of multiple LiDARs.

Point- and Feature-Level 3D Data Fusion. Several approaches [26, 30, 35, 42] directly fuse raw point clouds from two sensors. However, such point-level data fusions require transmitting high-bandwidth raw point clouds. Several studies [20, 63] propose to transfer segmented point clouds or even point cloud patches belonging to pre-defined objects to reduce the overhead of communication. However, these methods are not applicable in vehicle-infrastructure scenarios because of the enormous difference in viewing angles and the resultant small overlap of point clouds for the same object. In addition, point-wise registration incurs excessive computational overhead on the vehicle and hence is ill-suited for real-time autonomous driving. Recently, several solutions are proposed for feature-level point cloud fusion that incurs a lower compute/communication overhead. Some solutions [12, 21, 48] use decentralized SLAM for feature point fusion. However, focused on 3D map construction

and vehicle localization, they are not designed to capture moving targets in the scene. Qi *et al.* [19] propose to detect dynamic objects cooperatively based on pre-defined shared features, which require substantial overlaps of the objects' point clouds. Moreover, feature-level data fusion assumes the same feature representation (e.g., keypoints or feature maps) from different sensors, which does not apply to infrastructure-assisted driving applications.

High-Level 3D Data Fusion. Recently, a few approaches are proposed to extract high-level results from multiple views to reduce data transmission. Some of them focus on specific tasks like risk prediction for driving [62] and vehicle re-identification [66]. However, they require both sides to run the same downstream model, which presents a major barrier for real-world deployment in infrastructure-assisted driving scenarios. In [17, 29, 54], several approaches can fuse the 3D data that has high localization accuracy, which is not practical in real-world settings. Arnold *et al.* [14] propose to share aligned and merged object detection results from multiple infrastructures based on the prior knowledge of every sensor's pose. However, since localization errors of vehicles (e.g., based on GPS and inertial sensors) can be up to several meters [37], this alignment method cannot be applied to the vehicle-infrastructure perception fusion. He *et al.* [32] propose the first vehicle-infrastructure point cloud registration system that aligns saliency points from a predefined set of objects (e.g., traffic signs and crosswalk lines) in the two point clouds. However, this method targets point-level registration and thus requires transmitting raw point clouds between vehicles and the infrastructure.

3 A MOTIVATIONAL CASE STUDY

This section presents a case study to understand the performance of current perception fusion approaches. We generate numerical results from CARLA [25], a popular autonomous driving simulator that has been used in developing industrial autonomous vehicle systems such as Apollo [6] and Autoware [7]. We use the APIs provided by CARLA to create customized road maps, traffic flow, sensor locations, vehicle speed, etc. CARLA then generates high-quality and realistic sensor data based on physics laws and sensor data (e.g., LiDAR point cloud) with Unreal Engine [8], an advanced 3D graphic rendering engine. Our testbed evaluation based on real-world datasets is presented in Section 6.

3.1 Benefits of Perception Fusion

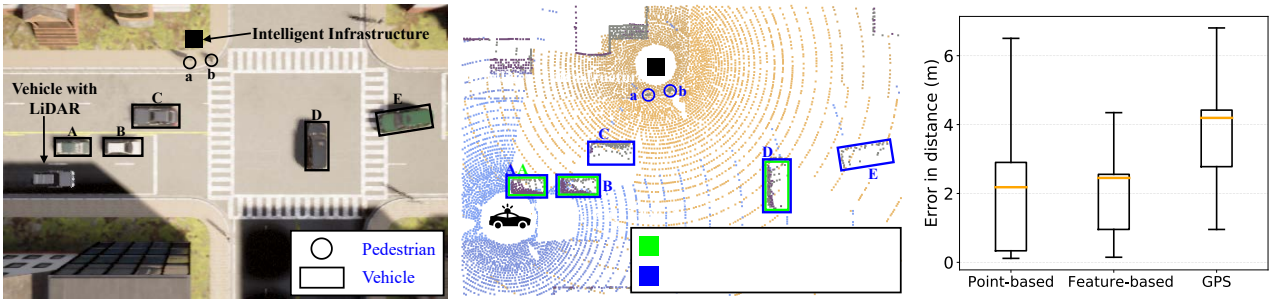
Fig. 2(a) shows an example of a crossroad simulated by CARLA. Two LiDARs are installed on the autonomous vehicle (1.6 m in height) and on a roadside infrastructure (4 m in height), respectively. We set the LiDARs with the same parameters (32 channels, 360° field of view (FOV)) used in several popular benchmark datasets, including KITTI [31] and Nuscenes [18]. We use a traffic manager to simulate traces of the vehicles and pedestrians, and record all the point cloud data continuously. Fig. 2(b) shows a bird-eye view of the point clouds from the vehicle (i.e., dots in blue) and the infrastructure (i.e., dots in brown). The point clouds are fused using the ground truth locations generated by CARLA. Blue and green boxes show the detected objects in the views of the vehicle and the infrastructure, respectively. When the two pedestrians are about to cross the road, vehicle E turns left while the vehicle ahead is still passing the intersection. Due to the occlusion, the vehicle cannot see the two pedestrians (circle a and b) and two vehicles (box C and E), which may result in a traffic accident. In contrast, the roadside infrastructure has a broader field of view and is less prone to occlusion than vehicles. Fig. 2(b) shows that almost all targets (in blue boxes) in the scene can be detected in the point cloud from the infrastructure, which complements the vehicle's field of view well. Therefore, a real-time scene perception system like this is highly desirable to prevent accidents caused by human factors like violations of traffic rules [53].

3.2 Point/Feature-level Fusion Performance

To understand the performance of current perception fusion approaches, we deploy several existing perception approaches for measurements. We first evaluate a state-of-the-art point-based registration algorithm [35] on the simulated road trace. Fig. 2(c) shows that the average fusion error is $\sim$ 2 m while the maximum error can be up to $\sim$ 6 m, where the error bars indicate the 5th and 95th percentiles. This is because the point-based registration highly relies on substantial overlapping areas between the two point clouds, and performs poorly in our evaluation due to the serious interference of the non-overlapped contents. Moreover, the end-to-end fusion pipeline would require the transmission of raw point clouds between vehicle and infrastructure, point-wise registration, and substantial subsequent processing, whose excessive compute/communication overhead poses a major challenge in meeting the stringent real-time requirement of autonomous driving applications. For instance, the Velodyne HDL32E LiDAR [2] generates point clouds at a frame rate of 10 Hz, yielding a data volume of about 40 Mbps.

As discussed in Section 2, feature-level point cloud fusion can reduce the overhead of transmitting and registering raw data clouds. However, its performance is highly susceptible to the interference of the non-overlapped areas. As illustrated in Fig. 2(b), the merged point cloud contains a very limited amount of overlapped points from the infrastructure and vehicle point clouds. Fig. 2(c) presents the performance of a state-of-the-art feature-based fusion algorithm [48]. Compared with the point-based method, the average error of perception fusion is slightly increased, and the overall errors are more evenly distributed. This is because the extraction of features can filter out some non-characteristic parts in the point cloud, resulting in more consistent performance. However, the 3 m average perception error is not applicable for the safety-critical applications in autonomous driving. Moreover, such methods lack generality because they require all agents to adopt the same feature extraction strategy.

In addition to the point- and feature-level perception fusion, there are a few recent approaches [14, 54] using object-level features from the raw data. However, they rely on the centimeter-level localization accuracy of the vehicles. We collect a real-world GPS data trace on campus with a commercial GPS and use a professional RTK-GNSS unit as ground truth. We run the object-level fusion approach [14] on the real GPS data trace and present the perception error in Fig. 2(c). The average error is around 4 m, which cannot meet the requirement of localization for autonomous driving.



(a) Bird-eye view of a sample scenario. (b) Bird-eye view of merged point clouds. (c) Perception error of existing methods.

Figure 2: A motivation study. (a) An example scenario where a vehicle drives into an intersection with the smart infrastructure. (b) Point clouds collected by a vehicle (blue dots) and the infrastructure (brown dots) and the detection in their views. (c) The distance errors of three existing perception fusion approaches.

4 SYSTEM DESIGN

4.1 System Overview

The design objective of VIPS is to achieve high-precision (i.e., sub-meter level) and real-time (up to 100 ms) perception fusion between driving vehicles and roadside infrastructure. Moreover, VIPS should be highly robust to different sensor data rates, dynamics in vehicle-infrastructure communication, and unpredictable errors in object detection and vehicle localization. As a result, the perception fusion results of VIPS can benefit various downstream tasks for autonomous driving, such as accident alarming, route planning, and vehicle localization.

