VRF: Vehicle Road-side Point Cloud Fusion

Kaleem Nawaz Khan*
Rochester Institute of Technology
kk5271@rit.edu

Ali Khalid*
Rochester Institute of Technology
ak5013@rit.edu

Yash Turkar
University at Buffalo
yashturk@buffalo.edu

Karthik Dantu
University at Buffalo
kdantu@buffalo.edu

Fawad Ahmad
Rochester Institute of Technology
fawad@cs.rit.edu

ABSTRACT

Autonomous vehicles and human drivers are prone to line-of-sight limitations. Road-side mounted 3D sensors like LiDARs can augment a vehicle's on-board perception. However, this entails fusing 3D frames at low latency and high accuracy. Road-side and vehicle 3D frames are captured from different viewpoints. This adversely affects alignment accuracy and can be computationally expensive. To this end, VRF optimizes for both latency and accuracy by decoupling the alignment process into indirect and direct alignments. First, VRF indirectly aligns the 3D frames by aligning them to a common reference point *i.e.,* a vehicle's on-board 3D map. Then, it directly aligns the two point clouds to refine this alignment. To ensure high accuracy, it incorporates novel offline registration and alignment accuracy forecasting modules. To ensure low latency, it uses a fast fusion pipeline that caches previous and offline computations. To our knowledge, VRF is the first vehicle road-side cooperative system to ensure cm-level accuracy and end-to-end latency less than 20 ms. Most importantly, its latency is below the 100 ms threshold required for autonomous vehicles to react to external events. Finally, VRF can improve reaction time to external events by as much as 5 seconds[1].

CCS CONCEPTS

• **Networks** → **Sensor networks**; • **Computing methodologies** → **Cooperation and coordination**;

KEYWORDS

Autonomous Cars, Cooperative Perception, Infrastructure-assisted Autonomous Driving

ACM Reference Format:
Kaleem Nawaz Khan, Ali Khalid, Yash Turkar, Karthik Dantu, and Fawad Ahmad. 2024. VRF: Vehicle Road-side Point Cloud Fusion . In *The 22nd Annual International Conference on Mobile Systems, Applications and Services (MOBISYS '24), June 3–7, 2024, Minato-ku, Tokyo, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3643832.3661874

*Authors made equal contributions to this paper.
[1]Video demos at: https://www.youtube.com/@NSSL_Official/

MOBISYS '24, June 3–7, 2024, Minato-ku, Tokyo, Japan

ACM ISBN 979-8-4007-0581-6/24/06.
https://doi.org/10.1145/3643832.3661874

Figure 1: The red truck(s) block the gray vehicle's view. It cannot see the oncoming yellow cab violating the red light in (a) and taking the unprotected left turn in (b).

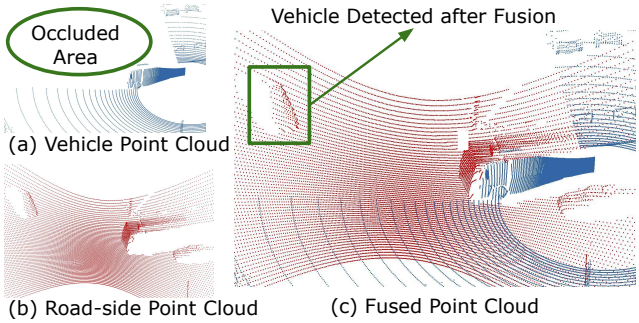


Figure 2: By fusing the RSU point cloud with the vehicle point cloud, the vehicle can see the red-light violator vehicle (Fig. 1a).

1 INTRODUCTION

The past decade has seen a significant infusion of technology in vehicles. This includes blind spot monitoring [28], adaptive cruise control [30], lane keep assist [29], and autonomous driving with limited or no human intervention. Critical to these technologies is the ability of the vehicle to understand its surroundings. For this, vehicles use sensors like cameras, LiDARs, and RADARs to build 3D representations of their surroundings. A LiDAR sends out millions of light pulses multiple times per second. From the reflections of these light pulses, a LiDAR builds a 3D point cloud (Fig. 2). This point cloud consists of points defined by their 3D positions along with other attributes like intensity and color. However, these sensors are prone to occlusions and line-of-sight limitations (Fig. 2a).

Line-of-sight limitations are not only a problem for these sensors but also for human drivers. The National Highway Transportation and Safety Authority (NHTSA [43]) has identified several scenarios in which occlusions can cause traffic accidents [42] (Fig. 1). In one such scenario (Fig. 1a), the gray vehicle (equipped with a LiDAR) will crash into the oncoming red-light violating vehicle (yellow cab) because the red truck occludes its LiDAR's view (Fig. 2a). If the gray vehicle could see the oncoming vehicle, it would be able to avoid the

Corrected Version of Record. V.1.1. Published September 4, 2024.

traffic accident. One way to do this is to extend the vehicle's sensor range with road-side unit (RSU) sensors.

Well-positioned RSU LiDARs can monitor the entire intersection and are less prone to occlusions (Fig. 2b). In such a scenario, the RSU LiDARs share raw 3D point clouds with vehicles. Vehicles fuse these point clouds with ones from their on-board sensors by aligning them in their own coordinate system to overcome line-of-sight limitations (Fig. 2c). Increasing RSU LiDAR [13, 26, 35, 45, 46, 55], 5G [1], and edge-compute deployments [61] are making this possible. However, this is challenging because of the high accuracy and low latency requirements of autonomous vehicles.

An autonomous driving pipeline reacts to its surroundings in three steps *i.e.,* perception, planning, and control. Perception understands the scene and localizes the vehicle in it. Planning plans a trajectory for the vehicle. Control follows that trajectory using low-level control signals like steering, throttle, and brake. Industry standards require that an autonomous vehicle runs its entire pipeline at least 10 times per second with a 99^{th} percentile latency less than 100 ms [11, 50]. In addition, these standards require that a vehicle localizes itself and objects in its surroundings with cm-level accuracy [33]. This, in turn, means that the RSU LiDAR point cloud must be fused with the vehicle's point cloud with high accuracy and low latency.

Aligning the vehicle and RSU point clouds is challenging because they are captured from very different viewpoints. As a result, the overlapping region between the point clouds can be small, and sparse (*i.e.,* contain fewer 3D points). Alignment algorithms, however, require a large overlap between the point clouds. This makes fusion challenging. To tackle this, our *key insight is to de-couple the alignment.* Instead of aligning two point clouds directly with one another, we align them to a common third point cloud with which they share a large overlap *i.e.,* a vehicle's on-board 3D map. By doing so, we can indirectly align the two point clouds.

A 3D map is a dense 3D point cloud of the environment, containing points belonging only to static objects (*e.g.,* the road, sidewalk, traffic-signs). Autonomous vehicles localize themselves by aligning their LiDAR point clouds to a common on-board 3D map. This technique is very well-established in the industry (Waymo [53], Cruise [18], Zoox [67]) and in open-source autonomous driving stacks (Autoware [41], Baidu Apollo [12]).

In our approach, both the vehicle and RSU LiDAR align themselves to the 3D map. Then, using their poses in the 3D map, the vehicle transforms the RSU point cloud into its own view, thus indirectly aligning them. This has three advantages. In terms of latency, because the RSU LiDAR is stationary, we can align it with the 3D map offline. In terms of accuracy, because the 3D map is captured from a vehicle's perspective, it has a larger, denser overlapping region with the vehicle. In terms of overlap, because we align the two point clouds to a 3D map (not to each other), we can align them even if they do not have a direct overlap between them. However, this is not straightforward.

Challenges. Scan matching techniques like Normal Distribution Transform (NDT [10]) and Iterative Closest Point (ICP [62]) can accurately align a vehicle's point cloud to a 3D map, but are not effective for point clouds captured from RSU LiDARs. This is because they require an accurate coarse-grained alignment between the point clouds, which standard technologies like GPS cannot provide. On the other hand, feature-matching algorithms [49, 66] are not robust to changes in viewpoints.

Even after aligning the RSU point cloud to the 3D map, the map contains noise which can affect the fusion accuracy of the point clouds from the RSU and the vehicle. An additional lightweight direct alignment of RSU and vehicle point clouds can significantly improve fusion accuracy in some cases. However, it is not possible to determine accuracy without actually doing the fusion and comparing it to ground truth.

Accurately fusing the point clouds through multiple alignment operations comes at the cost of increased network and compute latency. Transmitting raw RSU point clouds incurs network latency. On the other hand, the vehicle must align itself to the 3D map, receive the RSU point cloud through the network, and then run a direct alignment with it. Combined, these operations can incur significant compute latency.

Contributions. Our key design principle is to *minimize online, per-frame operations*. For this, we *run operations offline and re-use computations from previous frames*. In doing so, we build VRF which makes the following contributions.

- To align RSU point clouds to the vehicle's 3D map, we propose an offline coarse-grained alignment algorithm that aligns the RSU point cloud with the 3D map. We use this coarse-grained alignment as input to a scan-matching algorithm to obtain a finer alignment.
- To decide whether to perform direct alignment, we propose an alignment accuracy forecasting algorithm. This algorithm forecasts if direct alignment will improve fusion accuracy. It uses the *inter-point density* in the overlapping region between the two point clouds to estimate the chances of a successful alignment.
- We propose a fast fusion software stack that re-uses computations from prior frames, leverages application-specific settings and GPU offloading and uses careful pipelining to reduce end-to-end fusion latency.

An end-to-end implementation of VRF can achieve 5 cm fusion accuracy in 20 ms on modest hardware. Relative to prior work [25], this is an *order of magnitude improvement in both accuracy and latency.* To our knowledge, VRF *is the first end-to-end system that can enable autonomous vehicles to use raw fused 3D point clouds and react to them within the 100 ms latency budget.*

2 BACKGROUND AND MOTIVATION

Point Cloud Fusion. Point cloud fusion is a two-step process: alignment and stitching. Given a point cloud P_r captured from a RSU LiDAR and P_v captured from a vehicle LiDAR , the goal of point cloud alignment is to find a rigid transformation T that aligns P_r to P_v. The transformation T is a 4x4 matrix, consisting of a rotation matrix R and a translation vector t. This transformation T, when applied to P_r, yields a transformed point cloud $P_r^{'}$ in the same coordinate system as P_v. Stitching simply appends $P_r^{'}$ to P_v to obtain a fused point cloud P_f that contains all the points from $P_r^{'}$ and P_v.

