# VI-Eye: Semantic-based 3D Point Cloud Registration for Infrastructure-assisted Autonomous Driving

Yuze He
The Chinese University of Hong Kong
Hong Kong SAR, China
hy019@ie.cuhk.edu.hk

Li Ma
The Hong Kong University of Science and Technology
Hong Kong SAR, China
lmaag@connect.ust.hk

Zhehao Jiang
The Chinese University of Hong Kong
Hong Kong SAR, China
jz018@ie.cuhk.edu.hk

Yi Tang
The Chinese University of Hong Kong
Hong Kong SAR, China
ytang@ie.cuhk.edu.hk

Guoliang Xing*
The Chinese University of Hong Kong
Hong Kong SAR, China
glxing@cuhk.edu.hk

## ABSTRACT

*Infrastructure-assisted autonomous driving* is an emerging paradigm that aims to make affordable autonomous vehicles a reality. A key technology for realizing this vision is real-time *point cloud registration* which allows a vehicle to fuse the 3D point clouds generated by its own LiDAR and those on roadside infrastructures such as smart lampposts, which can deliver increased sensing range, more robust object detection, and centimeter-level navigation. Unfortunately, the existing methods for point cloud registration assume two clouds to share a similar perspective and large overlap, which result in significant delay and inaccuracy in real-world infrastructure-assisted driving settings. This paper proposes *VI-Eye* - the first system that can align vehicle-infrastructure point clouds at centimeter accuracy in real-time. Our key idea is to exploit traffic domain knowledge by detecting a set of key semantic objects including road, lane lines, curbs, and traffic signs. Based on the inherent regular geometries of such semantic objects, *VI-Eye* extracts a small number of *saliency points* and leverage them to achieve real-time registration of two point clouds. By allowing vehicles and infrastructures to extract the semantic information in parallel, *VI-Eye* leads to a highly scalable architecture for infrastructure-assisted autonomous driving. To evaluate the performance of *VI-Eye*, we collect two new multi-view LiDAR point cloud datasets on an indoor autonomous driving testbed and a campus smart lamppost testbed, respectively. They contain total 915 point cloud pairs and cover three roads of $1.12km$. Experiment results show that *VI-Eye* achieves centimeter-level accuracy within around 0.2s, and delivers a 5X improvement in accuracy and 2X speedup over state-of-the-art baselines.

*Corresponding author


*ACM MobiCom '21, January 31-February 4, 2022, New Orleans, LA, USA*

ACM ISBN 978-1-4503-8342-4/22/01…$15.00
https://doi.org/10.1145/3447993.3483276

## CCS CONCEPTS

• **Computing methodologies** → *Vision for robotics.*

## KEYWORDS

Point cloud registration, Point cloud alignment, Infrastructure-assisted autonomous driving, Vehicle-infrastructure information fusion

**ACM Reference Format:**
Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. 2022. VI-Eye: Semantic-based 3D Point Cloud Registration for Infrastructure-assisted Autonomous Driving. In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '21), January 31-February 4, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3447993.3483276

## 1 INTRODUCTION

Autonomous driving has the potential of revolutionizing transportation mobility and safety. However, the mission-critical nature of autonomous driving has resulted in increasing complex and expensive vehicular systems, which pose significant barriers for adoption in practice. An emerging paradigm that aims to address this challenge is *infrastructure-assisted autonomous driving*, where intelligent roadside infrastructures such as lampposts equipped with computing and sensing units provide autonomous vehicles various real-time services such as object detection and dynamic route planning. A key technology for such a paradigm is real-time data fusion of sensors, e.g., LiDARs, cameras, radars, of vehicles and infrastructures. One of the most widely used sensors adopted by mainstream autonomous driving platforms [1, 2] is LiDAR, due to its precise and long-distance range detection capability. Recent advances in LiDAR technologies have led to a range of affordable off-the-shelf products that can be deployed on vehicles as well as roadside infrastructures [4]. In this work, we study the problem of *aligning/registering* the two 3D point clouds generated by LiDARs on vehicle and infrastructure to form a single integrated point cloud. Such an approach is a key enabling technology for infrastructure-assisted autonomous driving since it allows the vehicle to directly take advantage of the additional LiDAR sensing capability for increased sensing range, more robust object detection/tracking, route planning, localization, and navigation.

Although there exist methods for aligning two point clouds, they are not designed for real-time vehicle-infrastructure point cloud registration and yield unsatisfactory performance in delay, robustness and accuracy. Classic registration methods [6, 10, 12, 38] require similar positions and large overlap between point clouds to achieve good registration performance [11, 15, 27]. Similarly, several latest registration methods [31, 50] assume the two point clouds are scanned from the same vehicle within a short interval of time, and hence have very similar perspective and large overlap. Moreover, most current registration methods [9, 40, 46, 51] focus on aligning small-scale synthetic data (e.g., ModelNet [48]) in indoor scenes. These existing solutions incur prohibitively high computation overhead [19] and poor accuracy when applied to infrastructure-assisted traffic scenarios. First, vehicle- and infrastructure-mounted LiDARs have significantly different field of views, which also change dynamically due to the movement of vehicle. Second, each point cloud scanned in real-world traffic scenes contains tens or even hundreds of thousands of points with significant variations in the point density, noise, and outliers. Yet, in order to support high-precision autonomous driving, two point clouds need to be aligned with the centimeter-level accuracy [29].