To achieve these design objectives, VIPS exploits highly efficient matching of graph structures that encode objects' lean representations as well as their relationships, such as locations, semantics, sizes, and spatial distribution. As a result, VIPS enables high-precision perception fusion only using compact heterogeneous semantic and spatial features, without precise shape and surface geometry that are costly to extract, which is in sharp contrast to existing point cloud registration approaches. Moreover, by leveraging the tracked object motion trajectories, VIPS can maintain the spatial and temporal consistency of the scene, which effectively mitigates the impact of asynchronous data frames and unpredictable communication/compute delays between vehicles and roadside infrastructure.

As shown in Fig. 3, VIPS fuses the perceptions from the vehicle and infrastructure through four data processing steps, which are described as follows: 1) First, VIPS detects locations, orientations, and labels of 3D objects in the point clouds from the infrastructure and the vehicle separately. A multi-object tracking (MOT) algorithm is designed to handle missing frames caused by inconsistent frame rates and packet loss. 2) Second, VIPS rectifies the frames from the infrastructure based on their motion speeds to deal with inconsistent frame rates or missing packets. 3) Third, VIPS builds two multi-affinity graphs based on the object information (e.g., object's locations, classes, and identifiers). An efficient graph matching approach is developed to identify the matched pairs of co-visible objects. 4) Lastly, based on the object pairs, the object alignment module estimates the transformation and fuses the objects into the vehicle's view.

4.2 3D Object Detection and Tracking

We employ a 3D object detector to extract objects from the point clouds collected by LiDARs on the infrastructure and vehicle, respectively. The results are represented by labeled 3D bounding boxes, which are consistent with mainstream 3D object detection benchmarks in traffic scenarios [18, 31]. Our 3D object detector is based on Pointpillars [38], a state-of-the-art 3D object detection framework adopted by many industry-level autonomous driving platforms [6, 7]. This detector detects each input point cloud and extracts the 3D information of objects in the frame. Specifically, for each frame t, the output of 3D detector is a set of n_t detections $D_t = \{D_t^1, D_t^2, \cdots, D_t^{n_t}\}$. Each detection D_t^j, where $j \in \{1, 2, \cdots, n_t\}$, is represented as a tuple $(c, \mathbf{x}, \mathbf{b}, \theta, s)$, including class label c denoting vehicle or pedestrian, location center $\mathbf{x} = (x, y, z)$ in the 3D space, object's bounding box size $\mathbf{b} = (l, w, h)$, heading angle θ and confidence score s. Note that the infrastructure and the vehicle perform the object detection independently, so the outputs of detectors on both sides are based on their respective LiDAR coordinate systems. We denote the detection sets of infrastructure and vehicle at time t as $D_{inf,t}$ and $D_{veh,t}$, respectively.

Lightweight 3D MOT. Most state-of-the-art 3D MOT methods adopt machine learning techniques incurring excessive computational costs [15, 28], which is not applicable in autonomous driving with real-time requirements. Inspired by [58], we adopt a classical approach that integrates the Kalman filter [33] and the Hungarian method [36] to yield a satisfactory tracking performance with low latency. Fig. 4 shows the processing pipeline of the multi-object tracking. The details of the tracking procedure are described as follows: We add an object velocity vector $\mathbf{v} = (v_x, v_y, v_z)$ and a tracking ID u into the each object detection (i.e., D) to formulate the state of a tracked object (trajectory) as a tuple $T = (c, \mathbf{x}, \mathbf{b}, \theta, s, \mathbf{v}, u)$. In every frame, the state of the trajectories from the previous frame $T_{t-1} = \{T_{t-1}^1, T_{t-1}^2, \cdots, T_{t-1}^{m_{t-1}}\}$ will be propagated to frame t based on the linear motion model in the Kalman filter. The predicted trajectories T_{pred} are then associated with the detection D_t. The association is a bipartite graph matching problem, which can be efficiently solved with the Hungarian algorithm [36]. Then we can obtain the associated trajectories T_{match} and detections D_{match}, together with

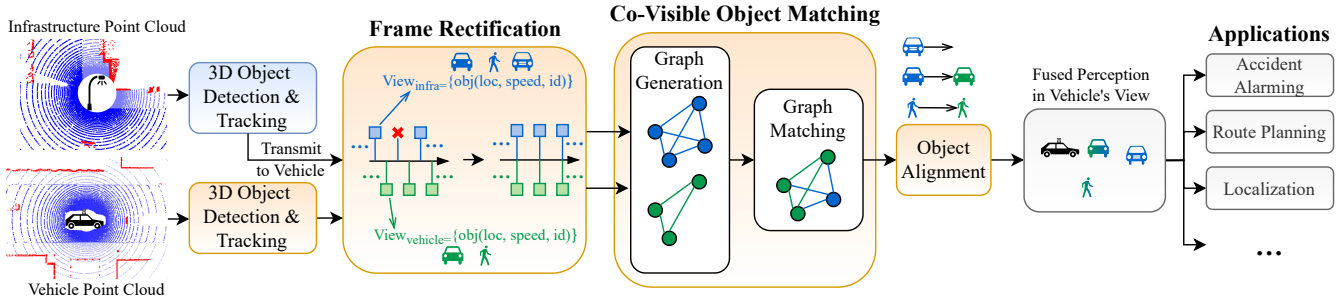


Figure 3: System architecture of VIPS. The blue and brown boxes denote operations on the infrastructure and the vehicle, respectively.

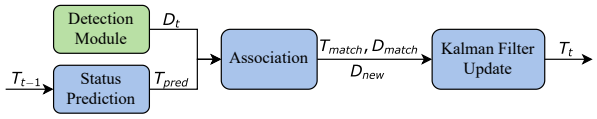


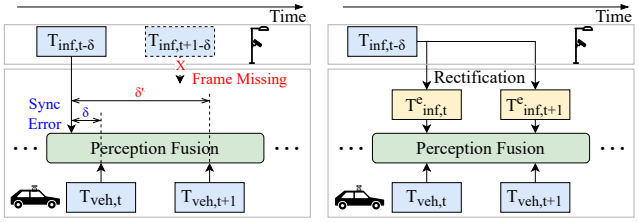
Figure 4: Multi-object tracking pipeline.

the new detections D_{new}. Lastly, we update the state of each trajectory in T_{match} based on its corresponding detection in D_{match} and add new trajectories based on new detections D_{new}. As a result, the infrastructure and the vehicle respectively obtain the trajectories they detect in frame t as $T_{inf,t} = \left\{T^1_{inf,t}, T^2_{inf,t}, \cdots, T^{m_t}_{inf,t}\right\}$ and $T_{veh,t} = \left\{T^1_{veh,t}, T^2_{veh,t}, \cdots, T^{n_t}_{veh,t}\right\}$. Once obtaining the state of detected objects (e.g., $T_{inf,t}$ at time t), the infrastructure transmits it to the vehicle.

4.3 Motion-Aware Frame Rectification

Ideally, the vehicle continuously aligns $T_{inf,t}$ and $T_{veh,t}$ that have the same timestamp t. However, as shown in Fig. 5(a), the frames from the infrastructure may not always be aligned with the vehicle frame in time. There are two major reasons: (i) *synchronization error.* The standard sampling rate of LiDARs is 10 Hz, which can cause the frames from infrastructure and vehicle to be misaligned by tens of milliseconds, shown as the left frame from the infrastructure in Fig. 5(a). (ii) *frame missing.* Due to limited computing resources, the infrastructure may perform object detection at a lower frequency than autonomous vehicles. In addition, there may be occasional packet loss in data transmission from the infrastructure to the vehicle. Therefore, the vehicle may miss some frames from the infrastructure, shown as the right frame from the infrastructure in Fig. 5(a). We address the above challenges by exploiting the motion speed of objects obtained from the MOT. Specifically, we extend the propagation of the object poses from previous frames to the current frame in the MOT to estimate their poses in future frame time. For instance, the vehicle obtains the object information in frame t as $T_{veh,t}$, and the latest data from infrastructure is $T_{inf,t-\delta}$, where δ denotes the time difference caused by the factors mentioned above. Then, we estimate $T^e_{inf,t}$ with the same linear motion model in the MOT:

$$\mathbf{x}^e_t = \mathbf{x}_{t-\delta} + \delta \cdot \mathbf{v}_{t-\delta} \quad (1)$$



(a) Sync error and frame missing. (b) The frame after rectification.