Scan-matching. Broadly speaking, there are two types of alignment algorithms[2]: feature-matching [49, 66] and scan-matching [10, 62]. Of these, scan-matching is more accurate but compute-intensive. ICP (Iterative Closest Point [62]), a scan-matching technique, iteratively computes a transformation matrix T that minimizes the 3D distance

[2]Prior literature uses alignment and registration interchangeably.

Technique	Error (cm)	Latency (ms)
ICP	∞	523
GO-ICP [58]	∞	207
ICP with initial guess (GPS)	584	361
VRF	*1.55*	*12.2*

Table 1: Even with a reasonable initial guess, ICP and its variants cannot accurately align vehicle and RSU point clouds and consume significant latency. VRF, on the other hand, can align point clouds with cm-level accuracy at low latency.

from every point in P_r to its nearest neighbor in P_v. To do this, ICP requires both a large overlap between the two point clouds and a coarse-grained alignment between them. This coarse-grained alignment is an accurate initial guess of the transformation matrix T. Since vehicle and RSU point clouds are captured from very different viewpoints, they have low overlap. Hence, ICP cannot align the two point clouds (Tbl. 1). Partial matching techniques such as GO-ICP [58] are suitable for aligning dense point clouds with significant overlap. However, in the context of vehicle RSU point cloud fusion where the overlap is both small and sparse, they fail to converge. Even while using GPS as an initial guess, the alignment error[3] is significant (584 cm). This is because GPS can only estimate the LiDAR's position, not orientation. Additionally, ICP incurs significant latency. VRF, on the other hand, can register point clouds within 2 cm in less than 15 ms.

Prior Work in RSU Vehicle Fusion. For faster and more accurate vehicle RSU point cloud fusion, prior work has used feature-based registration techniques that extract and match features across point clouds. Doing so, these works have either optimized for accuracy [25], or latency [51], but not both (Tbl. 2). VI-Eye [25] extracts and matches scene-specific features (*e.g.,* road markings, and traffic signs) from point clouds for alignment. With this, it achieves decimeter-level accuracy (15 cm). However, the alignment latency, not including the time to transmit the point clouds, is on the order of 100s of milliseconds. To reduce latency, VIPS [51] further processes the point clouds to extract, and match vehicle bounding boxes. This comes at the cost of accuracy (*i.e.,* VIPS has accuracy on the order of 28-44 cm). Moreover, both works require a significant amount of overlap between the vehicle and RSU point clouds, without which they cannot align them.

Early Vs Late Fusion. Early fusion approaches (VI-Eye [25]) transmit, and fuse raw 3D point clouds. This enables generalization to multiple downstream modules (*e.g.,* localization, path-planning, and drivable space detection) and can even help detect partially visible objects. Fusing raw point clouds is relatively more accurate but can incur significant compute and network latency. To this end, late fusion approaches [15, 37, 51] process point clouds independently at the vehicle/RSU and then transmit and fuse the processed information. This reduces compute and network latency at the cost of accuracy and generalizability. As such, VRF uses early fusion for accuracy and generalizability, but we design it so that it ensures low latency as well.

Requirements for RSU Fused Perception. An autonomous driving stack must perceive, plan, and apply control signals to the vehicle at least 10 times per second, with a 99-th percentile latency less than

[3]Alignment error is the difference between estimated transformation and ground truth transformation. We define this formally in §4.1.

Strategy	Overlap-agnostic	Data	Latency	Accuracy
VI-Eye [25]	✘	Raw PC	✘	✘
VIPS [51]	✘	Proc PC	✔	✘
VRF	✔	Raw PC	✔	✔

Table 2: A ✔ in the **Overlap-agnostic** means the strategy *does not* require overlap between the vehicle and RSU and vice versa. **Data** refers to the granularity of the shared data. Raw PC refers to raw point clouds, and Proc PC refers to processed point clouds. A ✔ in **Latency** means the strategy's latency is within 100 ms. A ✔ in **Accuracy** means accuracy is within 10 cm.

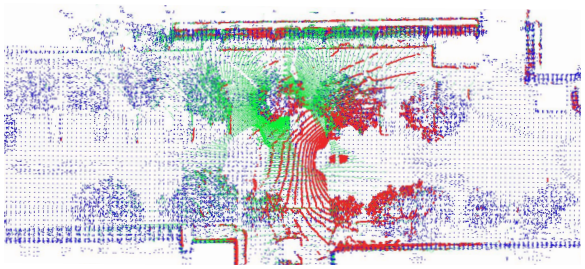


Figure 3: VRF aligns the RSU (green) and vehicle (red) point clouds to a pre-built 3D map (blue). This is a bird-eye view projection of VRF's deployment on a busy thoroughfare in the real world.

100 ms [11, 50]. As such, the RSU point cloud must be received and fused at the vehicle at very low latency, allowing enough time for the rest of the stack to process the fused point cloud. Moreover, the fusion must be highly accurate (on the order of a few cm), because downstream perception algorithms (*e.g.,* localization, object detection, tracking *etc.*) rely on the fused point cloud. Finally, in most cases, the vehicle and RSU point clouds may not have a large overlap. Thus, fusion must be overlap-agnostic, fast, and accurate. As Tbl. 2 shows, prior work does not meet all three requirements.

Our Approach. To this end, we present VRF, an end-to-end system that enables low latency point cloud fusion without sacrificing accuracy, even when the two point clouds have little or no overlap. As opposed to prior work that fuse the two point clouds directly, *our key insight in* VRF *is to de-couple the alignment into two steps.* More specifically, VRF aligns the vehicle and RSU point clouds separately to a 3D map (Fig. 3). 3D maps are dense 3D point clouds of the environment, captured from a vehicle's viewpoint, and are extensively used for localization [8] and path-planning [19] in autonomous driving [12, 18, 41, 54, 67] [4]. Once the two point clouds are aligned to the 3D map, *i.e.,* the transformation matrix for the vehicle is T_v, and the RSU LiDAR is T_r, we can transform the RSU point cloud to the vehicle's coordinate frame as $T_v^{-1} * T_r * P_r$, where P_r is the RSU point cloud.

Advantages of VRF. Indirectly aligning the vehicle and RSU point clouds to a 3D map has several advantages.

- *Low latency.* The RSU LiDAR is stationary, so VRF does not need to align it to the 3D map every frame. Instead, it can do so offline and re-use the transformation matrix.
- *High accuracy.* A 3D map is dense and has a large overlap with both the RSU, and vehicle point cloud. Thus, VRF can register both point clouds to the 3D map with high accuracy.

[4]These 3D maps are regularly updated to incorporate environmental changes [8] This is orthogonal to the focus of our paper.

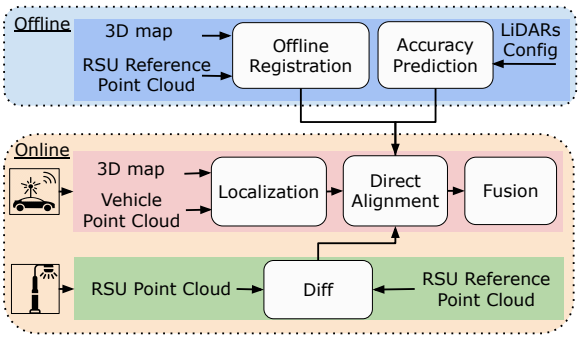


Figure 4: An overview of VRF.

• *Overlap-agnostic.* VRF aligns the vehicle and RSU point clouds to a 3D map. So, it can align the two point clouds even if there is little or no overlap between them.

Challenges. To do this, VRF must address three challenges.

• Although aligning vehicle point clouds to the 3D map is relatively easy (because they are captured from similar viewpoints), aligning the RSU point cloud is challenging. To accurately align the RSU point cloud, ICP requires an accurate initial guess of its transformation matrix with respect to the 3D map. However, it is not trivial to obtain this. Existing technologies and techniques (*e.g.,* GPS, and FGR [66]) cannot provide an accurate initial guess.

• The 3D map indirectly aligns the vehicle and RSU point clouds, but it can contain errors. These can be errors in the map itself, or they can arise whilst aligning the vehicle and RSU point clouds to the map. These errors are then propagated to the fused point cloud and can affect downstream perception algorithms. We observe that, in some cases, by directly aligning the vehicle and RSU point cloud (running a few quick iterations of ICP), we can significantly reduce these errors. However, in other cases, ICP can amplify these errors. Hence, it is important for VRF to determine when to run ICP. This is not trivial unless we know the ground truth.

• Even though VRF computes the transformation of the RSU LiDAR offline, other operations can incur significant latency. This includes the time to transmit the point cloud, localize the vehicle in the 3D map, perform direct alignment, and fuse the point clouds. This is undesirable. VRF must fuse the point clouds at low latency without sacrificing accuracy.

3 VRF DESIGN

Overview. To understand how VRF works, consider a vehicle passing through an intersection with an RSU mounted on a traffic light post (Fig. 4). The vehicle and RSU are equipped with a LiDAR, compute resources, and a wireless radio. In addition, they share a common 3D map. The RSU shares raw point clouds with the vehicle, which fuses them with its on-board point clouds to augment its perception. Unlike prior region-based sharing works (EMP [64] and Autocast [48]), VRF broadcasts the same RSU point cloud to all vehicles. We assume the vehicle and the RSU share a common 3D map of the area. This map is collected offline and does not contain dynamic objects (*e.g.,* vehicles, pedestrians).

Handshake. As the vehicle approaches the intersection, it initiates a handshake with the RSU to subscribe to its point clouds. During the handshake, the RSU sends the vehicle a reference RSU point cloud and the RSU LiDAR's pose in the shared 3D map. The reference RSU point cloud is a single frame from the RSU LiDAR and does not contain dynamic objects. It can be captured during RSU LiDAR installation. This handshake is relatively fast, and on average, takes only 12 ms (§4.9).

The RSU is aligned to the 3D map offline (§3.1).

Online Operations. To save network bandwidth, every frame, the RSU broadcasts the difference between the current point cloud and the reference RSU point cloud (§3.3). The vehicle adds this difference to the reference RSU point cloud that it received during the handshake process to reconstruct the RSU point cloud for the current frame. Simultaneously, the vehicle localizes in the 3D map using NDT [10]. At this stage, both the vehicle and RSU are aligned to the 3D map.