This paper proposes *VI-Eye* - the first system that can accurately align vehicle-infrastructure point cloud pairs in real-time for supporting various autonomous applications. Our key idea is to exploit domain knowledge in traffic scenarios to recognize a small number of key semantic objects and utilize them to align vehicle-infrastructure point cloud pairs. Specifically, *VI-Eye* first segments point clouds on both vehicle and infrastructure and detects several key semantic objects, including road, lane lines, curbs, and traffic signs. *VI-Eye* then employs a novel *saliency point extractor* to find key points from these semantic objects by leveraging their inherent regular geometries. Next, motivated by the fact that ground is the largest plane in traffic scenes, *VI-Eye* performs ground registration using semantic objects on the ground to obtain strong prior knowledge for more accurate alignment. Finally, *VI-Eye* aligns vehicle-infrastructure point cloud pairs by searching for corresponding saliency point pairs with two proposed heuristics to achieve real-time registration. *VI-Eye* offers several key advantages. First, it allows vehicles and infrastructures to extract the semantic information from their point clouds independently, which leads to a highly scalable architecture for infrastructure-assisted autonomous driving. Second, by taking advantage of the inherent geometric regularities in semantic objects, *VI-Eye* can achieve precise and real-time vehicle-infrastructure registration.

To evaluate the performance of *VI-Eye* in practice, we collect the first multi-view LiDAR point cloud datasets, which contain both the vehicle and the infrastructure perspectives based on an indoor autonomous driving testbed and a campus smart lamppost testbed, respectively. They contain total 915 point cloud pairs and cover three roads of 1.12km. Experiments show that *VI-Eye* achieves real-time centimeter-level accuracy with a success rate above 94% in the real traffic dataset. Compared with state-of-the-art baselines, *VI-Eye* achieves 5X improvement in accuracy and 2X speedup.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 presents a motivating case study. In Section 4, we introduce the design of *VI-Eye*. We discuss the collection of two datasets and experiment results in Section 5 and 6, respectively. Section 7 concludes the paper and discusses the future work.

## 2 RELATED WORK

**Pairwise Point Cloud Registration.** The ICP algorithm [10] is the most commonly used method for pairwise point cloud registration and has many variants [14, 18, 23, 42, 49]. NDT [12] represents point clouds by a set of Gaussian distributions for quick alignment. These two algorithms have been widely applied in the field of robotics [20, 33, 45]. However, ICP works well only when pair of scans has significant overlapped area (e.g., 90%) and close distance (e.g., 10m) [28]. Similar to ICP, the performance of NDT also highly depends on similar initial positions between two point clouds [19, 35], which are not available in traffic scenes as LiDARs on the vehicle and the infrastructure have significant differences in height and perspective. Feature-based methods, such as SAC-IA [38], ICL [8], and FGR [57], first extract features and then identify correspondences to align point cloud pairs. These methods cannot achieve high precision in traffic scenes, where a large number of repeated structures, such as building facades and roadside trees, make the feature correspondence ambiguous.

In the field of computer vision, learning-based registration methods [9, 40, 46, 51] can achieve good performance on synthetic datasets like ModelNet [48] or small-scale indoor scenes. However, they incur high computation overhead in processing large volumes and irregular point clouds, thus are difficult to be generalized to outdoor scenes. 3DFeatNet [50] and DeepICP [31] are proposed for traffic scenarios, but for aligning two point clouds that are scanned consecutively from the same vehicle perspective. These two methods can not be directly applied to vehicle-infrastructure point cloud registration, where the two point clouds are generated by different sensors with limited overlap.

**Vehicle-infrastructure information fusion.** Most existing vehicle-infrastructure information fusion schemes are designed for fusing high-level application-specific information. The method in [54] integrates the vehicle positioning in a Road Side Unit (RSU) camera with the vehicle GPS to help locate the vehicle. Michael et al. [25] propose to fuse estimations from multiple roadside sensors and vehicle onboard state information to achieve trajectory-level fusion. Siegfried et al. [41] propose to provide the vehicle the object detection results of infrastructure to enhance its perception capabilities. Different from these studies, we focus on fusing raw LiDAR point clouds to enable various autonomous driving applications such as object detection, route planning, localization and navigation.

## 3 A MOTIVATING CASE STUDY

In this section, we use a case study to first show the benefits of vehicle-infrastructure point cloud registration in improving the sensing range and perceptional performance of vehicles. We then discuss the challenges and the shortcomings of existing registration methods. Fig. 1(a) and Fig. 1(b) show a typical scenario of infrastructure-assisted autonomous driving, where two Livox Horizon LiDARs are mounted on a vehicle (1.5m in height) and a roadside lamppost (3.5m in height), respectively. Fig. 1(c) and Fig. 1(d) show the two generated point clouds.

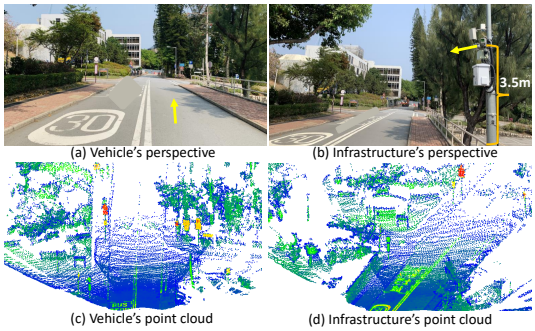



**Figure 1: Illustration of the case study setup. The pointing direction of each LiDAR is labeled.**

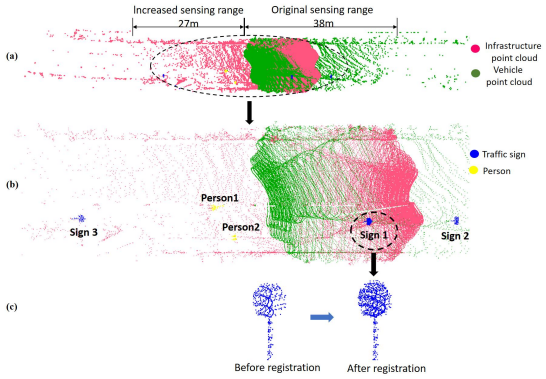



**Figure 2: Illustration of the benefits of vehicle-infrastructure point cloud registration. (a) A bird-eye view of a registration result. (b) A bird-eye view of the detected objects. (c) A closer look at the point cloud of a speed limit sign before and after the registration.**