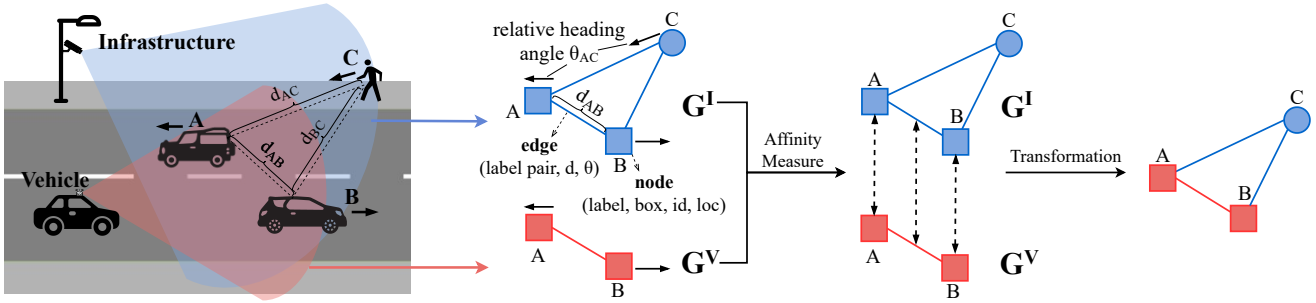
Figure 5: An illustration of frame errors and motion-aware frame rectification.

where the $\mathbf{x}_{t-\delta}$ and $\mathbf{v}_{t-\delta}$ denote the 3D location and velocity of the object in $T_{inf,t-\delta}$, respectively. Here we exploit a constant velocity model to reduce the computation and communication overhead, based on the assumption that the motion state of objects does not change drastically within milliseconds. Fig. 5(b) shows that the frame rectification can enable the vehicle to perform the perception fusion with the time-aligned frames from the infrastructure despite significant frame errors between detection results from the vehicle and infrastructure.

4.4 Co-Visible Object Matching

The fusion/alignment of two object sets (i.e., $T_{inf,t}$ and $T_{veh,t}$) relies on the objects co-visible to the vehicle and infrastructure. However, one major challenge is the object sets only contain the objects of interest (i.e., vehicles and pedestrians) represented by compact semantic and spatial information, without precise shape and surface geometry. Therefore, we design an efficient graph-based co-visible object matching approach to address the limited consensus information from two object sets, which is shown in Fig. 6. In particular, we formulate the identification of co-visible objects as a graph matching problem [39, 65]. Unlike the point cloud registration methods that require rigid point displacements, graph-based representation enables the encoding of highly heterogeneous information about objects and their relationships such as object locations, classes, sizes, and spatial structure of object distribution.

4.4.1 Graph Definition. A graph $\mathcal{G}$ is characterized by $(\mathcal{V}, \mathcal{E}, \mathcal{A})$ where $\mathcal{V}, \mathcal{E}, \mathcal{A}$ are the set of nodes, edges, and attributes, respectively. Each edge $e_{ij} \in \mathcal{E}$ that connects node v_i and v_j is assigned an attribute a_{ij}, and node attributes are denoted as a_{ii} for node v_i. We



(a) Infrastructure and vehicle detection. (b) Graph generation. (c) Graph matching. (d) Object alignment.

Figure 6: Illustration of co-visible object matching and alignment.

generate two graphs $\mathcal{G}^V = (\mathcal{V}^V, \mathcal{E}^V, \mathcal{A}^V)$ and $\mathcal{G}^I = (\mathcal{V}^I, \mathcal{E}^I, \mathcal{A}^I)$ based on object sets $T_{veh,t}$ and $T_{inf,t}$, with $|\mathcal{V}^V| = |T_{veh,t}| = n$ and $|\mathcal{V}^I| = |T_{inf,t}| = m$. The nodes and edges in each graph denote objects and their pair-wise relationships. We aim to establish an assignment $\Omega = \{(v_i, v_{i'}) \mid v_i \in \mathcal{V}^V, v_{i'} \in \mathcal{V}^I\}$ between the nodes of two graphs, so that a criterion over the matched nodes and edges is optimized. As a result, the objects represented by the matched nodes indicate the intersection of the two object sets, i.e., the co-visible objects.

Let $\mathbf{w} \in \{0,1\}^{nm\times 1}$ be an indicator vector of the assignment Ω where $\mathbf{w}_{ii'} = 1$ if $(v_i, v_{i'}) \in \Omega$ and 0 otherwise. We build a symmetric positive square matrix $\mathbf{M} \in \mathbb{R}^{nm\times nm}$, called *affinity matrix*, where $\mathbf{M}_{ii';jj'}$ measures how well the edge e_{ij} matches with edge $e_{i'j'}$. Specifically, the diagonal entries denote node-to-node affinities whereas the off-diagonal entries represent edge-to-edge affinities. The affinities are measured according to the attributes of nodes or edges (c.f., Section 4.4.2). Then, the matching problem can be solved by finding the optimal assignment $\mathbf{w}^*$ to maximize the overall affinity score as follows:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\ \mathbf{w}^\top \mathbf{M}\mathbf{w}, \quad s.t.\ \mathbf{C}\mathbf{w} \le \mathbf{1},\ \mathbf{w} \in \{0,1\}^{nm\times 1}, \tag{2}$$

where the binary matrix $\mathbf{C} \in \mathbb{R}^{nm\times nm}$ encodes one-to-one mapping constraints: $\forall i'\ \sum_i \mathbf{w}_{ii'} \le 1$ and $\forall i\ \sum_{i'} \mathbf{w}_{ii'} \le 1$.

4.4.2 Multi-Attribute Affinity Measure. The heterogeneous information in $T_{inf,t}$ and $T_{veh,t}$ is encoded into the node and edge attributes in the graphs, with which we measure the affinities between nodes and edges and build the *affinity matrix* $\mathbf{M}$.

Node-to-Node Affinity. We define the attribute a_{ii} of node v_i based on the object's class c_i, bounding box size $\mathbf{b}_i$, tracking ID u_i and location $\mathbf{x}_i^*$, which is shown in Fig. 6(b). Here we transform the object location from $\mathbf{x}_i$ which is under the LiDAR coordinate to $\mathbf{x}_i^*$ under the world coordinate based on the location of the vehicle or infrastructure. We measure multiple affinities between two nodes $v_i \in \mathcal{V}^V$ and $v_{i'} \in \mathcal{V}^I$ based on their attributes. Specifically, we measure the semantic affinity and size affinity as $f_{ii'}^{(1)} = equal(c_i, c_{i'})$ and $f_{ii'}^{(2)} = \exp(-\lambda_1 \|\mathbf{b}_i - \mathbf{b}_{i'}\|^2)$, where the *equal* function measures the similarity of the object classes. Here as we only focus on two types of objects, the *equal* function outputs 1 if the semantic labels are the same and 0 otherwise. To take advantage of the tracking information, we define $f_{ii'}^{(3)} = exist([u_i, u_{i'}])$ as the trajectory affinity, where $exist(u_i, u_{i'})$ represents whether the nodes with tracking ID u_i and $u_{i'}$ are matched in the last frame pair. In addition, we measure the distance of the two objects under the world coordinate to define the location affinity as $f_{ii'}^{(4)} = \exp(-\lambda_2 \|\mathbf{x}_i^* - \mathbf{x}_{i'}^*\|)$. λ_1 and λ_2 are scale factors, which can be flexibly adjusted according to application scenarios. Here we empirically set them to 0.5 and 0.1 as in [65]. We combine the affinities as:

$$\mathbf{M}_{ii';ii'} = f_{ii'}^{(1)} f_{ii'}^{(3)} \left[\mu_1 f_{ii'}^{(2)} + \mu_2 f_{ii'}^{(4)}\right], \tag{3}$$

which is the diagonal entry $\mathbf{M}_{ii';ii'}$ in the *affinity matrix*. Here μ_1 and μ_2 are weight factors, which are set to 0.5 to balance the weights of different affinities. Note that as the world coordinates of objects (e.g., $\mathbf{x}_{i'}^*$) are derived from GPS or the cellular network infrastructure which usually have errors up to several meters [37]. Therefore, in Eqn. 3 we linearly sum up the location affinities. As a result, when the location error is large, the value of $f^{(4)}$ diminishes thus the overall affinity is dominated by the other affinities.