Directly aligning the two point clouds can further refine the alignment in some cases, and deteriorate it in others. To this end, VRF uses an alignment accuracy forecaster (§3.2) to predict if direct alignment will improve alignment accuracy. Then, the vehicle transforms the reconstructed RSU point cloud into its own coordinate system and fuses it.

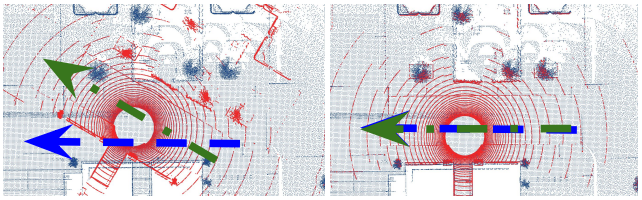
3.1 Offline Registration

The Problem. VRF's key insight is to de-couple the alignment and use two indirect alignments. That is, it fuses the RSU and vehicle point clouds by aligning them with a 3D map. The RSU is stationary, so VRF can align it with the 3D map offline and cache and re-use the transformation. However, this is not trivial. Feature-matching algorithms (SAC-IA [49], FGR [66]) are not robust to changes in viewpoints. Scan-matching algorithms (ICP [62]) need an accurate initial guess which is not straightforward to find. Although GPS can give the rough 3D location of the RSU, it cannot estimate orientation. We validate this in Tbl. 8.

Our Approach. VRF uses a novel algorithm that leverages plane matching and trajectory matching to find an accurate initial guess for the RSU's pose in the 3D map. ICP uses this initial guess as input to refine the alignment. The inputs to this algorithm (apart from the RSU point cloud and the 3D map) are: a) the rough GPS location of the RSU, and b) a short trajectory of the RSU installer's vehicle in both GPS coordinates and RSU LiDAR coordinates.

The rough GPS location need not be accurate (demonstrated in §4.7). It can easily be obtained by pinpointing the location of the RSU on a mapping service like Google Maps [3]. For the installer vehicle's trajectory, VRF only needs the vehicle's position across a small number of frames. The RSU installer vehicle's GPS trajectory can be obtained from an on-board smartphone. The trajectory of the vehicle in the RSU LiDAR can also be easily obtained by tracking its position across a small number of frames. This is a one-time process; it can be done when installing the RSU. The output of this algorithm is a 6-DoF pose of the RSU in the 3D map. VRF then feeds this 6-DoF transformation, along with the RSU point cloud and 3D map to ICP, which further refines the alignment.

First, VRF converts the GPS location of the RSU to the 3D map's coordinate system using the Mercator projection [52]. This determines the rough 2D location (x, y) of the RSU in the 3D map. Then, VRF uses RANSAC [21], a plane-fitting algorithm, to find the road surface in the RSU point cloud and the 3D map. RANSAC detects multiple planes, but because the road surface is the largest



(a) Misaligned trajectories **(b)** Aligned trajectories.

Figure 5: By aligning the vehicle's trajectory in GPS (blue) and RSU LiDAR coordinates (green), VRF computes the yaw angle.

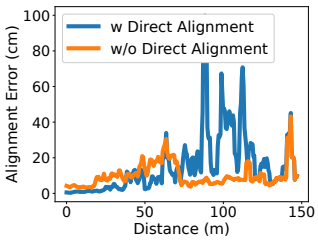


Figure 6: Alignment errors with and without direct alignment.

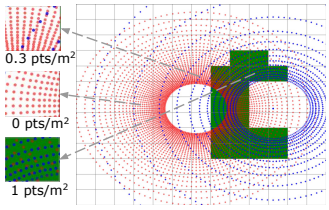


Figure 7: Inter-point density of vehicle (red), and RSU (blue) point clouds.

plane, VRF can identify it. Running RANSAC on the entire 3D map can be computationally expensive. So, VRF crops a square region around the rough 2D location of the RSU (x, y). Each side of the square region can be the maximum range of the RSU LiDAR *e.g.,* 120m for an Ouster-64 beam LiDAR [5]. VRF then runs RANSAC on this cropped map to find the road surface.

Then, VRF determines the height of the road surface at the center of the RSU point cloud (h_r). It also computes the height of the road surface in the 3D map at the rough 2D location of the RSU (h_m). Using these, VRF determines the z-coordinate of the RSU in the 3D map (h) as $h = h_m - h_r$. With this, VRF has the rough 3D position (x, y, z) of the RSU in the 3D map.

To find the pitch (β) and roll (γ), VRF aligns the normal vectors of the road surface in the RSU point cloud ($\overrightarrow{n_r}$) and the cropped map ($\overrightarrow{n_m}$). To find the yaw angle (α), VRF uses trajectory alignment. For this, VRF uses the trajectory of the vehicle in both GPS coordinates and the RSU LiDAR coordinates. It first converts the GPS coordinates of the vehicle to the 3D map's coordinate system using the Mercator projection. Then, VRF takes any two points of the vehicle trajectory in the map and RSU's point cloud. From these, it then calculates the yaw angle (α) which aligns the two trajectories (Fig. 5). Now, VRF has the three rotation angles of the RSU in the 3D map. VRF then combines R_α and $R_{\beta,\gamma}$ to get the final rotation matrix ($R_{\alpha,\beta,\gamma} = R_\alpha \cdot R_{\beta,\gamma}$).

VRF uses the 3D position and rotation angles to build a 6-DoF transformation T_{ig} of the RSU in the 3D map. Using this and the RSU point cloud and 3D map, ICP refines this transformation and outputs the final 6-DoF transformation T_r of the RSU in the 3D map. Because this process is offline, VRF can run ICP for a large number of iterations to get an accurate transformation. With this, VRF is able to accurately align the RSU point cloud to the 3D map. Our alignment is *orders of magnitude* better than simple GPS alignment or feature matching (Tbl. 8).

3.2 Alignment Accuracy Forecaster

On the vehicle, at every frame, VRF aligns the vehicle's point cloud to the 3D map using NDT [10]. Like ICP, NDT is reasonably accurate because (a) the overlap between the vehicle point cloud and the 3D map is high, and (b) we use the vehicle's pose from $n-1$ as an initial guess for NDT in frame n. At this point, both the vehicle and RSU point clouds are aligned to the 3D map, and indirectly to each other.

The Problem. The accuracy of this indirect alignment is dependent on the 3D map. Errors in the 3D map (*e.g.,* caused by changes in the environment [8]) or the vehicle's and RSU's ability to align with it, can propagate to the fused point cloud. To improve the alignment, we can use an additional direct alignment step. In this, we align the vehicle and RSU point clouds using their relative positions in the 3D map as an initial guess. However, this does not always guarantee improved alignment; it could worsen it in some cases as well.

To evaluate this, we drove a vehicle towards and away from the RSU LiDAR while capturing point clouds. For these point clouds, we evaluated VRF's alignment error with and without direct alignment on top of an indirect alignment (Fig. 6 plots error as a function of distance). In cases with a large overlap and an accurate initial guess, a direct alignment step (blue line) can significantly improve alignment accuracy. In other cases, where overlap between the point clouds is very low, it can reduce accuracy.

Key Insight. Given two point clouds, ICP iteratively finds the transformation that minimizes the 3D distance from every point in one point cloud to its closest point in the other point cloud. ICP's alignment accuracy depends on its ability to find and match the closest points of one point cloud to another. This, in turn, depends on a) the extent of overlap between the two point clouds, b) the point densities in the region of overlap, and c) the accuracy of the initial guess. The point density is the number of points per unit area for a given point cloud. We observe that *a large overlap and a high point density in the overlapping region* implies that ICP has a higher chance of finding accurate neighboring points (hence more accurate alignment), and vice versa.

Our Approach. With an analytical LiDAR model, given a LiDAR's parameters (*e.g.,* channels, FoV, range *etc.*) and its transform, VRF derives the 2D coordinates of every point the LiDAR's beams will hit on a flat surface. VRF uses this model to generate synthetic point clouds for the vehicle and RSU LiDARs (red dots represent RSU and blue dots represent vehicle point clouds in Fig. 7). Next, VRF divides the synthetic RSU point cloud into a grid. For each grid cell, VRF computes the *inter-point density (IPD)*. IPD is the *min* of the point density of the vehicle and RSU point clouds in that cell. We use the *min* to ensure that IPD is not biased by the point density of one of the point clouds. The left portion of Fig. 7 shows the IPD for three cells. Using *mean* point density instead can be biased towards a single point cloud, as the top two regions on the left of Fig. 7 indicate. Green cells in Fig. 7 represent regions with high IPD (*i.e.,* above a minimum inter-point density δ pts/m^2).

A higher IPD at a given cell means that direct alignment (ICP) has a higher chance of finding accurate neighboring points in that cell, and vice versa. Determining whether to run ICP based on a single cell's IPD is not robust. So, VRF considers IPD across the entire grid. That is, for a given scenario, if many cells have high IPD, it indicates a higher chance of accurate direct alignment, and vice versa. Fig. 8 and Fig. 9 illustrate IPD distributions for accurate and inaccurate direct alignments. Fig. 8 demonstrates a scenario when

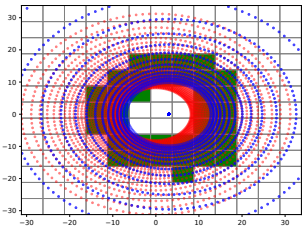


Figure 8: IPD with a high chance of accurate direct alignment.

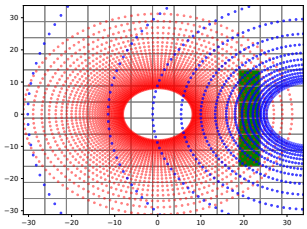


Figure 9: IPD with a low chance of accurate direct alignment.

the vehicle and RSU are in proximity, hence a higher number of cells with high IPD (shown in green). On the other hand, Fig. 9 shows a scenario when the vehicle and RSU are far away, hence a low number of cells with high IPD (shown in green). In our implementation of VRF, if 10% of the cells have high IPD, we run direct alignment[5].

Alternate Approach. Instead of using synthetic point clouds, we could have used the actual point clouds, as they capture the environment better. However, this incurs additional latency and is not needed. We prove in §4.7 that even with synthetic point clouds we get comparable accuracy.

Deployment Optimization. In practice, VRF will know the LiDAR parameters of all RSU LiDARs for a given area, and the vehicle LiDAR. Offline, for every RSU LiDAR, VRF will compute the *inter-point density* for a wide range of distances and use that to forecast whether direct alignment will help. The vehicle will store these results as a lookup table. Then, given a RSU point cloud and the vehicle's transform in the 3D map, it uses the lookup table to determine whether it should run direct alignment.