We present an example of the registration result in Fig. 2 (a), which shows that after aligning/registering the two point clouds, the effective sensing range of the vehicle can be extended from about 38m (green) to 65m (green+pink). Vehicular perception algorithms such as object detection can thus benefit from the extended perception field. In Fig. 2(b), more objects (two people and traffic sign 3) can be detected after registration. Besides, the registration increases the resolution of the original vehicle point cloud, as shown in Fig. 2(c), where more points lie on the traffic sign 1 after the registration, making it easier to be detected. These results show that *VI-Eye* can effectively improve both the sensing range and point cloud density [1].

However, vehicle-infrastructure point cloud registration faces several challenges in practice. First, there often exists significant perspective gap and little overlap between vehicle-infrastructure point cloud pairs. Existing point cloud registration methods can achieve accurate alignment only for point clouds with a small rotation [37] and significant overlap [11]. For example, it is shown [17, 27] that the state-of-the-art point cloud registration methods require the point cloud pair to overlap by at least 30%, which is often not the case in the vehicle-infrastructure point cloud registration due to the significant difference in the field of views of sensors. Fig. 1 shows that there exist huge differences in scanning perspectives (indicated by yellow arrows) and scanning heights of the LiDARs on vehicle and infrastructure. Therefore, it requires significant rotation and translation to align the point cloud pairs. Moreover, overlapping sensing area of the vehicle and infrastructure changes greatly due to the movement of the vehicle, which brings great challenges to vehicle-infrastructure point cloud registration. We also measured the registration errors of four widely used point cloud registration methods on our real traffic dataset (see Section 6.4). It shows that they result in significant errors (up to around 10m) due to the significant perspective gap and little overlap between vehicle-infrastructure point cloud pairs.

Second, to support various real-time autonomous driving applications, vehicle-infrastructure point cloud registration needs to process large amounts of data in a limited time. Consider a real traffic scenario, where a pedestrian obscured by the roadside bushes is about to cross the road. The vehicle speed is 48km/h, the corresponding braking distance is about 14m [47], and the distance from the pedestrian is 20m. We assume the LiDAR point rate is 1333,000 points/s, and the scanning frequency is 10Hz, which are consistent with the settings of mainstream autonomous driving bnechmarks like KITTI Dataset [26]. In this case, to allow the vehicle to take advantage of vehicle-infrastructure point cloud registration and avoid a traffic accident, the registration and the subsequent pedestrian detection need to be completed within $(20-14)/48 = 0.45s$. The run time of the state-of-the-art 3D object detection algorithm is around 0.1s [43]. Therefore, two point clouds containing 266,600 points need to be aligned within 0.35s, which does not account for other delays such as communication between the infrastructure and the vehicle. We list the time complexity of three widely used registration algorithms and their run time on KITTI Dataset using a desktop PC with an Intel i7 2.90GHz CPU and an NVIDIA RTX2060Super GPU in Table 1. It shows that these methods have a complexity of at least $O(nlogn)$ where n is the number of points in a point cloud, and running time of at least 6s, which cannot meet the real-time requirements of autonomous driving.

**Table 1: Time complexity and run time of three widely used point cloud registration algorithms.**

| Algorithms | 4PCS [7] | ICP [10] | SAC-IA [38] |
|---|---|---|---|
| Time Complexity | $O(n^2)$ | $O(nlog(n))$ | $O(n+nlog(n))$ |
| Run time on KITTI dataset (s) | 15.25 | 6.86 | 18.92 |

## 4 DESIGN OF VI-EYE

### 4.1 Approach Overview

Point cloud registration or alignment is the process of finding a transformation between two point clouds so that the overlapping portion matches each other. Given the infrastructure point cloud and the vehicle point cloud, our task is to find a transformation $\mathbf{T} \in SE(3)$ that aligns the two point clouds. $SE(3)$ is the set of all transformation matrices consisting of a rotation matrix $\mathbf{R}$ and a

[1]The results in Fig. 2 are based on two solid-state LiDARs with narrow FoV. Mechanical LiDARs that usually have 360° FoV can also benefit from the registration because of the improved effective sensing range and point cloud resolution.

translation vector **t**. We assume that smart infrastructures have sufficient computing resources to run mainstream deep learning models. Our approach can support two different vehicle-infrastructure data fusion modes. First, once registered, the two point clouds can be easily fused by the vehicle to form a unified point cloud. Second, the transformation between infrastructure and vehicle derived by our approach can also support fusing abstract information, such as objects' bounding boxes. Compared with raw point cloud fusion, abstract information fusion reduces communication overhead between infrastructure and vehicle. However, abstract representations are usually pre-defined and application-specific, which can only assist limited on-vehicle applications, whereas fusing the raw data provides vehicles with a more fine-grained and complete point cloud, which can be used to enhance all kinds of point cloud-based applications. For instance, the complete point cloud are vital to applications like point cloud segmentation for comprehensive scene understanding and path planing that requires precise scene structure. In the rest of this paper, in order to evaluate the performance of our approach, we assume applying registration to raw vehicle-infrastructure point clouds. VI-Eye can also be used to register point clouds of two individual vehicles as long as there exists a certain overlap between two point clouds. By registering point clouds of connected vehicles, a vehicle can perceive the unseen/occluded areas through the point clouds of nearby vehicles. Moreover, through the point cloud registration, a vehicle can also obtain the precise positions of nearby vehicles, which is critical for autonomous driving.