Edge-to-Edge Affinity. Fig. 6(b) also illustrates the affinities between edges. Specifically, the attribute of edge e_{ij} indicates the relationship between nodes v_i and v_j, which are defined based on their semantic label pair $\mathbf{c}_{ij} = (c_i, c_j)$, distance $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$, and relative heading angle $\theta_{ij} = \theta_i - \theta_j$. By exploiting the property of the LiDAR point cloud which captures the real distance information in the scene, the distance and relative orientation between two objects from different viewpoints are consistent without perspective distortions. For two edges $e_{ij} \in \mathcal{E}^V$ and $e_{i'j'} \in \mathcal{E}^I$, we measure their semantic affinities using $g_{ii',jj'}^{(1)} = equal(\mathbf{c}_{ij}, \mathbf{c}_{i'j'})$, where the *equal* function is the same as in the node-to-node affinity measurement. The distance and relative orientation affinities are measured with $g_{ii',jj'}^{(2)} = \exp(-\lambda_3(d_{ij} - d_{i'j'})^2)$ and $g_{ii',jj'}^{(3)} = \exp(-\lambda_4 |\sin(\theta_{ij}) - \sin(\theta_{i'j'})|)$, respectively, where λ_3, λ_4 are scale factors and are set to 0.5 and 0.1 as in [65]. The edge affinities are combined in:

$$\mathbf{M}_{ii';jj'} = g_{ii',jj'}^{(1)} \left[\mu_3 g_{ii',jj'}^{(2)} + \mu_4 g_{ii',jj'}^{(3)}\right], \tag{4}$$

where we sum up $g_{ii',jj'}^{(2)}$ with $g_{ii',jj'}^{(3)}$ as the relative pose affinity between the edges, μ_3 and μ_4 are weight factors and are set to 0.5 to balance the weights of affinities.

4.4.3 Graph Matching. The graph matching problem subject to the one-to-one mapping constraint (i.e., Eqn. 2) is known to be NP-hard. Here we employ a computationally efficient method [39] where the mapping constraint is relaxed to a normalized format and the optimization problem is as follows:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\ \mathbf{w}^\top \mathbf{M}\mathbf{w}, \quad s.t.\ \|\mathbf{w}\|^2 = 1. \tag{5}$$

Since $\mathbf{M}$ is symmetric and non-negative, the problem in Eqn. 5 can be solved by computing the leading eigenvector of $\mathbf{M}$, which is denoted as $\mathbf{w}^*$. In this design, the most time-consuming operation in the graph matching process is constructing the affinity matrix $\mathbf{M}$, which has a time complexity of $O(n^2)$. In contrast, the exhaustive search approach needs to measure the matching quality for every searched matching, whose complexity is $O(n!)$. Moreover, as $\mathbf{w}^*$ is non-negative and $\|\mathbf{w}^*\|^2 = 1$, the elements of $\mathbf{w}^*$ represent the node-to-node matching scores that necessarily lie in the interval $[0, 1]$. Therefore, the matched node pairs set Ω can be determined by setting a threshold and then maximizing the overall matching score, which can be solved with the Hungarian algorithm [36]. Here we set the threshold to 0.5 to ensure that matching results have the highest score to identify co-visible objects reliably. When there are no matched nodes, which means that none of the objects detected by the vehicle and the infrastructure are co-visible, the vehicle relies only on its own scene perception without utilizing the detection results from the infrastructure.

4.5 Object Alignment

After the graph matching, we have the overlapped nodes correspondence Ω between two graphs $\mathcal{G}^V$ and $\mathcal{G}^I$ (shown in Fig. 6(c)) which can be mapped to the co-visible object sets $T^c_{inf,t} \subseteq T_{inf,t}$ and $T^c_{veh,t} \subseteq T_{veh,t}$, where $|T^c_{inf,t}| = |T^c_{veh,t}| = |\Omega|$. Note that the bounding box and spatial pose for each co-visible object in $T^c_{veh,t}$ and $T^c_{inf,t}$ are available, we can obtain the transformation matrix $\mathbf{T}$ that aligns the detection results from the infrastructure to the vehicle by registering two small point sets consisting of the bounding box corners. Since registration corresponds to a linear least-square problem, we use Singular Value Decomposition (SVD) for the alignment. In particular, instead of treating the corners of each object's bounding box equally, we assign the weights to objects based on the detection confidence and matching scores in graph matching. Thus, the transformation matrix $\mathbf{T}$ is robust to errors in object detection and co-visible object matching. Eventually, the detection of infrastructure $T_{inf,t}$ can be transformed to vehicle's view with $\mathbf{T}$ and aligned with the vehicle's own detection $T_{veh,t}$ (as shown in Fig. 6(d)).

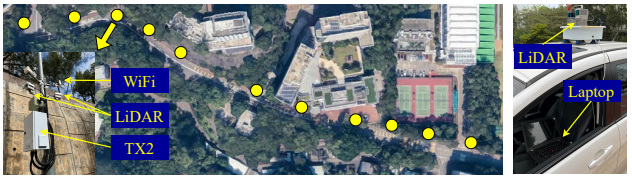
5 TESTBED AND DATASETS

We have built a real-world testbed consisting of smart lamppost nodes and a modified vehicle as shown in Fig. 7. We collect a new dataset on two roads (~ 1.75 km) with 16 self-deployed smart lampposts. Fig. 7(a) shows one of the two roads and the setup of the lamppost nodes. Each lamppost node is equipped with two LiDARs at the height of (~ 3.5 m) facing opposite directions of the road. We built two platforms to collect vehicle point clouds on the road: a small mobile cart installed with a LiDAR, and a real passenger car (see Fig. 1 and 7(c)). For both platforms, a LiDAR is mounted at about ~ 1.7 m above the ground. We use Livox Horizon LiDAR [11] on both smart lampposts and vehicles, which has an FOV of 82° and outputs 24,000 points per frame (~ 300 KB).

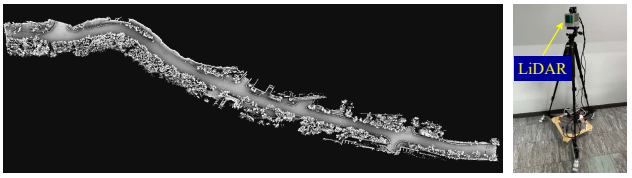
Table 1: Summary of two new datasets.

Datasets	length (km)	road types	#object per frame	ground truth	#point cloud pairs
Campus dataset	1.75	two-way	1~5	✓	1,428
CARLA dataset	5.78	single/two-way, intersection	1~15	✓	6,523

Existing LiDAR datasets for autonomous driving [18, 31] have point cloud data from the vehicle's view only, which cannot be used in infrastructure-assisted perception fusion. Therefore, we collect two new datasets, one on a university campus testbed and one generated by the CARLA simulator [25]. Both datasets contain point cloud pairs from both the vehicle and the roadside infrastructure. Table 1 summarizes the two datasets. The details are presented as follows.



(a) Campus smart lamppost testbed we installed. (c) Test vehicle.



(b) 3D high-precision map for ground truth. (d) Mobile cart.

Figure 7: A real-world smart lamppost testbed we deployed for data collection and system evaluation.

5.1 Real-world Campus Dataset

During the collection of the campus dataset, the vehicle speed is 5 m/s on average (up to around 8 m/s, i.e., 30 km/s) because of the speed limit on campus roads. We collect the vehicular location traces with a commercial GPS unit. To obtain the ground-truth location and orientation between the vehicle and infrastructure LiDARs, we build a high-precision 3D map (Fig. 7(b)) for the tested roads by repeatedly scanning the roads with LiDAR and merging the point clouds offline using commercial software. We also use a mobile cart shown in Fig. 7(d) for repeated data collection. We align each frame with the 3D map [48] to get the ground-truth pose and location. Note that such a 3D map construction is often used for generating commercial HD maps, which is extremely costly and labor-intensive and thus cannot be used for real-time data fusion. Moreover, we manually annotate the objects (vehicles and pedestrians) in the point clouds using [40] for the training data. The campus dataset covers 1.75 km, which is 1,428 point cloud pairs.

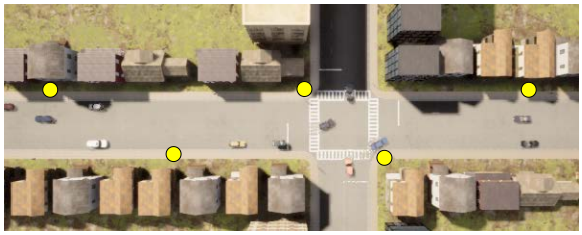


Figure 8: An example scene in CARLA dataset. The yellow dots denote the locations of roadside infrastructure.

5.2 CARLA Simulation Dataset

We render a larger dataset using CARLA [25]. Compared to the campus dataset with only simple road types and monotonic traffic flow, the CARLA dataset includes diverse road types and traffic densities (Fig. 8). Specifically, the road types include straight roads and intersections with two-way single or two lanes. We configure three different levels of traffic conditions according to the average number of objects within 50 m range of the vehicle: *light* (less than 4), *medium* (4 to 8), and *heavy* (more than 8). Besides, we set the vehicle speed up to 20 m/s (i.e., 72 km/h), which is consistent with the regular speed on urban roads. We configure 20 roadside infrastructure nodes at different locations on the CARLA map, each installing one LiDAR at the height of 4 m. Meanwhile, a LiDAR is present on a vehicle's roof, around 1.65 m above the ground. All the LiDARs on the vehicle and the infrastructure are configured with the same parameters (e.g., 32 channels, 360° FOV) used in mainstream autonomous driving datasets [18, 31]. CARLA generates the vehicle locations with errors that are extracted from the real-world GPS data traces from Fig. 2(c). The CARLA dataset covers 5.78 km, which is 6,523 point cloud pairs.