3.3 Fast Fusion Pipeline

The Problem. Although VRF runs RSU to 3D map alignment and the alignment accuracy forecaster offline, it can still incur significant latency for other operations. Without compromising accuracy, VRF must fuse point clouds at low latency to give enough time for downstream perception tasks to consume the fused point cloud. To understand why VRF can be slow, we describe a strawman pipeline for VRF (Fig. 10).

The vehicle node localizes the vehicle point cloud (P_v) in the 3D map (Fig. 10: NDT). Then, it waits for the RSU point cloud (P_r). The RSU compresses the RSU point cloud and sends it over the wireless network to the vehicle (Fig. 10: Comp). The vehicle node receives this and extracts the vehicle point cloud from it (Fig. 10: Extract). If needed, it runs an additional direct alignment step (Fig. 10: ICP). Finally, the vehicle node fuses the two point clouds (Fig. 10: Fusion). End-to-end, this pipeline can be slow. On modest hardware, the average end-to-end latency for the strawman pipeline is 40 ms whereas 99^{th} percentile is 60 ms.

Our Approach. To fuse point clouds at low latency, without compromising on accuracy, VRF uses a set of optimizations (Fig. 11). We describe these as follows.

Diff Clouds. 3D point clouds are voluminous and can exhaust a wireless network's bandwidth. For instance, raw point clouds from an Ouster-64 LiDAR require a wireless bandwidth of as much as 480 Mbps. Instead of sending a raw point cloud, the RSU sends a

[5]Through exhaustive simulations with multiple LiDAR models, we find δ to be 1 pt/m^2.

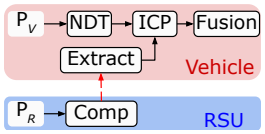


Figure 10: Strawman pipeline for VRF.

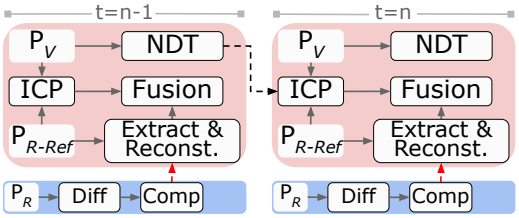
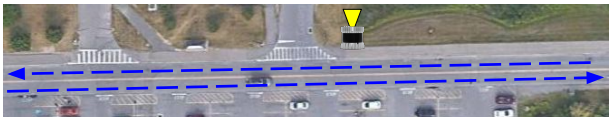


Figure 11: Optimized pipeline. Vehicle (red) and RSU (blue) ops.

diff cloud to the vehicle. A *diff cloud* is the difference between the current RSU point cloud (P_R), and the reference RSU point cloud (P_{R-Ref}). The reference RSU point cloud is captured during the RSU's installation when there are no dynamic objects in the scene (§3). At the RSU, VRF calculates the *diff cloud* by subtracting the P_R from the P_{R-Ref}. To calculate the diff, VRF builds a double-buffer Octree [27] that contains both RSU point clouds. Then, it uses XOR operations to detect nodes that are different between the point clouds. It stores the points from these nodes in a *diff cloud*, compresses it and sends it over the wireless network. At the vehicle, VRF decompresses and adds the *diff cloud* to the P_{R-Ref} to reconstruct P_R (Fig. 11: Extract & Reconst).

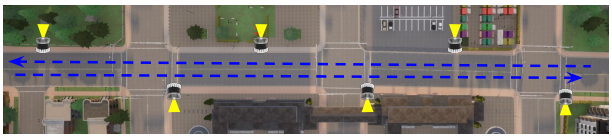
Alignment with the Reference RSU Point Cloud. In the strawman pipeline (Fig. 10), ICP must wait for the vehicle to receive, and reconstruct the RSU point cloud. Once that happens, then ICP aligns it with the vehicle point cloud. VRF removes this bottleneck by parallelizing ICP with wireless transmission of the RSU point cloud. To do this, instead of waiting for the diff cloud, VRF *aligns the RSU's reference point cloud (P_{R-Ref}) with the current vehicle point cloud (P_V)* (Fig. 11). Because the vehicle has already received the RSU's reference point cloud during the handshake, it can run ICP as soon as a new vehicle point cloud is available. This way, ICP computes the transformation matrix for the RSU reference point cloud. Then, when the vehicle receives and reconstructs the RSU's current point cloud, it simply applies the RSU reference point cloud transformation to it. We can use the RSU's reference point cloud for alignment because the RSU is stationary. Even though the vehicle point cloud has dynamic objects, ICP can use common stationary objects (*e.g.,* roads, curbs, buildings, traffic signs, *etc.*) to compute the transformation. Empirically, we observe from real-world and synthetic traces, a significant portion of the vehicle point cloud is made of stationary objects, even when driving in high traffic. With this, VRF parallelizes alignment with the wireless transmission of the *diff cloud*.

Initial Guess from Previous Frame. The direct alignment step using ICP needs an initial guess to quickly converge to the correct transformation. This initial guess is the transformation of the vehicle and RSU in the 3D map. Because the RSU is stationary, VRF uses its cached transformation as computed in §3.1. The vehicle can use NDT's estimated transform for the current frame, but for that, VRF must wait for NDT to complete before it can run ICP. To alleviate



(a) On-campus deployment with a single RSU

(b) Off-campus deployment with two RSUs



(c) CarLA scenario with 6 RSUs

Figure 12: VRF deployments in the real world and simulation with RSUs in yellow and vehicle trajectory in blue.

this bottleneck, VRF *extrapolates the vehicle's transform from the previous frame as an initial guess for ICP in the current frame (Fig. 11)*. For extrapolation, it computes a motion vector by simply subtracting the vehicle's transform in the previous two frames. Then, it uses that as an initial guess for ICP in frame n. With this, VRF can quickly compute an initial guess for ICP.

GPU Offloading. We run both NDT and ICP on the GPU.

4 EVALUATION

4.1 Methodology

Implementation. We implemented VRF in the Robot Operating System (ROS). Our implementation consists of three C++ ROS nodes: a) the RSU node, b) the NDT node, and c) the Fusion node. The RSU node resides at the RSU. It computes *diff clouds* from the RSU point cloud and broadcasts them over the wireless network. The NDT and Fusion nodes run at the vehicle. The NDT node localizes the vehicle in the 3D map. The Fusion node listens for vehicle point clouds and *diff clouds*. It reconstructs the RSU point cloud, aligns it with the vehicle point clouds, stitches them, and then transforms the fused point cloud into the vehicle's coordinate system. We have implemented offline registration and alignment accuracy as separate nodes that run offline. The external libraries we have used include PCL (Point Cloud Library) [6], and ROS [7]. To build 3D maps, we use FAST-LIO2 [57].

Real-world Testbed. We built our own testbed for VRF (Fig. 12 and Fig. 13) consisting of a vehicle and one or multiple RSUs. In our experiments, we used four different types of LiDARs. Two of them are 128-beam LiDARs (OS0-128 and OS1-128) with 90° and 45° field-of-views (FoV). The other two are 64-beam LiDARs (OS0-64 and OS1-64) with 90° and 45° FoVs. We mounted the RSU LiDARs 3-5 m from the ground. On-board the vehicle, we had a laptop with an Intel i7 CPU, 32 GB RAM, and an NVIDIA GeForce RTX 3060 GPU. At the RSUs, we used a laptop with an Intel i9 CPU, 16 GB RAM, and an NVIDIA GeForce RTX 3070 Ti GPU. Vehicle and RSUs used an ASUS dual band AX6000 router to communicate with each other over Wi-Fi 802.11ax at 5 GHz.

We deployed this testbed at two different locations *i.e., on-campus* (Fig. 12a) on a two-way street adjacent to a parking lot, and *off-campus* (Fig. 12b) on a busy two-way public road. From these deployments, we also record LiDAR traces for accuracy evaluations. To generate ground truth, we manually align the vehicle and RSU point clouds using CloudCompare [2]. In the *on-campus* deployment, we mounted OS1-128 LiDARs on both the vehicle and the RSU. The *off-campus* deployment had two RSU nodes with OS1-64 and OS1-128 LiDARs and a vehicle node with an OS0-64 LiDAR. These LiDAR traces consist of 63,512 point clouds in which the vehicle drove for around 12 km. VRF's code and dataset are open-source[6].

Simulation. We use CarLA [20], an industry-standard photo-realistic simulator, to capture additional LiDAR traces for our experiments. Compared to the real-world dataset, with CarLA, we can simulate diverse traffic conditions, multiple LiDAR settings, and different traffic scenarios. We capture LiDAR traces from CarLA and then process them offline using the same compute resources as the real-world setup. These LiDAR traces consist of 20,000 point clouds in which the vehicle drove for 2 km.

Evaluation Metrics. In our evaluations, we measure the latency and accuracy of VRF. Latency is the time duration from when a RSU LiDAR captures a 3D point cloud to when the vehicle fuses it with its own point cloud. This latency includes both network latency, and compute latency.

For accuracy, we use the relative translational error (RTE) and relative rotational error (RRE) as defined in the KITTI benchmark [22]. Both measure the root mean square error between the predicted and ground truth transformations. For both, lower is better. RTE is measured as $||t - t_g||_2$ where t is the translational vector for VRF and t_g for the ground truth transformation. RRE is measured as $\Sigma_{i=1}^{3} |\ angle\ (i)\ |$. The angle is $F\ (R_g^{-1} \cdot R)$ where $F(.)$ transforms a given rotation matrix into three Euler angles. R_g and R are the ground truth and the estimated transformations, respectively.

4.2 End-to-end Experiments: Performance

Real-world Deployment. To evaluate performance in real-world deployments, we connected the RSU nodes to the vehicle node through Wi-Fi (to emulate 5G). We drove a vehicle back and forth (blue trajectory in Fig. 12a and Fig. 12b) as it received and fused point clouds from the RSUs in both deployments. In these experiments, we measured end-to-end latency. During our experiments, there was heavy pedestrian and vehicular traffic on the road.