We propose *VI-Eye*, an approach that accurately aligns the vehicle-infrastructure point cloud pairs in real-time. The framework of *VI-Eye* is shown in Fig. 3. The core idea behind *VI-Eye* is to leverage the domain knowledge in autonomous driving applications to shrink the registration domain from the entire traffic scenario to carefully selected semantic objects including road, curbs, lane lines, and traffic signs, and then extract *saliency points*, which refer to the critical points clearly distinguishable from other points. *VI-Eye* has two key advantages. (i) It leverages semantic objects that are static and common objects in traffic scenarios, thus are robust to dynamic scenes and can be easily recognized. Moreover, Ground-related objects provide strong prior knowledge for more accurate alignment. (ii) It allows vehicles and infrastructures to extract the semantic information from their point clouds independently, which leads to a highly scalable architecture for infrastructure-assisted autonomous driving.

To fully utilize semantic information, we propose a new approach to extract saliency points. Different from existing 3D key point detection methods, which detect key points uniformly in the entire point cloud and tend to generate a large number of key points, we take advantage of the inherent geometric characteristics of lane lines and traffic signs to extract only few but effective saliency points at per-instance level. The extracted points, such as vertices and centers of semantic objects are geometrically important and also strongly interpretable. Moreover, the small number of saliency points also significantly reduces the search space of the subsequent registration module, thus greatly speeding up the whole process. Lastly, motivated by the observation that ground is the largest plane in the traffic scenario, we pre-estimate the rotation **R** to further accelerate the registration.

Specifically, *VI-Eye* consists of four components. We first apply semantic segmentation to remove unimportant points and segment remaining points into instances with semantic labels on both vehicle and infrastructure, respectively. We divide these semantic labels into two categories, one with regular shape, including traffic signs and lane lines, the other is ground-related, including road, curbs and lane lines. Instances with regular shapes are then fed into *saliency point extractor* to extract saliency points. In parallel, the *ground registration module* exploits the ground-related semantic objects to find the rotation matrix between the vehicle-infrastructure point cloud pairs, which serves as a registration prior and will be used to speed up the registration module. Lastly, the *Registration module* takes the saliency points in both ends as well as the rotation matrix and estimates the transformation **T** using a RANSAC [22]-based algorithm. We also propose two heuristics that greatly accelerate this process.

## 4.2 Point Cloud Semantic Segmenter

We employ the CNN-based point cloud semantic segmentation algorithm RangeNet++ [34] to segment the point clouds on both infrastructure and vehicle. All the objects in the point cloud are segmented, including both dynamic objects and static objects. The semantic objects that are not helpful for registration will be automatically excluded, such as dynamic objects like cars and pedestrians. Only four categories of static semantic objects, road, curbs, lane lines and traffic signs are left for further processing, which ensures a robust registration performance despite the traffic dynamics. RangeNet++ leverages the working principle of LiDARs to project the 3D point cloud $\mathbb{P} = \{\mathbf{p}_0, \mathbf{p}_1, ...\}$ to a cylinder plane and obtain a 2D range image and use CNN to segment the image. Semantic labels are then projected back and assigned to each 3D point. RangeNet++ is designed for Velodyne HDL-64E LiDAR and assumes a 360° FoV. To extend it to LiDARs with different FoVs, we redefine the mapping $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ that converts each 3D point $\mathbf{p}_i = (x, y, z)$ to a pixel coordinate $(u, v)$ in the range image as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x) f_h\right] w \\ \left[1 - \left(\arcsin\left(zr^{-1}\right) + f_{up}\right) f^{-1}\right] h \end{pmatrix}, \tag{1}$$

where $(h, w)$ is the constant size of the range image. $f = f_{up} + f_{down}$ is the vertical FoV of the LiDAR, including both the vertical FoV upward and downward. $f_h$ is the horizontal FoV, and $r = \|\mathbf{p}_i\|_2$ is the range of point $\mathbf{p}_i$. For the Livox Horizon LiDAR used in this paper, $f = 25.1°$, $f_{up} = f_{down} = 12.55°$, $f_h = 81.7°$, and we set $h = 128$ pixels, $w = 512$ pixels.

After segmentation, we apply Euclidean point cloud clustering to separate different instances in the same semantic category. Each instance will be processed independently in the subsequent steps.

## 4.3 Saliency Point Extractor

The saliency point extractor is designed to extract semantically and geometrically meaningful points from two categories of semantic objects: lane lines and traffic signs. The novelty of this module lies in three aspects: (i) Classic key point detection methods [24, 44, 55] use features such as normal and curvature to extract corner points, which may result in a large number of key points that are not clearly distinguishable from each other. In contrast, *VI-Eye* uses