6 IMPLEMENTATION AND EVALUATION

In this section, we conduct extensive experiments to validate the performance and advantages of VIPS. We first describe the system implementation and experimental settings in Section 6.1 and define evaluation metrics in Section 6.2. Second, we present an end-to-end implementation of VIPS based on the smart lamppost testbed on campus in Section 6.3. Third, we evaluate the application-level performance of VIPS for autonomous vehicles in Section 6.4, which shows that VIPS significantly extends the vehicle's perception distance and provides them with information about the occluded objects. In addition, we evaluate the performance of VIPS in Section 6.5 and Section 6.6. Lastly, we validate VIPS's robustness in diverse road scenarios and also in the presence of substantial errors in vehicle localization and frame time in Section 6.7.

6.1 Implementation and Experiment Setup

We implement VIPS on the smart lamppost testbed on a university campus. Each lamppost equips an NVIDIA Jetson TX2 for computing and an 802.11ac WiFi router for wireless communication with vehicles (see Fig. 7(a)). The test vehicle (see Fig. 7(c)) carries a laptop to run VIPS code. The laptop equips an Intel i7 2.60 GHz CPU and an NVIDIA RTX 2070 GPU. We train the detection models with the OpenPCDet platform [52] using two datasets from Section 5 on 4 NVIDIA TITAN Xp GPUs. We set the detecting radius of our model to 50 m on the horizontal plane, and 2 m on the vertical plane. Such detection range settings follow the mainstream benchmarks [18, 31] for 3D object detection in autonomous driving. Moreover, we export the trained models in ONNX format [10] for inference using TensorRT [3] on the Jetson TX2.

6.2 Evaluation Metrics

6.2.1 RRE, RTE, and the success rate. We adopt the relative rotation error (RRE) and relative transnational error (RTE) defined in the KITTI benchmark [31] to measure the errors between the estimated transformation $\mathbf{T}_e$ (c.f. in Section 4) and the ground-truth transformation $\mathbf{T}_t$. Specifically, RRE is defined as: $RRE = \left\| F(R_t^{-1} R_e) \right\|_1$, where R_t and R_e are the rotation matrices of $\mathbf{T}_t$ and $\mathbf{T}_e$, respectively. $F(\cdot)$ denotes the function that calculates three Euler angles from the rotation matrix. RTE is defined as: $RTE = \|t_t - t_e\|_2$, where t_t and t_e are the translation vectors of $\mathbf{T}_t$ and $\mathbf{T}_e$, respectively. In addition, following the previous work [32], we define the success rate as the ratio of the estimated transformations whose RTEs are below 2 m.

6.2.2 Absolute trajectory error. We adopt the absolute trajectory error (ATE) from [45] to measure the absolute distances between estimated and ground-truth trajectories. Specifically, ATE is defined as the root mean square error from trajectory matrices: $ATE = (\frac{1}{n} \sum_{i=1}^{n} \|E(tra(\mathbf{T}_e), tra(\mathbf{T}_t))\|^2)^{\frac{1}{2}}$, where n denotes the frame number. $tra(\cdot)$ transforms the transformations to trajectory matrices, while $E(\cdot, \cdot)$ calculates the errors between the estimated and ground-truth trajectories.

6.2.3 Recall and Precision. We evaluate the detection accuracy of our perception fusion approach using recall and precision rates. A positive detection means that its intersection over union (IoU) of the bounding boxes of the transformed object and its ground truth is larger than 50%. The recall and precision are defined as: $Recall = TP/(TP + FN)$, $Precision = TP/(TP + FP)$, where TP, FN, FP are true positives, false negatives and false positives respectively. The recall rate describes how accurate VIPS's results are. The precision rate denotes how much the user can trust VIPS in its fusion results.

6.3 End-to-end System Evaluation

As shown in Fig. 7, we collect data traces using the test vehicle equipped with a top-mounted LiDAR and the mobile cart platform to evaluate the end-to-end system performance of VIPS. The data collection is based on two roads (i.e., around 180 m and 350 m with the installation of the smart lampposts) during a period of three weeks. A video clip of the real-time perception fusion of the test vehicle using object detection results is available[1]. Fig. 9 shows a sample of results on the 180 m road segment with 5 smart lampposts. The upper figure shows the estimation of the test vehicle's trajectory by VIPS. The color of the dots denotes the ATE. The average ATE is around 0.65 m, and over 80% of the trajectory is estimated within a 1 m error, which demonstrates that VIPS fuses the perception accurately. The lower figure in Fig. 9 measures the end-to-end latency of VIPS. VIPS takes an average of 58 ms per frame, which means that VIPS can well support tasks in real-time autonomous

[1] https://youtu.be/zW4oi_EWOu0

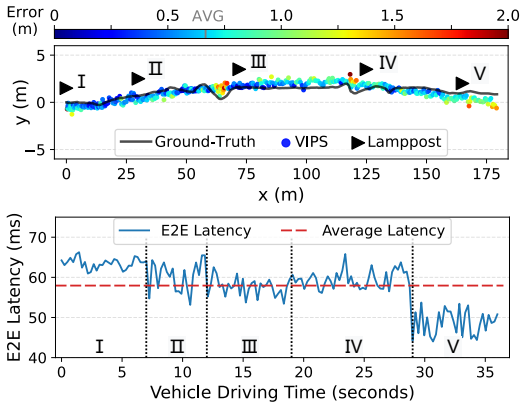


Figure 9: An End-to-end evaluation on a road segment. Above: errors of the test vehicle's trajectory estimated by VIPS. Bottom: the end-to-end latency of VIPS during the driving trace.

driving. In addition, we observe that the accuracy and the end-to-end latency of VIPS vary across lampposts. For example, VIPS generates larger trajectory errors and smaller perception fusion latency when assisted by the lamppost V. This is because there are fewer objects in the lamppost and vehicle's views when the vehicle passes by the lamppost. We evaluate the influence of the traffic density in Section 6.5 and Section 6.7.

6.4 Micro Benchmarks

Autonomous vehicles can benefit from VIPS which provides vehicles with more information about objects on the road. Fig. 10 shows the images and point clouds of two scenes. (a) and (b) are captured from the CARLA dataset, in which most occluded objects (the red boxes) are in distance from the ego vehicle (i.e., the test vehicle). In this case, the vehicle can better plan its route with the fused perception from the infrastructure. (c) and (d) are from the campus dataset, in which most objects are close to the ego vehicle. The vehicle can leverage the relative location and the motion of the occluded vehicle for accident alarming to improve driving safety. Another key advantage of VIPS is that the ego vehicle only receives object detection results from the infrastructure without the raw point clouds (colored in blue in Fig. 10(b, d)). The figures show that VIPS can: a) enable the detection of occluded/distant objects; and b) fuse the object detection from both views.

We evaluate the additional detected objects with the help of VIPS. Fig. 11 shows the cumulative number of detected objects within the distance from the ego vehicle in the CARLA dataset. For example, within the range of 120 m, there are 9 and 13 detected objects on average from the views before and after the perception fusion, respectively. Since the typical detection range of vehicles is 50 m, VIPS can extend the perception range of the ego vehicle to 120 m, which enables the vehicle to see significantly further. Meanwhile, VIPS can provide four more detected objects on average to support downstream autonomous driving. In addition to the perception extension, VIPS can provide important information about nearby occluded objects when the range is less than 50 m, which can be used by safety-critical applications like accident alarming and pedestrian intrusion detection. Besides, as the increased perception capability can vary with the relative positions of the ego-vehicle and the infrastructure, we also count the average number of objects the vehicle can detect at different distances from the infrastructure. Fig. 12 shows that vehicles can benefit more from the infrastructure when they are far from it. However, VIPS can increase the number of objects detected by vehicles even when the vehicles are very close to the infrastructure (i.e., <10 m) since the LiDAR on the infrastructure is higher off the ground, thus less susceptible to occlusions.