Fig. 14 plots the end-to-end latency (both compute and network) as a function of time for the real-world deployments with three different combinations of vehicle and RSU LiDARs for 9-minute drives. In all three scenarios, the average latency was within 20 ms and the 99^{th} percentile (p99) was within 34 ms. These numbers are well within the 100 ms latency budget for autonomous driving [11, 50] and give ample time (80 ms on average) to downstream perception modules to process the fused point clouds. *This demonstrates that* VRF *can fuse raw vehicle and RSU point clouds in approximately 20 ms, even for data-intensive 128-beam LiDARs!*

CarLA Traces. In CarLA, we mounted six 64-beam LiDARs on a road and drove a vehicle with a 64-beam LiDAR through it as

[6]GitHub Repository: https://github.com/nsslofficial/VRF

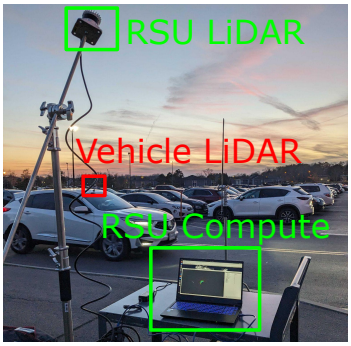


Figure 13: VRF deployment setup.

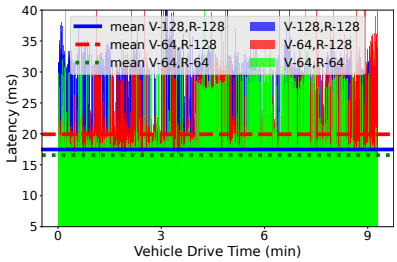


Figure 14: VRF latency on our real world testbed with different LiDAR combinations.

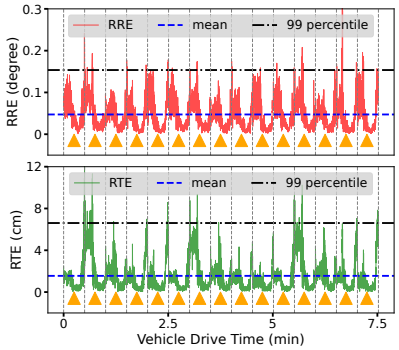


Figure 15: VRF's accuracy on CarLA dataset.

Dataset	RTE (cm)			RRE (°)		
	Mean	*p95*	*p99*	*Mean*	*p95*	*p99*
On-campus (913 pairs)	4.9	15.8	22.3	0.1	0.2	0.4
Off-campus (420 pairs)	6.9	16.0	21.7	0.4	0.8	1.5
CarLA (4500 pairs)	1.6	5.0	6.6	0.05	0.1	0.15

Table 3: VRF end-to-end accuracy across multiple datasets.

shown in Fig. 12c. In our simulations, there were approximately 10-15 vehicles near the vehicle node at all times. We collected RSU and vehicle LiDAR traces from CarLA, and then processed them offline.

We replayed both traces on our vehicle and RSU nodes in a lab setup and measured the end-to-end latency. In this setup, we used Wi-Fi for communication. On average, the end-to-end latency was 12.2 ms whereas the 99^{th} percentile latency was 27.7 ms (we omit the graph for brevity). The results reinforce those from the real-world deployment *i.e.,* VRF fuses raw vehicle and RSU point clouds with minimal latency, well within the 100 ms latency budget.

4.3 End-to-end Experiments: Accuracy

Using the same setup described in the previous experiments, we evaluated VRF's accuracy using RTE and RRE.

Real-world Deployment. In our real-world deployments, we recorded vehicle and RSU point clouds in ROS bags. Then, for point cloud pairs with a reasonable amount of overlap, we annotated and registered them offline using CloudCompare [2]. We manually registered over 1300 point cloud pairs for accuracy evaluations. Of these, 420 point cloud pairs were from our on-campus dataset, and 913 point cloud pairs were from the off-campus dataset.

For point clouds with a lower overlap, we could not generate ground truth and leave those evaluations for CarLA. The average RTE and RRE for the off-campus dataset were 6.9 cm and 0.4°, and those for the on-campus dataset were 4.9 cm and 0.1° (Tbl. 3). Besides mean errors, the p95 and p99 RTE errors were also within 16 cm and 22 cm, respectively. *This demonstrates that* VRF *can ensure cm-level accuracy for point cloud fusion, a requirement of autonomous driving systems, and do so within 20 ms of end-to-end latency!*

CarLA. Because it is easier to generate ground truth in simulations, we more thoroughly evaluated VRF in CarLA. Fig. 15 plots VRF's RTE and RRE as a vehicle drove back and forth in the CarLA scenario (Fig. 12c). While driving, the vehicle would switch from one RSU to another. The vertical dotted lines in Fig. 15 delineate

Approach	Easy group			Hard group		
	RTE (cm)	RRE (°)	Latency (ms)	RTE (cm)	RRE (°)	Latency (ms)
VI-Eye	14.65	2.01	223	16.69	2.08	218
VRF	*1.49*	*0.06*	*16.3*	*1.49*	*0.08*	*15.5*

Table 4: Evaluating VRF on the VI-Eye [25] dataset

these coverage regions with the yellow triangles showing the position of the RSU. The average RTE for over 4500 point cloud pairs was 1.6 cm whereas RRE was 0.05°. The 99^{th} percentile numbers were also relatively low *i.e.,* 6.6 cm and 0.15°. The errors are lowest at the center of the coverage region when the vehicle is nearest the RSU. At this stage, the overlap between the vehicle and RSU point clouds was the highest, and direct alignment significantly improved the previous indirect alignment. As the vehicle moved away from the RSU, the overlap decreased, and hence RTE/RRE would increase. *This demonstrates that* VRF *can accurately fuse vehicle, and RSU point clouds with cm-level accuracy, even when the vehicle is away from the RSU.*

4.4 Comparison with Prior Work

We compared VRF with VI-Eye [25], a state-of-the-art vehicle RSU point cloud fusion system. For a more than fair comparison, we evaluated alignment accuracy and end-to-end latency on VI-Eye's dataset. This dataset contains point clouds captured from a vehicle and multiple RSUs spanning a length of 1.12 km. The point clouds were captured with Livox Horizon LiDARs [4] with a vertical and horizontal FoV of 25° and 81°, respectively. Because the LiDAR has a limited horizontal FoV, point cloud pairs in which the vehicle was traveling opposite to the direction of the RSU LiDAR are labeled as *hard group* (because the overlap is low), and others are labeled as *easy group*.

In terms of accuracy, compared to VI-Eye, VRF *reduces RTE by an order of magnitude, and RRE by two orders of magnitude for easy and hard groups (Tbl. 4).* VI-Eye matches road-specific features (*e.g.,* lane markers, traffic signs) to align point clouds. As opposed to feature-matching, scan-matching techniques (ICP) are more accurate but can be expensive. VRF uses scan-matching but reduces computational complexity by a) indirect alignment with a 3D map, b) offline RSU registration, and c) offline alignment accuracy forecasting. Then, every frame, it can run an additional lightweight direct alignment to further reduce alignment error without increasing latency.

We ran VI-Eye and VRF on the same vehicle and RSU nodes and measured their end-to-end latency (including network transmission

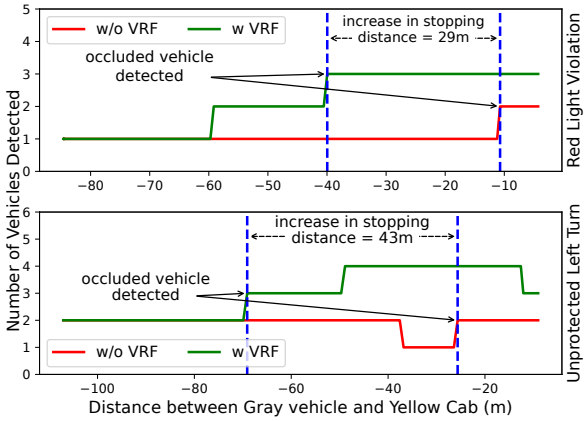


Figure 16: VRF improves a vehicle's reaction time.

time). Relative to VI-Eye, VRF *reduces latency by an order of magnitude!* For VI-Eye, the latency was 223 ms, whereas for VRF, it was only 16.3 ms.

4.5 Application-level Benefits

In this section, we demonstrate the efficacy of VRF in improving the performance of autonomous driving applications.

Reaction Time. Fig. 1 shows two scenarios from the NHTSA pre-crash topology [14] where occlusions can result in a traffic accident. For these scenarios, we define *reaction time* as the time duration from when the grey vehicle first sees the yellow cab to when the distance between their colliders is effectively 0 (*i.e.,* they crash). A higher *reaction time* indicates the grey vehicle can perceive the yellow cab early on, and hence take evasive action. To demonstrate that VRF can improve *reaction time*, we simulate both scenarios in CarLA. In doing this, we install an RSU at the intersection. We attach 64-beam LiDARs to the RSU and the grey vehicle. We collect LiDAR traces and process them offline.

From these LiDAR traces, we build two sets of point clouds *i.e.,* un-fused vehicle point clouds and fused point clouds. We obtain the fused point clouds by running VRF on the LiDAR traces. The un-fused vehicle point clouds contain point clouds captured from the grey vehicle. On both sets of point clouds, we run an object detector. This detector takes a point cloud and performs background subtraction against a 3D map to find dynamic points and then groups them into objects using Euclidean clustering. Fig. 16 plots the number of detections from this detector as the grey vehicle drives towards the intersection (with the distance between the vehicle and RSU on the x-axis). The number of detections in fused and un-fused point clouds are green and red, respectively.

In these scenarios, the vehicle point clouds (red) first observe the yellow cab at a distance of 11 m and 27 m in red light violation and unprotected left turn scenario, respectively. VRF (green), on the other hand, observes the yellow cab at distances of 40 m, and 70 m. In these scenarios, for a vehicle traveling at 30 km/hr, VRF *improves reaction time from 1 second to 5 seconds (*i.e., *a factor of 5).*

Point Density and Coverage Volume. Downstream perception modules like object detection [60] and semantic segmentation [40] need point clouds with a high point density. Point density is the number of points per unit volume. The blue line in Fig. 17 represents the

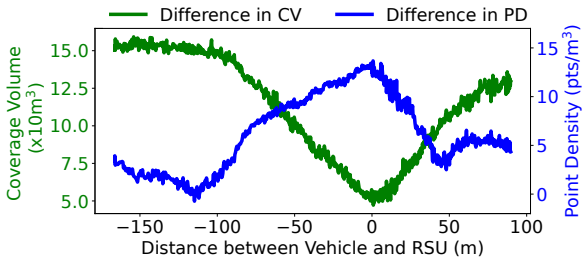


Figure 17: VRF's improves point density and coverage volume relative to an un-fused vehicle point cloud. Both y-axes depict raw numbers (not percentages).

difference in the point density (right y-axis) for VRF's fused point cloud and a baseline un-fused cloud *i.e.,* single vehicle perception. VRF improves point density in all cases. Even when the vehicle is 100 m away from the RSU, VRF improves point density by as much as 3 pts/m^3. As expected, the increase in point density is very significant (14 pts/m^3) when the vehicle is near the RSU.