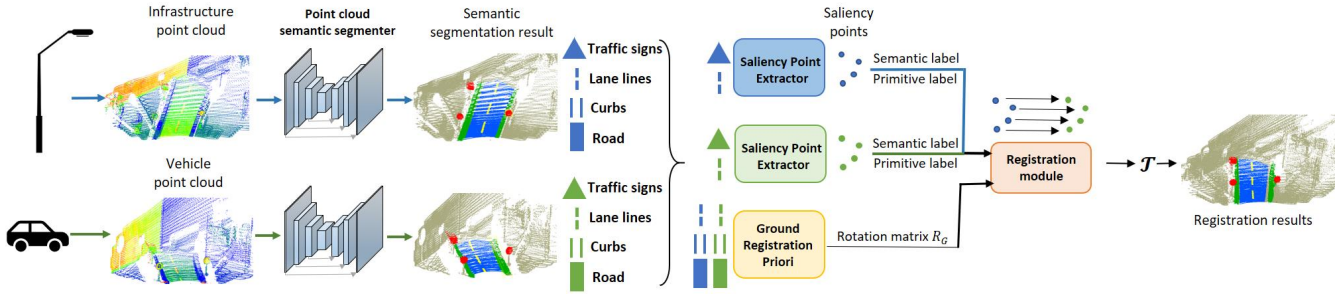



**Figure 3: Framework of our vehicle-infrastructure point cloud registration approach**

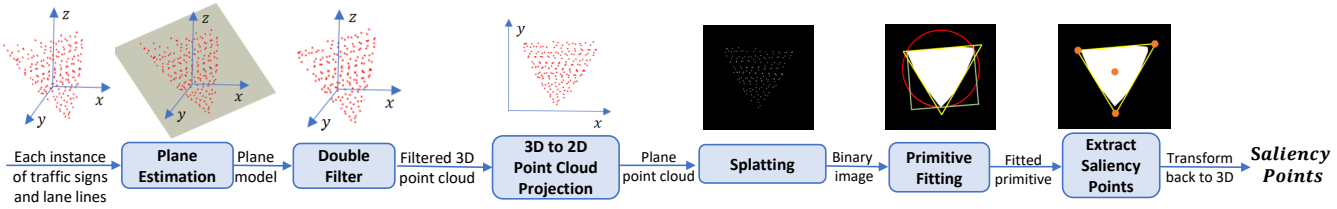



**Figure 4: Design of the Saliency Point Extractor**

semantic and primitive features to extract saliency points, which are strongly distinguishable and descriptive. (ii) Learning-based 3D feature extraction methods [52, 53] usually first convert point clouds into regular 3D voxels and then use the 3D CNN to calculate the features. However, voxelization leads to information loss and its accuracy is affected by the resolution of the 3D grid. *VI-Eye* only processes semantically important objects' point clouds by projecting them to different 2D planes, which is more efficient. (iii) Similar to *VI-Eye*, several methods [36, 56] also represent the point cloud as images rendered from one or multiple views before applying image CNNs to calculate features. However, their performance is greatly affected by the projection direction, point density, and projected image resolution. By exploiting traffic domain knowledge, *VI-Eye* represents point clouds as multiple 2D binary images and derives the projection direction for each semantic object, which enables highly robust registration despite significant variations in point density and viewpoints of point clouds.

*4.3.1 Preprocessing.* The movement of the vehicle may cause "tails" on the edge of the instance point cloud. We define "tails" as points that belong to an object but are outside the contour of the object due to the movement of the vehicle and the scanning delay of the LiDAR. Moreover, the segmentation may introduce points that do not belong to the semantic instance. Therefore, preprocessing is vital towards the robustness of saliency point extraction.

Fig. 5 shows an example before and after applying the filtering. Specifically, in the first round we fit a plane model $\mathcal{P}(\mathbb{S})$ using least square for each semantic instance $\mathbb{S}$, and filter out points with distance larger than 5cm from the plane. In the second round we use *StatisticalOutlierRemoval* [39] to further remove sparse outliers and obtain filtered point cloud $\mathbb{S}_f$.

*4.3.2 Converting to binary images.* After filtering the point cloud, we first perform 3D to 2D point cloud projection, which is achieved by orthogonally projecting point cloud $\mathbb{S}_f$ onto the plane $\mathcal{P}(\mathbb{S})$

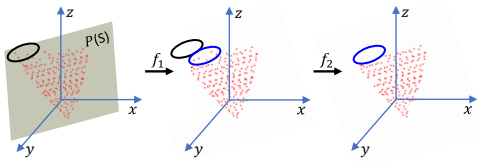



**Figure 5: Illustration of preprocessing: first filter the points outside the plane model $\mathcal{P}(\mathbb{S})$ (black circles), and then filter the outliers that do not belong to the semantic instance (blue circles).**

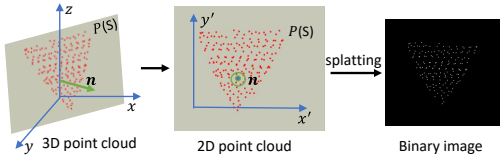



**Figure 6: Converting 3D point cloud to binary image: first project point cloud of each instance $\mathbb{S}_f$ to its plane $\mathcal{P}(\mathbb{S})$, then splat projected point cloud $\mathbb{S}_p$ to image I.**

estimated in the last step, and then remove the $z$ coordinate to obtain 2D point cloud $\mathbb{S}_p$. An illustration of this step is shown in Fig. 6.

After we obtain the set of 2D points continuously distributed on a bounded 2D space, we convert them to an image $\mathbf{I}$ by splatting points into a regular rectangular grid (shown in Fig. 6). Each point $(x, y)$ is splatted into the nearest grid with coordinates $(u, v)$ by a mapping $\boldsymbol{Q}$:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} [(x - x_{\min})/\alpha + \beta] \\ [(y - y_{\min})/\alpha + \beta] \end{pmatrix} \tag{2}$$

where $[\cdot]$ is the rounding operation, $\alpha$ and $\beta$ are used for correcting the scale and the translation, respectively. $\alpha$ indicates the resolution

of image **I**, e.g., when $\alpha = 1000$, the resolution of image **I** is 1mm. Since most LiDARs have millimeter-level detection accuracy, $\alpha$ can be set to 1,000 in most cases. $\beta$ indicates the margin of image **I**, which is only used for visualization. In principle, $\beta$ can be any value, which won't affect the results of *VI-Eye*. We empirically set $\beta = 100$, leaving a margin of 100 pixels around $\mathbb{S}_p$. For each mapped coordinates we set the correspond pixels to 1, which results in a binary image **I**.