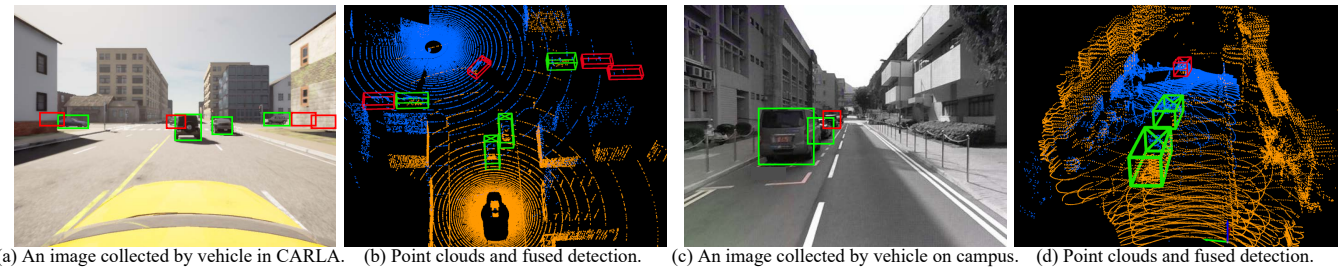
Table 2: The average size of shared data per frame and the transmission time measured on an 802.11ac network.

Shared data type	Shared data size (KB)	Transmission time (ms)
Raw point cloud	412.4	31.2
Feature points	56.9	5.4
Detection results in VIPS	**1.2**	**0.17**

6.5 Real-Time Performance

We evaluate the runtime of VIPS's individual components and the whole system separately. Fig. 13 shows the runtime of each component under different traffic densities using the CARLA dataset. The error bars indicate the 5th and 95th percentiles of the runtime across all frames with the same setting. Since the computational time of motion-aware rectification is trivial, we include it in the overall time. The results show that the overall computational time of VIPS is less than 70 ms under all traffic conditions, which is less than the frame rate of LiDAR (i.e., 10 Hz). Therefore, VIPS enables a real-time perception of surrounding objects by vehicles with the assistance of the roadside infrastructure. The 3D object detection takes around half of the total computational time, which only depends on the size of the point cloud. We note that this runtime can be further reduced with the development of more efficient 3D object detection algorithms. While the runtime of the other three modules grows with the number of objects, the total runtime is always less than 40 ms under heavy traffic.

We also compare the communication overhead of VIPS with the approaches that transmit raw point cloud data or feature points. Here we adopt a state-of-the-art SLAM algorithm LIO-SAM [48] to extract the feature points. We measure the data transmission time traces outdoor under an 80 MHz 802.11ac network with a WiFi router wire connected to a Jetson TX2 (to simulate the infrastructure) and an 802.11ac-compliant laptop moving at a speed of 5 m/s (to simulate the vehicle). We use the existing WiFi on the testbed for evaluations although VIPS can broadcast perception results from the infrastructure to the vehicle via other V2X protocols. The average data volume and the transmission time per frame are shown in Table 2. VIPS transmits 1.2 KB of object detection and tracking results at 10 Hz in an average of 0.17 ms, which is negligible in the overall system overhead. VIPS reduces the data transmission volume and time by about 343× and 183× compared with transmitting raw point cloud data, and 47× and 32× compared with transmitting feature points. Therefore, VIPS can be deployed with a wide range of V2X networks that even have low communication bandwidth.



(a) An image collected by vehicle in CARLA. (b) Point clouds and fused detection. (c) An image collected by vehicle on campus. (d) Point clouds and fused detection.

Figure 10: Two scenes from CARLA and campus datasets. Green and red boxes are objects from the vehicle's view and the infrastructure's view, respectively. The yellow and blue dots are point clouds from the vehicle and infrastructure, respectively.

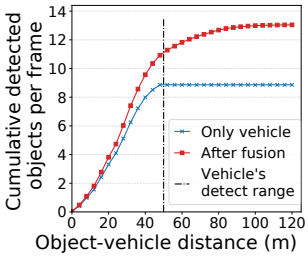


Figure 11: Average number of objects that vehicle can detect within range.

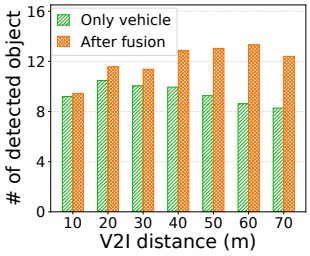


Figure 12: Benefit to vehicle at different distances to infrastructure.

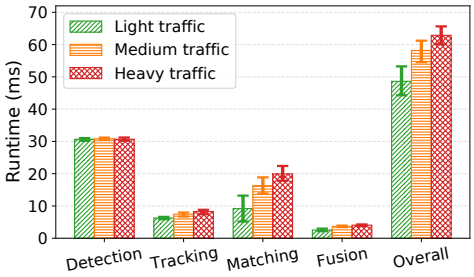


Figure 13: Runtime of each module and overall system under different traffic conditions.

6.6 Accuracy of Perception Fusion

We conduct extensive accuracy evaluation on VIPS from three aspects: object alignment, trajectory estimation, and object perception.

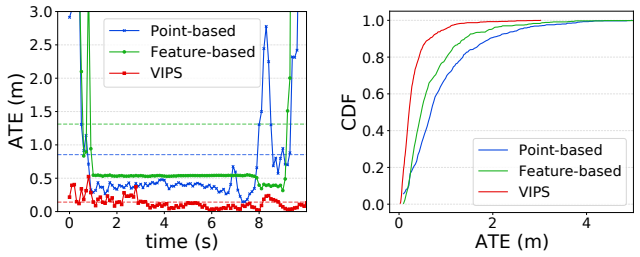
6.6.1 Object alignment Accuracy. As mentioned in Section 4, VIPS computes the transformation **T** between the perspectives of the vehicle and the infrastructure by aligning the co-visible objects, based on which the vehicle can fuse the detection results from the infrastructure into its scene perception. Existing approaches obtain such a transformation by registering the raw point clouds or extracted feature points. We evaluate the object alignment accuracy of VIPS by comparing it with four baselines, including three point-based algorithms: PP-ICP [42], Filter-REG [30], FAST-GICP [35], and one feature-based registration approach LIO-SAM [48]. Other recent works, such as [26, 32], rely on pre-defined regularly shaped objects such as traffic signs, crosswalks, etc., which are not always available in general road scenarios.

Table 3: Alignment performance comparison between VIPS and existing algorithms. The left and right numbers denote the results on the CARLA dataset and the campus dataset, respectively.

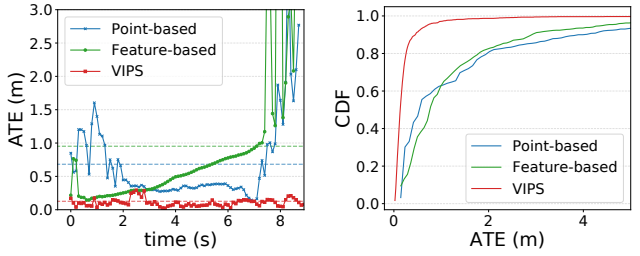
Type	Methods	RRE (°)	RTE (m)	Success rate (%)	Time (ms)
Point-based	[42]	0.61/3.69	1.28/1.59	79.7/77.9	351.3/98.7
	[30]	3.88/6.72	3.41/1.69	64.7/61.5	113.2/119.6
	[35]	0.41/1.29	1.24/0.49	86.5/89.3	63.3/72.2
Feature-based	[48]	0.57/3.66	1.11/1.29	82.6/84.5	82.7/51.6
VIPS		**0.39/0.92**	**0.28/0.44**	**97.8/93.4**	**56.3/49.4**

We conduct the evaluation based on the RRE, RTE, and success rate metrics using both the CARLA and campus datasets. The results are shown in Table 3. VIPS outperforms the existing methods on both of the two datasets, despite that VIPS only utilizes the object detection results from the point clouds. This is because VIPS extracts the overlaps, i.e., the co-visible objects in the two point clouds, and obtains the correspondence of them, which removes the interference from non-overlapped parts of the two point clouds. Although the 3D object detection results may contain errors, the interference of these errors can be reduced by leveraging all the co-visible objects according to their detection confidence and matching scores, as mentioned in Section 4.5. Another key insight in Table 3 is that the runtime of VIPS is less than all the existing approaches. In addition, the output of VIPS is the fused object detection results, which can be directly used by the downstream applications of autonomous driving based on object detection results, including motion prediction, driving decision making, etc. In contrast, the existing approaches only align two point clouds together while the vehicle still needs to conduct object detection on the merged point cloud for the driving applications.

6.6.2 Estimated Trajectory Accuracy. We also evaluate the accuracy of the vehicle trajectory, which indicates the performance of the vehicle fusing perception information from the infrastructure in continuous time. The trajectory is estimated with the transformation **T** in every frame and the position of the fixed infrastructure. Here we compare VIPS with two baselines including a point-based approach [35] and a feature-based approach [48] using both the CARLA and campus datasets. Fig. 14(a) and 14(c) show the ATE of trajectories estimated by the three approaches on a sample sequence in the CARLA and campus datasets, respectively. VIPS outperforms



(a) ATE of a CARLA trace. (b) CDF of ATE in CARLA data.



(c) ATE of a campus trace. (d) CDF of ATE in campus data.