To demonstrate that VRF improves point density over the entire point cloud, we measure coverage volume. It is that region of the point cloud where point density is high. To calculate coverage volume, we divide a point cloud into a grid, compute point density for each cell, and find the volume of cells where point density is above δ (in this case, 8 pts/m^3). The green line in Fig. 17 plots the difference between coverage volume (left y-axis) for VRF's fused point cloud and a baseline un-fused point cloud. VRF improves coverage volume significantly in all cases, especially when the overlap between the point clouds is small. The increase in coverage volume is low when the vehicle is near the RSU because both point clouds capture the same spatial area.

4.6 VRF Pipeline Optimizations

In this section, we quantify the contribution of each optimization in VRF's fusion pipeline in reducing end-to-end latency (without compromising accuracy). To do this, we compare VRF against three other pipelines (Tbl. 5) on 1500 point cloud pairs. The first row represents the strawman pipeline (Fig. 10) with no optimizations enabled. The last row represents VRF (Fig. 11) with all optimizations enabled. The three columns (*Diff*, *Reference Cloud*, and *NDT*) show the effects of the three optimizations described in §3.3.

The strawman pipeline (a serialized design) has an average latency of 39 ms. VRF parallelizes most of these components to reduce the end-to-end latency by a factor of 3 *i.e.,* from 39 ms to 12.5 ms, without sacrificing accuracy.

Point clouds are voluminous and transmitting them over the network is expensive. By sending diff clouds (*Diff*), the RSU can reduce the point cloud sizes by 56 x. This reduces network latency, and hence end-to-end latency by a factor of 2. This comes at the cost of an increase of 0.03 cm in RTE.

By using the reference RSU point cloud for direct alignment (*Reference Cloud* in Tbl. 5), the vehicle runs ICP in parallel while it waits to receive the diff cloud. However, this reduces end-to-end latency slightly *i.e.,* by 1.02 ms. In addition to the two-point clouds, ICP also needs an initial guess, which it computes using the vehicle's position. For this, it is blocked until the vehicle runs NDT. To alleviate this bottleneck, VRF computes motion vectors for the vehicle and then uses them to extrapolate the vehicle's position from

Optimization			Latency (ms)	Alignment Error	
Diff	*Ref Cloud*	*NDT*		*RTE (cm)*	*RRE (°)*
✘	✘	✘	39.28	1.75	0.05
✔	✘	✘	19.84	1.78	0.05
✔	✔	✘	18.82	1.65	0.05
✔	✔	✔	12.55	1.44	0.05

Table 5: Optimizations in VRF's fusion pipeline. A ✔ indicates the optimization is enabled, and ✘ indicates it is disabled.

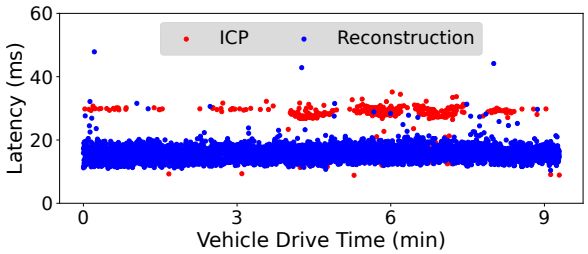


Figure 18: Latency distribution of VRF on a real-world dataset.

Error Source			Alignment Error	
Mapping	*Localization*	*Registration*	*RTE (cm)*	*RRE (°)*
✔	✔	✔	1.49	0.05
✘	✔	✔	1.29	0.04
✘	✘	✔	1.26	0.04
✘	✘	✘	0	0

Table 6: Breakdown of errors in the VRF's end-to-end alignment.

the last frame into the current frame. This way, VRF's vehicle node can run ICP as soon as a new vehicle point cloud is available. As soon as ICP converges, it applies that transformation matrix to the received RSU point cloud. This way, ICP and network transmission run in parallel.

Fig. 18 plots the end-to-end latency for VRF. The red dots show frames where the RSU point cloud is received and reconstructed at the vehicle but VRF waits for the ICP to estimate a transformation matrix. The blue dots show frames where ICP has already estimated a transformation matrix but the vehicle has not received and reconstructed the RSU point cloud. This shows that, on average, VRF's bottleneck is the network transmission of the RSU point cloud.

Using the reference point cloud as opposed to the current RSU point cloud for ICP actually improves alignment results (by 0.13 cm). This is because, in practice, the clocks of the RSU LiDAR and vehicle LiDAR can be off in a range of 0 to 50 ms. In a scene with dynamic objects, their positions in one point cloud might be captured earlier as compared to the other point cloud. As such, the alignment results can be slightly off. By using a reference point cloud, with no dynamic objects, ICP aligns only the static regions which do not change over short timescales.

4.7 Ablation Studies

Alignment Error Sources. We quantify external sources of error that contribute to the end-to-end alignment error (Tbl. 6). The three sources of error are mapping error, localization error, and offline registration. Mapping error represents an inherent error present in the 3D map. Localization error is the error in estimating the pose of the vehicle. Offline registration is the error in estimating the pose of the RSU LiDAR.

Approach	RTE (cm)	RRE (°)
ICP	858	16.37
NDT	329	30.67
VRF	2.25	0.02

Table 7: VRF's ability to align the RSU to the 3D map.

Approach	RTE (cm)	RRE (°)	Latency (ms)
GPS	618	42.97	364
FGR	3896	339	242
SAC-IA	4832	130	224
VRF	2.41	0.04	129

Table 8: VRF's ability to generate accurate initial guesses.

The first row represents VRF with all three sources of error. When VRF uses a perfect ground truth map (second row), it significantly reduces RTE and RRE because both vehicle localization and offline registration use the 3D map. However, even with a perfect ground truth map, estimating the pose of the vehicle and RSU LiDARs can have errors (second and third rows). When the vehicle uses ground truth localization, the only source of error is offline registration (third row). In a perfect world, with a ground truth 3D map and perfect RSU and vehicle localization, the RTE and RRE will be 0.

Offline Registration. In this section, we quantify the ability of VRF to align the RSU point cloud to the 3D map. Scan matching algorithms like ICP and NDT can align point clouds if they have an accurate initial guess of the transformation between the point clouds. Without this, ICP and NDT cannot align the RSU point cloud to the 3D map (Tbl. 7). VRF, on the other hand, generates an accurate initial guess (§3.1) and then uses ICP to finely align the RSU point cloud to the 3D map. By doing so, it reduces RTE by two, and RRE by three orders of magnitude as compared to ICP.

We evaluated the ability of VRF to generate an accurate initial guess for ICP against three other baselines: GPS, FGR [66], and SAC-IA [49] (Tbl. 8). We generated initial guesses using these three techniques, fed them to ICP, and then measured the alignment accuracy. GPS is reasonably accurate (up to 1-10 m), but it has no way to estimate the RSU LiDAR's orientation, hence cannot guarantee accurate alignment.. FGR, and SAC-IA extract and match features from point clouds. However, the RSU point cloud is captured from a different viewpoint as compared to the 3D map. As such, FGR and SAC-IA cannot find and match common features between the two point clouds, leading to an inaccurate initial guess and hence inaccurate alignment. Conversely, VRF uses viewpoint-agnostic techniques (*i.e.,* plane-matching and trajectory-matching) on raw point clouds and can align the RSU point cloud to the 3D map accurately and quickly.

Alignment Accuracy Forecaster. In this section, we evaluate VRF's ability to forecast whether direct alignment will improve the alignment between the RSU and vehicle point clouds. To do this, we compare VRF against four other techniques. Direct alignment (DA) runs ICP every frame. Indirect alignment (IA) does not run ICP and uses relative poses from the 3D map. Online forecasting (OF) determines whether to run direct alignment every frame by using the *inter-point density* of current point clouds.

We collected multiple datasets from CarLA for the same route using different combinations of LiDARs. The x-axis of Fig. 19 shows the channel combination of the vehicle and RSU LiDAR. That is,

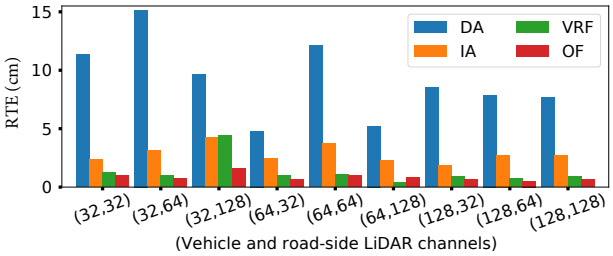


Figure 19: Alignment Accuracy Forecaster.

Traffic	Alignment Error		Latency (ms)
	RTE (cm)	*RRE (°)*	
Low (6 – 10 vehicles)	1.42	0.05	11.85
Medium (10 – 15 vehicles)	1.64	0.05	12.11
High (15 – 20 vehicles)	1.55	0.05	12.61

Table 9: Performance of VRF in various traffic conditions

(32, 32) means both RSU and vehicle LiDARs had 32 beams. For each combination, we calculated the alignment accuracy (RTE) of each approach for a range of distances from 0 to 180 m between the RSU and the vehicle node. The y-axis of Fig. 19 represents the RTE of each scheme relative to an *ideal* scheme. The *ideal* scheme is the theoretical upper limit of the forecaster's accuracy *i.e.,* it uses ground truth to determine when to use ICP.

As evident from Fig. 19, for all LiDAR combinations, relative to direct alignment and indirect alignment, VRF reduces RTE by 7.65 cm and 1.32 cm, respectively. ICP can only improve accuracy if there is a significant overlap between the two point clouds, otherwise, it degrades alignment accuracy. Because VRF runs *inter-point density* calculations on forecasted point cloud distributions for a range of distances, it can accurately predict when ICP will correctly align two point clouds.