*4.3.3 Primitive fitting.* The semantic instances in our problem usually have regular shapes. We leverage this strong prior by fitting the binary image **I** into a regular primitive image **M** through connected component analysis. The primitives in this paper refer to a set of geometries:$\mathbb{G} = \{circle, triangle, rectangle\}$, which covers the most shapes of objects we concerned. We first decide the best primitive label by an *extent* score:

$$extent\ (\mathbf{I})_i = \frac{\mathcal{A}(\mathcal{H}(\mathbf{I}))}{\mathcal{A}\left(\mathcal{P}_i(\mathcal{H}(I))\right)}, i \in \mathbb{G} \tag{3}$$

where $\mathcal{H}(\mathbf{I})$ is the convex hull of image **I**, $\mathcal{P}_i(\cdot)$ calculates the minimum enclosing primitive with label $i$ of the convex hull. $\mathcal{A}(\cdot)$ is the area. *Extent* measures the similarity between the primitive $\mathcal{P}_i(\mathcal{H}(\mathbf{I}))$ and the semantic instance's geometry. We choose the label $i$ that maximizes this score as the primitive, and extract the vertices and center of $\mathcal{P}_i(\mathcal{H}(\mathbf{I}))$ as saliency points. The detailed primitive fitting process is shown in Fig. 7.

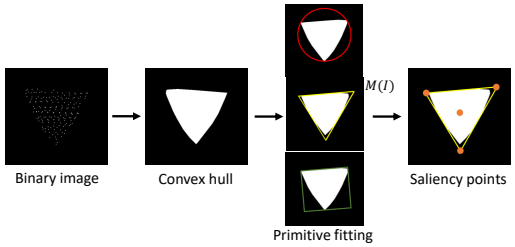



**Figure 7: The primitive fitting process that finds the best primitive for a binary image.**

*4.3.4 Labeling of the saliency point.* Primitive label is an important categoric feature that can be used for better matching. However, we observed that there are cases (although rare) where the primitives fitted on vehicle and infrastructure are different for the same traffic sign. For example, a speed limit sign may be fitted to a circle on vehicle but a rectangle on infrastructure, which makes a pair of salient points fail to match. Another observation is that the center, different from vertices, yields strong robustness against wrong primitive fitting. Thus we label each point with two labels $(s, p)$: $s$ is the semantic label, and $p \in \mathbb{G} \bigcup \{center\}$, where *center* indicates that the saliency point is a center point, regardless of which kind of primitive it belongs to. Finally, we back project the saliency points to 3D and get set of saliency point $\mathbb{S}$, with each point attached with two labels.

## 4.4 Ground Registration

Ground is a unique and strong prior in traffic scenes. By exploiting the ground, we can robustly estimate a rotation matrix that accelerates the registration process. Specifically, we focus on the point clouds of semantic objects on the ground, e.g. road, lane lines and curbs. On both the infrastructure and the vehicle, we first use the points on the road $\mathbb{R}$ to estimate a plane model $\mathcal{P}(\mathbb{R})$. We then exploit lane lines or curbs to estimate the direction of the road **d**. By aligning the normal vectors $\mathbf{n_I}$ and $\mathbf{n_V}$ of the ground plane and the direction vectors $\mathbf{d_I}$ and $\mathbf{d_V}$ of the road, which are estimated on infrastructure and vehicle respectively, we can obtain a rotation matrix $\mathbf{R}_G$. $\mathbf{R}_G$ is an estimation of the rotation transformation that rotates the infrastructure point cloud to align with the vehicle point cloud. Now we have eliminated three DoF and only translation remains ambiguous. Next, we need to match saliency point pairs to obtain the complete transformation between vehicle-infrastructure point clouds, i.e., a rotation matrix and a translation vector. The ground registration process is shown in Fig. 8.

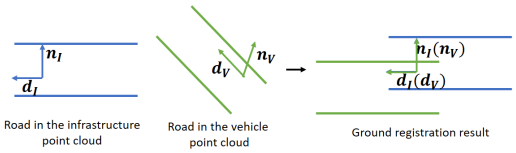



**Figure 8: Ground registration.**

## 4.5 Registration

We now have the rotation matrix $\mathbf{R}_G$ from the ground registration and two saliency point sets $\mathbb{S}_V$ and $\mathbb{S}_I$ extracted from the vehicle and the infrastructure respectively. We adopt the idea from RANSAC [22] to quickly find the correspondence in $\mathbb{S}_V$ and $\mathbb{S}_I$ and estimate $\mathbf{T}_{I2V}$ that aligns the infrastructure point cloud to the vehicle point cloud. The details are shown in Algorithm 1.

Note that most time consumed in the naive implementation is counting the number of inliers, which is an O(n) operation. Therefore, we propose two key extensions to the original RANSAC algorithm to achieve real-time performance. The first extension, namely *two-tier label verification*, narrows the search space when guessing the correspondence. Specifically, for each saliency point $\mathbf{p}_I^i$ in $\mathbb{S}_I$, we only consider the saliency points in $\mathbb{S}_V$ that have the same two labels as $\mathbf{p}_I^i$, which greatly increases the probability of a correct guess. Another interesting property of this approach is that, unlike RANSAC that simply drops the wrong guess, we can benefit from the failures. When there do not exist any saliency points in $\mathcal{S}_\mathcal{V}$ that have the same label, e.g. the same object is fitted into different primitives on vehicle and infrastructure, or the object exists only in one view, it is guaranteed that no correspondence can be found for $\mathbf{p}_I^i$. So we can safely delete the saliency points that have the same label as $\mathbf{p}_I^i$ to further reduce the search space in the following iterations.