Figure 14: Evaluations of trajectory estimation. (a, c): The ATE of VIPS and two baselines based on the CARLA and the campus datasets, respectively. (b, d): The CDF of ATE of VIPS and two baselines on the CARLA and the campus datasets, respectively.

the baselines over almost the entire sequence, especially at the beginning and the end, where the vehicle enters and leaves the area around the infrastructure. The point-based and feature-based approaches perform poorly due to limited FOV overlaps when the vehicle is far from the infrastructure, where the non-overlapped portions of the point clouds can significantly affect the alignment accuracy. The point-based method performs similarly or even better than VIPS only occasionally when the FOVs are mostly overlapped. However, the point clouds from the infrastructure and the vehicle have significant viewing angle differences in most cases. VIPS overcomes this challenge by conducting the alignment only according to the extracted co-visible objects. We also estimate all the trajectories of both the CARLA and campus datasets with VIPS and the two baselines. Fig. 14(b) and 14(d) show the cumulative distribution functions (CDFs) of ATE. The average errors of trajectories estimated by VIPS are about 0.5 m and 0.6 m on the CARLA and campus datasets, respectively, while the baselines can achieve such accuracy only around 25% of the time. In summary, VIPS enables the vehicle to accurately fuse perception information from smart lampposts across the two datasets.

6.6.3 Object Perception Accuracy. In order to validate VIPS at the application level, we evaluate the accuracy of VIPS perceiving objects on both the CARLA and the campus datasets. We compare VIPS with the point-based and feature-based baselines used earlier, we also present the vehicle's object detection result without information from the infrastructure. Note that the baseline approaches are designed for point cloud registration instead of object perception. Here we use their output transformation to align the object detection results and compare them with the output of VIPS. Table 4

Table 4: Object perception recall and precision of VIPS and existing approaches (IoU threshold is set to 0.5). The left and right numbers denote the results on the CARLA dataset and the campus dataset, respectively. "N/A" means no object is detected.

Methods	Near (<50m)		Far (>50m)	
	Recall (%)	Precision (%)	Recall (%)	Precision (%)
Only vehicle	80.2/82.6	85.8/80.4	N/A	N/A
Point-based	86.3/85.2	83.9/87.1	50.3/78.2	63.5/67.8
Feature-based	83.7/83.5	81.8/85.5	37.8/41.9	46.9/37.5
VIPS	**87.7/90.1**	**86.7/89.9**	**69.8/84.9**	**75.5/84.0**

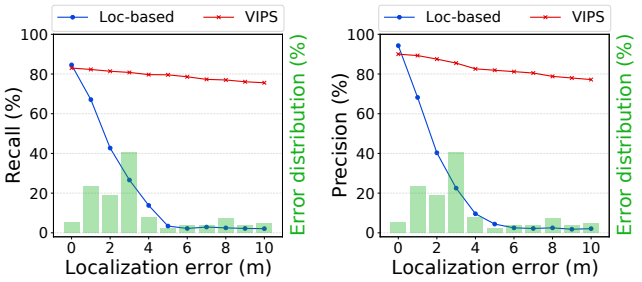


Figure 15: Recall (left) and Precision (right) of VIPS's perception fusion under different localization errors (IoU threshold is set to 0.5).

presents recall and precision of object perception results. We divide them into two parts: the recall and precision of objects within or out of the vehicle's detection range (50 m), denoted by "Near" and "Far", respectively. For the "Far" group, the vehicle's perception of objects is entirely dependent on the detection results from the infrastructure, and VIPS significantly outperforms baseline methods because it achieves a more accurate alignment between the vehicle and infrastructure perspectives. As shown in the "Near" group, VIPS also improves the vehicle's perception within its detection range because VIPS helps the vehicle perceive occluded objects and corrects part of the vehicle's 3D object detection errors with the detection result from the infrastructure.

6.7 Robustness of VIPS

6.7.1 Vehicle localization error. To validate VIPS's robustness to the vehicle localization error, we compare VIPS with the localization-based perception fusion based on the GPS traces in the campus dataset. The red and blue curves in Fig. 15 show the recall and precision of VIPS and the localization-based perception fusion under different localization errors, respectively. The green bars in Fig. 15 present the distribution of GPS localization error. The performance of the localization-based method drops sharply with the increase of the localization error, while the recall of VIPS is over 75% and the precision is over 78% across all localization errors. This is because VIPS exploits the information that is consistent over the vehicle's location (such as relative positions and semantic labels of objects) to match co-visible objects from the vehicle's and infrastructure's views. Hence, VIPS is robust to localization errors.

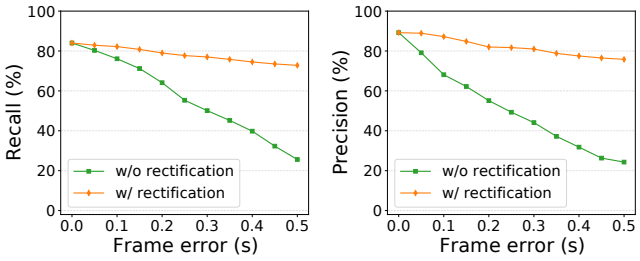


Figure 16: Recall (left) and Precision (right) of VIPS's perception fusion under different frame errors (IoU threshold is set to 0.5).

Table 5: Recall and precision of object perception in different road types and traffic conditions.

Road type	Traffic condition	Recall (%)	Precision (%)
Intersection	Light	78.5	83.1
	Heavy	83.4	83.7
Straight road	Light	78.4	78.8
	Heavy	87.2	87.5

6.7.2 Frame errors between vehicle and infrastructure. As described in Section 4.3, VIPS adopts MOT to obtain the motion information of the detected objects, thus rectifying the frames from the infrastructure to align the time with the vehicle's frames. Such a design makes VIPS resilient to fluctuating network conditions and computing resources while the vehicle is moving. To assess how tolerant VIPS is for frame errors, we compare the recall of object perception of VIPS with and without the frame rectification using the campus dataset, which is shown in Fig. 16. The result shows that the performance of object perception drops significantly when fusing the objects from the vehicle and the infrastructure with misaligned frames. However, with the frame rectification, the vehicle can utilize the time-aligned information from the infrastructure to achieve a robust perception fusion when the error is up to 0.5 s, which is 5× of the frame interval. This evaluation confirms VIPS's robustness to frame errors that may be caused by the missing frames or the misalignment due to different frame rates, packet loss, or constrained computing resources on the infrastructure.

6.7.3 Road types and traffic conditions. We evaluate the accuracy of VIPS in different road types and traffic conditions using the CARLA dataset. The results are presented in Table 5. VIPS performs well on both intersections and straight roads (e.g., VIPS can achieve over 78% recall and precision under light traffic on a straight road), which shows that VIPS can adapt to different types of the spatial distribution of objects on the roads. We also note that the accuracy of VIPS perceiving objects drops slightly in light traffic compared to that in heavy traffic, which is due to the fewer co-visible objects in light traffic. This is acceptable because, in light traffic, the vehicles are less susceptible to occlusions and are less reliant on detection results from the infrastructure.

7 DISCUSSION

Resource Limitation of Intelligent Infrastructure. In this work, we assume vehicles are equipped with sufficient computing resources, whereas intelligent infrastructures are relatively resource-constrained. This is consistent with the fact that the existing roadside infrastructures (e.g., lampposts and traffic lights) have limited power supply. Therefore, in the design of VIPS, the infrastructure processes its own data and broadcasts the detection results to vehicles. However, if the infrastructure has sufficient resources, the vehicle can offload the frame rectification (Section 4.3) and perception fusion (Section 4.4-4.5) to the infrastructure. In this case, the vehicle transmits its detection results to the infrastructure and receives the fused perception in real time. Moreover, the infrastructure can analyze the occlusions by locating and tracking the vehicle and hence remove the static objects invisible to a vehicle, which further reduces the overhead of graph matching.

Inter-Vehicle Perception Fusion. Although VIPS focuses on the perception fusion between the vehicle and infrastructure, our core idea can be applied to inter-vehicle perception fusion as well. For example, a vehicle can receive detection results from surrounding vehicles and fuse them into its field of view with the graph representations. While the FOV overlaps between the vehicles are usually more limited than in the vehicle-infrastructure scenario, the system can leverage multiple vehicles or leverage other sources of information (e.g., precise vehicle localization) to facilitate perception fusion.

Network Settings. There exist a variety of communication technologies between vehicles and roadside infrastructures, such as cellular [12, 17], WiFi [63], millimeter-wave [57], 802.11p [27], and 802.11bd [44]. We note that the design of VIPS (c.f. Section 6.5) does not require a specific network setting and can greatly reduce the network bandwidth for data exchange between vehicles and roadside infrastructures.