Because online alignment uses point clouds from the current frame instead of synthetically generated ones, it can more accurately forecast ICP behavior and hence reduce RTE. However, it does so only marginally and comes at the cost of latency *i.e.,* online alignment takes 40 ms to execute per frame. Compared to the non-realistic ideal case, VRF increases RTE by only 1.5 cm. By trading off 1.5 cm in RTE, VRF eliminates the need to run alignment forecasting every frame (effectively reducing latency to 0 ms).

4.8 Sensitivity Analysis

Traffic Conditions. In CarLA, we recorded LiDAR traces in three different traffic density conditions (Tbl. 9). We ran VRF on each dataset and measured the end-to-end accuracy and latency. For this experiment, we ran the ROS bags on the vehicle node, and RSU node in our lab, using Wi-Fi for communication. As Tbl. 9 indicates, irrespective of the traffic conditions, VRF accurately fused RSU and vehicle point clouds within 13 ms with less than 2 cm error.

Vehicle Speed. To evaluate the effect of vehicle speed on VRF, we collected multiple traces from CarLA by varying a vehicle's speed from 10 km/hr to 80 km/hr as it crossed an intersection. Then, offline, we ran VRF on each trace and measured both the end-to-end latency and accuracy. Like before, for this experiment, we ran the ROS bags on vehicle and ROS nodes in our lab, using Wi-Fi for communication. Experimental results (Fig. 20 and Fig. 21) show

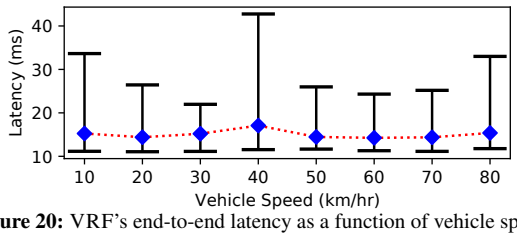


Figure 20: VRF's end-to-end latency as a function of vehicle speed.

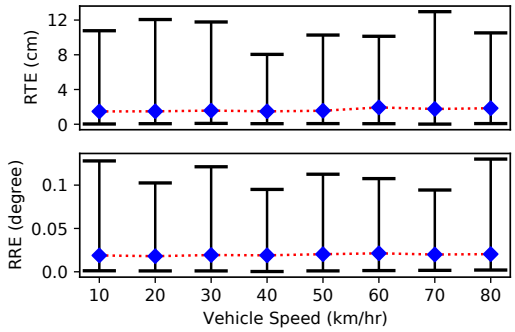


Figure 21: VRF's accuracy as a function of vehicle speed.

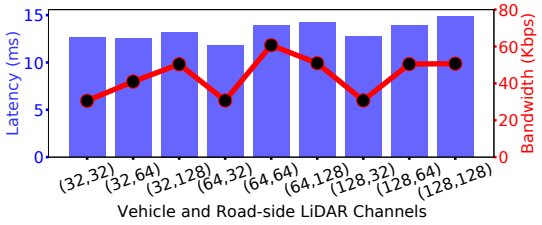


Figure 22: Latency and network BW for different LiDAR settings.

VRF's accuracy and latency are relatively insensitive to the vehicle's speed. Even at speeds as high as 80 km/hr, VRF's end-to-end latency was 15.5 ms, whereas alignment accuracy was 1.8 cm and 0.02°, respectively. Across all traces, the average end-to-end latency was only 15 ms, and alignment error was only 1.6 cm and 0.02°.

Heterogeneous Sensors. We evaluated VRF on multiple RSU and vehicle LiDAR configurations and measured the accuracy and latency. As in the previous experiment, we collected data from CarLA and then replayed those ROS bags using Wi-Fi for communication in our lab. Fig. 22 shows the end-to-end latency and bandwidth requirement for each LiDAR combination. Irrespective of the LiDAR combination, VRF's latency is within 13 ms. As expected, as the number of channels for the RSU LiDAR increases, so does the required bandwidth. Even then, the required bandwidth is only in Kbps.

4.9 Offline Operations and Handshake Overhead

In this section, we quantify the overheads of VRF's vehicle-RSU handshake, its offline operations, and on-board vehicle storage.

Handshake. As part of the handshake, the RSU sends the vehicle a compressed reference point cloud. To evaluate the overhead of this, we collected 200 traces from CarLA simulating vehicle-RSU handshakes. Our evaluations show that, on average, the size of the

compressed reference point cloud is 500 KB, and it takes only 12 ms to send to the vehicle.

Offline Operations. The compute latency for aligning the RSU to the 3D map (including plane alignment, trajectory alignment, and ICP) is approximately 7 seconds. Because the RSU is stationary, VRF aligns it with the 3D map offline, and then reuses the alignment every frame.

On-board Storage Requirements. Offline VRF runs the alignment accuracy forecaster to determine, for some given vehicle and RSU LiDAR parameters, the maximum vehicle-RSU distance below which to run additional alignment and then stores these on-board the vehicle as a lookup table. For large metropolitan cities like San Francisco (6,399 intersections) and New York City (13,543 intersections), the lookup tables will only take 77 KB and 163 KB of on-board vehicle storage, respectively. In addition, VRF assumes the presence of an on-board 3D map. The size of three maps we built for our *on-campus*, *off-campus*, and CarLA datasets were 6 MB, 3 MB, and 2 MB, respectively.

5 RELATED WORK

Cooperative perception addresses sensor occlusions by using external sensors to extend a vehicle's perception range beyond its own sensors. These external sensors might be on-board other vehicles [32, 38, 44, 59], or they might be mounted on road-side infrastructure [24, 25, 51]. Of these works, AVR [47] shares raw 3D point clouds between two vehicles and aligns the point clouds using a 3D map. However, sharing raw 3D sensors consumes significant network bandwidth and cannot scale to a large number of vehicles. To this end, region-based data-sharing approaches like AutoCast [48] and EMP [64] determine what data is relevant for a given vehicle, and only share that over the network, scaling to a larger number of vehicles. Additionally, the clocks of 3D sensors on different vehicles might be out-of-sync. RAO [63] tackles this by sending point clouds that are motion-compensated using occupancy flow prediction. Changes in the environment can render 3D maps stale, and this can adversely affect cooperative perception accuracy. To this end, CarMap [8] uses crowdsourced real-time map updates to incorporate environmental changes in 3D map. Similarly, VI-Map [23] uses road-side LiDARs to update 3D maps.

Like VRF other works have also explored using road-side infrastructure for vehicle-RSU cooperative perception [9, 16, 24, 25, 31, 36, 39, 51, 64, 65]. Perhaps the most relevant to VRF is VI-Eye [25] which uses early fusion to fuse a vehicle point cloud with raw RSU point clouds. However, like other early fusion approaches, the end-to-end latency is significantly high *i.e.,* on the order of 200 ms. Late fusion approaches like VIPS [51] instead fuse bounding boxes to reduce end-to-end latency. However, this comes at the cost of alignment accuracy.

On the other hand, intermediate fusion approaches (TransIFF [17], V2X-ViT [56], DiscoGraph [34], and F-Cooper [16]) use end-to-end neural networks to extract point cloud features, transmit them and then fuse them as bounding boxes. Both intermediate and late fusion approaches lack generality to other downstream modules. To our knowledge, VRF is the first system that ensures both high accuracy and low latency whilst enabling generality to downstream modules.

ICP [62] and NDT [10] are classical scan-matching methods for point cloud alignment. ICP uses a pair-wise alignment, while NDT employs a probabilistic approach to estimate the transformation between point clouds. These methods are compute-intensive and their accuracy heavily depends on an accurate initial guess of the transformation. Feature-based methods like SAC-IA [49] and fast global registration (FGR) [66] find an accurate initial guess of the transformation. These methods perform noticeably worse in cooperative perception because of the repeating structures in traffic scenarios. VRF uses raw scan matching for alignment and reduces latency by running computations offline.

6 DISCUSSION AND FUTURE WORK

V2V Cooperative Perception. Though in this paper we focus on vehicle-RSU point cloud fusion, VRF can also be used for vehicle-to-vehicle (V2V) point cloud fusion. In that case, both vehicles will localize themselves in the 3D map online and improve fusion accuracy with an additional direct alignment. However, because the sender vehicle is dynamic, the V2V fusion pipeline cannot use a reference point cloud to pre-compute the direct alignment or to reduce network latency. We leave a more thorough examination of this to future work.

Late Fusion with VRF. VRF can enable high accuracy and low latency late fusion with minimal changes to its pipeline. To do this, the RSU will extract and send bounding boxes to the vehicle. At this point, the vehicle would already have aligned with the RSU's reference point cloud and will only need to transform the RSU bounding boxes using that alignment.

Reliance on On-board 3D Map. VRF assumes the on-board presence of a 3D map, the alignment to which reduces the computational complexity of the vehicle-RSU direct alignment. Because 3D maps can be expensive to construct and update, future work can explore using GPS (along with other techniques) for relative localization. Finally, future work can also explore improving alignment accuracy using prior registrations.

7 CONCLUSIONS

Vehicle and road-side point cloud fusion can improve on-board vehicle perception. However, aligning the point clouds can be expensive and inaccurate. Prior work focuses on latency or accuracy, but not both. VRF optimizes for both. VRF's key insight is to align the road-side and vehicle point clouds indirectly *i.e.,* by aligning them to a 3D map. In doing so, it incorporates novel offline registration and alignment accuracy forecasting algorithms. VRF ensures low latency by running operations offline and leveraging previous computations. On real-world testbeds and CarLA traces, VRF fused point clouds with an average accuracy of 5 cm and an end-to-end latency of less than 20 ms. This is an order-of-magnitude improvement over prior work both in terms of latency and accuracy.

Acknowledgements: We are thankful to our shepherd, Yunxin Liu, and the anonymous reviewers for their insightful comments and feedback that helped improve the quality of our paper.

Artifact Appendix: The research artifacts accompanying this paper are available via https://doi.org/10.5281/zenodo.11094357.

REFERENCES

[1] 2020. Introducing 5G Technology and Networks. https://www.thalesgroup.com/en/markets/digital-identity-and-security/mobile/inspired/5G.

[2] 2023. *CloudCompare*. https://www.danielgm.net/cc/

[3] 2023. *Google Maps*. https://maps.google.com/

[4] 2023. *Livoxv Horizon LiDARs*. https://www.livoxtech.com/

[5] 2023. *Ouster-64 beam LiDAR*. https://ouster.com/products/scanning-lidar/os1-sensor/

[6] 2023. *Point Cloud Library*. https://pointclouds.org/

[7] 2023. *ROS - Robot Operating System*. https://www.ros.org/

[8] 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.