The other extension, referred to as *early check*, aims at terminating the iteration early to reduce unnecessary computation when there exists false correspondence. After we randomly guess three pairs, we quickly check whether the assumption is a valid one by comparing the similarity of **R** and $\mathbf{R}_G$, which is an O(1) operation:

$$Similarity(\mathbf{R},\ \mathbf{R}_G) = \frac{\mathbf{r} \cdot \mathbf{r}_G}{|\mathbf{r}| \cdot |\mathbf{r}_G|} \tag{4}$$

where **r** and $\mathbf{r}_G$ are vector representations of the rotation matrix **R** and $\mathbf{R}_G$. Note that the road direction vectors $\mathbf{d_I}$ and $\mathbf{d_V}$ estimated

**Algorithm 1** Extended RANSAC Registration Algorithm

**Input:** Ground registration: $\mathbf{R_G}$, Saliency points in vehicle end: $\mathbb{S}_\mathbb{V}$, Saliency points in infrastructure end: $\mathbb{S}_\mathbb{I}$.
**Output:** Transformation between the vehicle-infrastructure point cloud pair: $\mathbf{T_{I2V}}$.

```
for n = 1 to MaxIterationNum do
    S ← Randomly select 3 points from S_I;
    for each p_I^i ∈ S do                          ▷ Two-tier label verification
        S' ← Points in S_V with the same labels as p_I^i;
        if |S'| = 0 then
            Remove points in S_I with same labels as p_I^i;
            End this iteration and start a new iteration;
        else
            p_V^i ← Randomly select 1 points from S';
            Add (p_I^i, p_V^i) to base B;
        end if
    end for
    T_I2V ≡ [R t] ← CalculateTransforamtion(B);
    if Similarity(R, R_G) > ε or Similarity(R, R_G') > ε then
                                                   ▷ Early check
        Apply transform T_I2V to S_I, update base B;
        Re-estimate T_I2V using the new base B;
        Count the number of inlier saliency points and
        determine whether to end the registration;
    else
        End this iteration and start a new iteration;
    end if
end for
```

by the infrastructure and the vehicle can either be the same or opposite in the world coordinate. Therefore, we need to compare $\mathbf{R}$ with both $\mathbf{R}_G$ and $\mathbf{R}'_G$, and calculate two similarities. $\mathbf{R}'_G$ is obtained by replacing $\mathbf{d_I}$ with $-\mathbf{d_I}$ in Ground Registration. We deem the correspondence as being correct when either $Similarity(\mathbf{R}, \mathbf{R}_G)$ or $Similarity(\mathbf{R}, \mathbf{R}'_G)$ is bigger than a certain threshold $\varepsilon$. If so, the original RANSAC algorithm is used to count inliers. Otherwise, the current iteration is ended early. $\varepsilon$ is an error tolerance set by users. When $\varepsilon$ is small, the run time of *VI-Eye* can be short but may lose some registration accuracy. When $\varepsilon$ is large, the registration is more precise at the price of longer searching time. So $\varepsilon$ can be set to achieve a desirable tradeoff between accuracy and speed.

# 5 MULTI-VIEW LIDAR POINT CLOUD DATASETS

We collect two new point cloud datasets with multi-view LiDARs. The first dataset is collected based on an indoor autonomous driving testbed, and the second dataset is collected on a university campus. Unlike existing autonomous driving public datasets, such as KITTI Dataset [26], Oxford RobotCar Dataset [32], etc., both of our two datasets contain the point clouds from two different views: the vehicle's view and the fixed roadside infrastructure's view. We summarize our two datasets in Table 2. For the indoor dataset, we set up 89 simulated traffic scenarios and collected 445 vehicle-infrastructure point cloud pairs. For the campus dataset, we collected 470 point cloud pairs, covering three roads with a total length of 1.12km. In addition, we tagged each point cloud with semantic labels and provide the ground-truth transformation of each point cloud pair. The examples of collected point clouds are shown in Section 6. Below we describe the data acquisition process of each dataset in detail.

**Table 2: Summary of two datasets.**

| Datasets | #scene settings | #point cloud pairs | length | semantic labels | registration ground truth |
|---|---|---|---|---|---|
| Indoor dataset | 89 | 445 | 8m | ✓ | ✓ |
| Campus dataset | - | 470 | 1.12km | ✓ | ✓ |

## 5.1 Campus Traffic Dataset

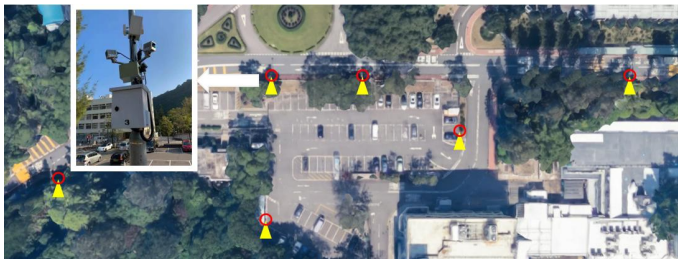


**Figure 9: Smart lamppost testbed.**

This dataset is collected based on the smart lamppost testbed deployed on a university campus, covering complex traffic scenes, such as steep slopes, intersections, and road constructions. Currently, the testbed contains six smart lampposts, whose locations are shown in Fig. 9. Different sensors are installed on each lamppost, including one thermal camera, one Radar, and two LiDARs. The two LiDARs on each lamppost are installed at the height of about 3.5m and facing opposite directions of the road (Fig. 9). Moreover, we use a trolley to simulate a vehicle on the road and collect the vehicle point clouds by the LiDAR mounted on it, which is about 1.5m above the ground. The LiDARs installed on the smart lampposts and the trolley are all Livox Horizon LiDARs. To cover more diverse traffic scenarios, we also collected point cloud data on two other roads of the campus using a moving pole and keep the height of the LiDAR on the pole consistent with the smart lampposts.