Moreover, in the further, we will further optimize the object detection model and improve the security of VIPS. A model with higher precision or more classes can benefit the detection of co-visible objects and thus boost the performance of VIPS. While we assume that the data transmission between the infrastructure and the vehicle is trusted, we will investigate security issues like adversarial examples for infrastructure-assist autonomous driving, with a focus on perception fusion.

8 CONCLUSION

In this paper, we present VIPS, a novel system that fuses the objects detected by the vehicle and the infrastructure to expand the vehicle's perception in real time, which facilitates a number of autonomous driving applications. We implement VIPS end-to-end and evaluate its performance on two self-collected datasets. The experiment results show that VIPS outperforms the existing approaches in accuracy, robustness, and efficiency.

ACKNOWLEDGMENTS

This work is supported in part by the Hong Kong Innovation and Technology Commission under Grant PiH/124/22, and the Centre for Perceptual and Interactive Intelligence (CPII) under Grant EW01610 (RP4-3).

REFERENCES

[1] 2021. Self-driving technology: Automated Transportation Systems. https://www.tusimple.com/technology/

[2] 2021. Velodyne's HDL-32E surround lidar sensor. https://velodynelidar.com/products/hdl-32e/

[3] 2022. Nvidia TENSORRT. https://developer.nvidia.com/tensorrt

[4] 2022. The providentia++ project. https://innovation-mobility.com/en/project-providentia/

[5] 2022. Street light. https://en.wikipedia.org/wiki/Street_light

[6] [n.d.]. Apollo. https://apollo.auto/.

[7] [n.d.]. The Autoware Foundation - open source for autonomous driving. https://www.autoware.org/

[8] [n.d.]. Carla Documentation. https://carla.readthedocs.io/en/stable/

[9] [n.d.]. The MEC-view system. https://www.uni-due.de/~hp0309/index.php/en/project-issues

[10] [n.d.]. Open Neural Network Exchange. https://onnx.ai/

[11] [n.d.] url=https://www.livoxtech.com/horizon, journal=Livox. Horizon.

[12] Fawad Ahmad, Hang Qiu, Ray Eells, Fan Bai, and Ramesh Govindan. 2020. {CarMap}: Fast 3D Feature Map Updates for Automobiles. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 1063–1081.

[13] PhD Alessandro Lori. 2019. Are self-driving cars safe? https://www.verizonconnect.com/resources/article/are-self-driving-cars-safe/

[14] 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).

[15] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. 2019. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1426–1433.

[16] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*. IEEE, 3464–3468.

[17] Michael Buchholz, Johannes Christian Muller, Martin Herrmann, Jan Strohbeck, Benjamin Volz, Matthias Maier, Jonas Paczia, Oliver Stein, Hubert Rehborn, and Rudiger-Walter Henn. 2021. Handling Occlusions in Automated Driving Using a Multiaccess Edge Computing Server-Based Environment Model From Infrastructure Sensors. *IEEE Intelligent Transportation Systems Magazine* (2021).

[18] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11621–11631.

[19] 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.

[20] 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.

[21] Titus Cieslewski, Siddharth Choudhary, and Davide Scaramuzza. 2018. Data-efficient decentralized visual SLAM. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2466–2473.

[22] Christian Creß and Alois C Knoll. 2021. Intelligent Transportation Systems With The Use of External Infrastructure: A Literature Survey. *arXiv preprint arXiv:2112.05615* (2021).

[23] Vinayak V Dixit, Sai Chand, and Divya J Nair. 2016. Autonomous vehicles: disengagements, accidents and reaction times. *PLoS one* 11, 12 (2016), e0168054.

[24] Zhen Dong, Fuxun Liang, Bisheng Yang, Yusheng Xu, Yufu Zang, Jianping Li, Yuan Wang, Wenxia Dai, Hongchao Fan, Juha Hyyppä, et al. 2020. Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing* 163 (2020), 327–342.

[25] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[26] Bertrand Douillard, A Quadros, Peter Morton, James Patrick Underwood, Mark De Deuge, S Hugosson, M Hallström, and Tim Bailey. 2012. Scan segments matching for pairwise 3D alignment. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, 3033–3040.

[27] Stephan Eichler. 2007. Performance evaluation of the IEEE 802.11 p WAVE communication standard. In *2007 IEEE 66th Vehicular Technology Conference*. IEEE, 2199–2203.

[28] Davi Frossard and Raquel Urtasun. 2018. End-to-end learning of multi-sensor 3d tracking by detection. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 635–642.

[29] Michael Gabb, Holger Digel, Tobias Müller, and Rüdiger-Walter Henn. 2019. Infrastructure-supported perception and track-level fusion using edge computing. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1739–1745.

[30] Wei Gao and Russ Tedrake. 2019. Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11095–11104.

[31] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

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

[33] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. (1960).

[34] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taixé. 2021. Eagermot: 3d multi-object tracking via sensor fusion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11315–11321.

[35] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. 2021. Voxelized gicp for fast and accurate 3d point cloud registration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11054–11059.

[36] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[37] Sampo Kuutti, Saber Fallah, Konstantinos Katsaros, Mehrdad Dianati, Francis Mccullough, and Alexandros Mouzakitis. 2018. A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet of Things Journal* 5, 2 (2018), 829–846.

[38] 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.

[39] Marius Leordeanu and Martial Hebert. 2005. A spectral technique for correspondence problems using pairwise constraints. (2005).

[40] E Li, Shuaijun Wang, Chengyang Li, Dachuan Li, Xiangbin Wu, and Qi Hao. 2020. SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. 1108–1115. https://doi.org/10.1109/IV47402.2020.9304562

[41] 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.

[42] Kok-Lim Low. 2004. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina* 4, 10 (2004), 1–3.

[43] Qian Luo, Yurui Cao, Jiajia Liu, and Abderrahim Benslimane. 2019. Localization and navigation in autonomous driving: Threats and countermeasures. *IEEE Wireless Communications* 26, 4 (2019), 38–45.

[44] 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.

[45] David Prokhorov, Dmitry Zhukov, Olga Barinova, Konushin Anton, and Anna Vorontsova. 2019. Measuring robustness of Visual SLAM. In *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE, 1–6.

[46] 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.

[47] Siegfried Seebacher, Bernd Datler, Jacqueline Erhart, Gerhard Greiner, Manfred Harrer, Peter Hrassnig, Arnold Präsent, Christian Schwarzl, and Martin Ullrich. 2019. Infrastructure data fusion for validation and future enhancements of autonomous vehicles' perception on Austrian motorways. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 1–7.

[48] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. 2020. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5135–5142.

[49] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. 2020. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10529–10538.

[50] Steve Shwartz. 2021. Are self-driving cars really safer than human drivers? https://thegradient.pub/are-self-driving-cars-really-safer-than-human-drivers/

[51] Jack Stilgoe. 2020. Who Killed Elaine Herzberg? In *Who's Driving Innovation?* Springer, 1–6.

[52] OpenPCDet Development Team. 2020. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet.

[53] Mabrouk Touahmia. 2018. Identification of risk factors influencing road traffic accidents. *Engineering, Technology & Applied Science Research* 8, 1 (2018), 2417–2421.

[54] Manabu Tsukada, Takaharu Oi, Masahiro Kitazawa, and Hiroshi Esaki. 2020. Networked roadside perception units for autonomous driving. *Sensors* 20, 18

(2020), 5320.

[55] Bill Vlasic and Neal E Boudette. 2016. Self-driving Tesla was involved in fatal crash, US says. *New York Times* 302016 (2016).

[56] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. 2019. Mots: Multi-object tracking and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7942–7951.

[57] Song Wang, Jingqi Huang, and Xinyu Zhang. 2020. Demystifying millimeter-wave V2X: Towards robust and efficient directional connectivity under high mobility. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*.

[58] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 2020. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10359–10366.

[59] Yan Yan, Yuxing Mao, and Bo Li. 2018. Second: Sparsely embedded convolutional detection. *Sensors* 18, 10 (2018), 3337.

[60] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. 2019. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1951–1960.

[61] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. 2021. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11784–11793.

[62] Lijun Yu, Dawei Zhang, Xiangqun Chen, and Alexander Hauptmann. 2018. Traffic danger recognition with surveillance cameras without training data. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 1–6.

[63] 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.

[64] Zhihe Zhao, Zhehao Jiang, Neiwen Ling, Xian Shuai, and Guoliang Xing. 2018. ECRT: An edge computing system for real-time image-based object tracking. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. 394–395.

[65] Feng Zhou and Fernando De la Torre. 2012. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 127–134.

[66] Yi Zhou and Ling Shao. 2018. Aware attentive multi-view inference for vehicle re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6489–6498.

[67] Yin Zhou and Oncel Tuzel. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4490–4499.