[9] Fawad Ahmad, Christina Shin, Weiwu Pang, Jacob Cashman, Branden Leong, Pradipta Ghosh, and Ramesh Govindan. 2024. Cooperative Infrastructure Perception. In *Proceedings of the 9th ACM/IEEE Conference on Internet of Things Design and Implementation (IoTDI '24)*.

[10] Naoki Akai, Luis Yoichi Morales, Eijiro Takeuchi, Yuki Yoshihara, and Yoshiki Ninomiya. 2017. Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1356–1363.

[11] An Open Source Self-Driving Car. 2023. An Open Source Self-Driving Car. www.udacity.com/self-driving-car.

[12] Apollo. 2020. Apollo. https://developer.apollo.auto/developer.html.

[13] Business Wire. 2023. Velodyne LiDAR's Intelligent Infrastructure Solution Deployed in Helsinki Traffic Safety Improvement Project. https://www.businesswire.com/news/home/20220622005178/en/Velodyne-Lidarś-Intelligent-Infrastructure-Solution-Deployed-in-Helsinki-Traffic-Safety-Improvement-Project.

[14] CarLA. 2019. Carla Autonomous Driving Challenge. https://carlachallenge.org/

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

[16] Qi Chen, 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.

[17] Ziming Chen, Yifeng Shi, and Jinrang Jia. 2023. TransIFF: An Instance-Level Feature Fusion Framework for Vehicle-Infrastructure Cooperative 3D Detection with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 18205–18214.

[18] Cruise. 2019. Cruise. https://medium.com/cruise/hd-maps-self-driving-cars-b6444720021c.

[19] Alejandro Diaz-Diaz, Manuel Ocaña, Ángel Llamazares, Carlos Gómez-Huélamo, Pedro Revenga, and Luis M Bergasa. 2022. Hd maps: Exploiting opendrive potential for path planning and map monitoring. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1211–1217.

[20] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. *arXiv preprint arXiv:1711.03938* (2017).

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

[22] 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.

[23] Yuze He, Chen Bian, Jingfei Xia, Shuyao Shi, Zhenyu Yan, Qun Song, and Guoliang Xing. 2023. VI-Map: Infrastructure-Assisted Real-Time HD Mapping for Autonomous Driving. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[24] Yuze He, Li Ma, Jiahe Cui, Zhenyu Yan, Guoliang Xing, Sen Wang, Qintao Hu, and Chen Pan. 2022. AutoMatch: Leveraging Traffic Camera to Improve Perception and Localization of Autonomous Vehicles. (2022).

[25] 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* (New Orleans, Louisiana) *(MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 573–586. https://doi.org/10.1145/3447993.3483276

[26] ITS International. 2022. Gridmatrix to deploy LiDAR in San Mateo to support Vision Zero goals. https://www.itsinternational.com/its2/its4/its5/news/gridmatrix-deploy-lidar-san-mateo-support-vision-zero-goals. Accessed: 2022-01-01.

[27] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach. 2012. Real-time compression of point cloud streams. In *2012 IEEE international conference on robotics and automation*. IEEE, 778–785.

[28] Kelley Blue Book. 2021. *Blind-Spot Monitors: Everything You Need to Know*. https://www.kbb.com/car-advice/blind-spot-monitors/

[29] Kelley Blue Book. 2021. *Lane-Keeping Assist: Everything You Need to Know*. https://www.kbb.com/car-advice/lane-keeping-assist/

[30] Kelley Blue Book. 2021. *What is Adaptive Cruise Control*. https://www.kbb.com/car-advice/adaptive-cruise-control/

[31] Peng-Yong Kong. 2020. Computation and sensor offloading for cloud-based infrastructure-assisted autonomous vehicles. *IEEE Systems Journal* 14, 3 (2020), 3360–3370.

[32] Xiangjie Kong, Haoran Gao, Guojiang Shen, Gaohui Duan, and Sajal K Das. 2021. Fedvcp: A federated-learning-based cooperative positioning scheme for social internet of vehicles. *IEEE Transactions on Computational Social Systems* 9, 1 (2021), 197–206.

[33] 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.

[34] Yiming Li, Shunli Ren, Pengxiang Wu, Siheng Chen, Chen Feng, and Wenjun Zhang. 2021. Learning distilled collaboration graph for multi-agent perception. *Advances in Neural Information Processing Systems* 34 (2021), 29541–29552.

[35] Velodyne LiDAR. 2022. *Intelligent Infrastructure and Vision Zero*. https://velodynelidar.com/blog/intelligent-infrastructure-vision-zero/

[36] Guangyi Liu, Seyedmohammad Salehi, Erdem Bala, Chien-Chung Shen, and Leonard J Cimini. 2022. Communication-Constrained Routing and Traffic Control: A Framework for Infrastructure-Assisted Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 23, 12 (2022), 23844–23857.

[37] Hansi Liu, Pengfei Ren, Shubham Jain, Mohannad Murad, Marco Gruteser, and Fan Bai. 2019. Fusioneye: Perception sharing for connected vehicles and its bandwidth-accuracy trade-offs. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[38] Feng Lyu, Hongzi Zhu, Nan Cheng, Haibo Zhou, Wenchao Xu, Minglu Li, and Xuemin Shen. 2019. Characterizing urban vehicle-to-vehicle communications for reliable safety applications. *IEEE Transactions on Intelligent Transportation Systems* 21, 6 (2019), 2586–2602.

[39] Arvind Merwaday, Satish C Jha, Kathiravetpillai Sivanesan, Ignacio J Alvarez, Leonardo Gomes Baltar, Vesh Raj Sharma Banjade, and Suman A Sehra. 2021. Infrastructure Assisted Efficient Collective Perception Service for Connected Vehicles. In *2021 IEEE Vehicular Networking Conference (VNC)*. IEEE, 119–120.

[40] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. 2019. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 4213–4220.

[41] Keita Miura, Shota Tokunaga, Noriyuki Ota, Yoshiharu Tange, and Takuya Azumi. 2019. Autoware Toolbox: MATLAB/Simulink Benchmark Suite for ROS-Based Self-Driving Software Platform. In *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)* (New York, NY, USA) *(RSP '19)*. Association for Computing Machinery, New York, NY, USA, 8–14. https://doi.org/10.1145/3339985.3358494

[42] Wassim G Najm, Raja Ranganathan, Gowrishankar Srinivasan, John D Smith, Samuel Toma, Elizabeth Swanson, August Burgett, et al. 2013. *Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications*. Technical Report. United States. National Highway Traffic Safety Administration.

[43] NHTSA. 2023. NHTSA | National Highway Traffic and Safety Authority. https://www.nhtsa.gov/.

[44] Jaswanth Nidamanuri, Chinmayi Nibhanupudi, Rolf Assfalg, and Hrishikesh Venkataraman. 2021. A progressive review: Emerging technologies for ADAS driven solutions. *IEEE Transactions on Intelligent Vehicles* 7, 2 (2021), 326–341.

[45] Ouster. 2023. How Chattanooga is Achieving Vision Zero with Ouster Lidar. https://ouster.com/blog/how-chattanooga-is-achieving-vision-zero-with-ouster-lidar/.

[46] Ouster. 2023. Ouster Doubles Smart Infrastructure Programs with Over 210 Awarded Projects in 2022. https://investors.ouster.com/news/news-details/2023/Ouster-Doubles-Smart-Infrastructure-Programs-with-Over-210-Awarded-Projects-in-2022/default.aspx.

[47] 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.

[48] Hang Qiu, Pohan Huang, Namo Asavisanu, Xiaochen Liu, Konstantinos Psounis, and Ramesh Govindan. 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 (MobiSys '22)*.

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

[50] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint*

arXiv:1610.03295 (2016).

[51] 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* (Sydney, NSW, Australia) *(MobiCom '22)*. Association for Computing Machinery, New York, NY, USA, 133–146. https://doi.org/10.1145/3495243.3560539

[52] Karen Vezie. 2016. Mercator's projection: a comparative analysis of rhumb lines and great circles.

[53] Waymo. 2020. Waymo. https://waymo.com/blog/2020/09/the-waymo-driver-handbook-mapping.html.

[54] Waze. 2013. Waze. https://www.waze.com/live-map/.

[55] Business Wire. 2022. U.S. Department of Transportation Secretary Pete Buttigieg Visits Deployment of Velodyne LiDAR's Intelligent Infrastructure Solution at Morgan State. https://www.businesswire.com/news/home/20220503005353/en/U.S.-Department-of-Transportation-Secretary-Pete-Buttigieg-Visits-Deployment-of-Velodyne-Lidarś-Intelligent-Infrastructure-Solution-at-Morgan-State.

[56] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. 2022. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In *European conference on computer vision*. Springer, 107–124.

[57] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. 2022. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics* 38, 4 (2022), 2053–2073.

[58] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. 2015. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE transactions on pattern analysis and machine intelligence* 38, 11 (2015), 2241–2254.

[59] Ryan Yee, Ellick Chan, Bin Cheng, and Gaurav Bansal. 2018. Collaborative perception for automated vehicles leveraging vehicle-to-vehicle communications. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1099–1106.

[60] Zhenxun Yuan, Xiao Song, Lei Bai, Zhe Wang, and Wanli Ouyang. 2021. Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 4 (2021), 2068–2078.

[61] ZDNET. 2020. Verizon, Amazon Demonstrate Connected Vehicles using 5G, Edge Computing with LG, Renovo, Savari. https://www.zdnet.com/article/verizon-amazon-demonstrate-connected-vehicles-using-5g-edge-computing-with-lg-renovo-savari/.

[62] Juyong Zhang, Yuxin Yao, and Bailin Deng. 2021. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2021), 3450–3466.

[63] Qingzhao Zhang, Xumiao Zhang, Ruiyang Zhu, Fan Bai, Mohammad Naserian, and Z Morley Mao. 2023. Robust Real-time Multi-vehicle Collaboration on Asynchronous Sensors. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[64] 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.

[65] Xiangmo Zhao, Kenan Mu, Fei Hui, and Christian Prehofer. 2017. A cooperative vehicle-infrastructure based urban driving environment perception method using a DS theory-based credibility map. *Optik* 138 (2017), 407–415.

[66] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2016. Fast global registration. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 766–782.

[67] Zoox. 2023. Zoox. https://zoox.com/journal/putting-our-robots-on-the-map/.