## 5.2 Indoor Simulated Traffic Dataset

Fig. 10 shows the hardware and set up for the point cloud data collection of the indoor traffic dataset. They include a pole used as a simulated roadside lamppost infrastructure and an F1/10 Autonomous Vehicle [5] equipped with one LiDAR and one heterogeneous embedded platform (NVIDIA TX2 with Orbitty Carrier board). Both of them are equipped with Livox Horizon LiDAR. Fig. 10(c) shows the scenes, and Fig. 10(d) shows the layouts of the testbed.

In this dataset, we use fences to simulate the road's curbs, and place different traffic signs and a pole beside the road. We set the width of the road be about a quarter of the real-world highway lane width. Other important parameters, such as the length of the lane lines, the size of the traffic signs, and the height of the LiDAR on

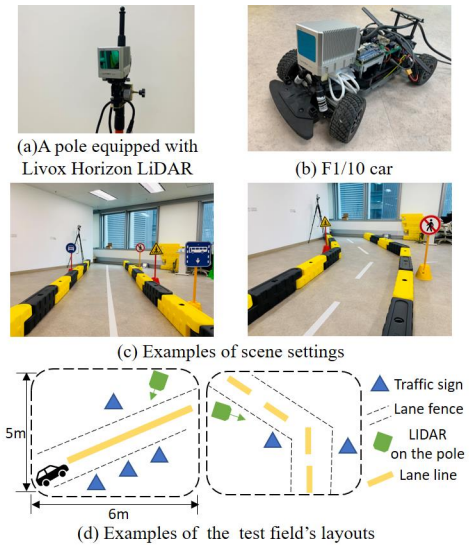



**Figure 10: The indoor dataset collection.**

the pole, are set to be about half of the sizes of real-world roads [3]. We simulate a variety of different traffic scenarios on this testbed, by changing the types of lanes, numbers and locations of traffic signs, pitch angles of the LiDAR on the pole, point cloud overlap ratios, and degrees of occlusion at the vehicle end, etc.

# 6 EVALUATION

In this section, we first describe the experimental setup and define evaluation metrics in Section 6.1 and Section 6.2. Second, we present application-level results in Section 6.3, which shows that *VI-Eye* significantly extends the vehicle's sensing distance and greatly improves the resolution of vehicle point clouds. Third, we validate *VI-Eye* on real traffic scenes and the indoor simulated traffic testbed in Section 6.4 and Section 6.5, respectively. In addition, we evaluate the performance of two key modules of *VI-Eye*, the saliency point extractor and the registration in Section 6.6 and Section 6.7.

## 6.1 Experimental Settings

We divide each of the two self-collected datasets into two groups: an easy group and a hard group. The hard group contains point cloud pairs with a large rotation ($> 100°$) or a small overlap ratio ( $< 50\%$). The rest of point cloud pairs constitute the easy group. The details of the two datasets are shown in Table 3. We use the open-source tool, CloudCompare to calculate the ground-truth transformation of each vehicle-infrastructure point cloud pair. Specifically, for each cloud pair, We first manually select several corresponding point pairs ($> 3$ pairs) and then use CloudCompare to calculate the transformation, which is regarded as the ground-truth.

We use Jetson TX2 as the computing platform on infrastructure, and a desktop PC with an Intel i7 2.90GHz CPU and an NVIDIA RTX2060Super GPU as the computing platform at the vehicle end. We finetune the public model of RangeNet++ using our two datasets. The semantic labels of our datasets are manually labeled using the CloudCompare software. Other modules are implemented using Point Cloud Library [39] and OpenCV library [13].

**Table 3: Grouping of two datasets used in evaluation.**

| Dataset | type | #scene settings | #point cloud pairs |
|---|---|---|---|
| Indoor | easy group | 43 | 215 |
| | hard group | 46 | 230 |
| Outdoor | easy group | - | 270 |
| | hard group | - | 200 |

## 6.2 Evaluation Metrics

*6.2.1 Effective sensing distance and point cloud density.* In order to measure how much autonomous vehicles can benefit from *VI-Eye* in practice, we define two application-level metrics, the *effective sensing distance* and the *point cloud density*. These two metrics respectively measure the range and quality of vehicle perception. The definition of effective sensing distance is as follows:

$$ESD = max\|p_g^i\|_2 + max\|p_g^j\|_2 \text{ s.t. } \left|N\left(p_g, d\right)\right| \geq threshold \quad (5)$$

where $p_g$ indicates the points on the ground, $i, j$ satisfy $p_g^i[x] < 0$ and $p_g^j[x] > 0$. $N(p, d)$ is the set of adjacent points of a point $p$ within a length of $d$ radius. Only points with the number of neighbour points more than *threshold* within a $d$ radius are considered effective sensing points. $[x]$ indicates the $x$ coordinate of point $p_g$, and x-axis is the direction where the vehicle is heading. Effective sensing distance indicates the distance within which the performance of object detection is above a predefined threshold. It reflects "how far the vehicle can see".

The point cloud density is defined within the effective sensing distance, which is calculated as: $PCD = \frac{|V(S_g)|}{S_g}$ where $S_g$ is the effective sensing area of the road, i.e., the effective sensing distance × the road width. $V(S_g)$ is a set of points that includes the points on $S_g$ and all points in the vertical space above $S_g$. This is equivalent to projecting the non-ground points orthographically onto the ground and then calculating the plane point density. This metric thus directly reflects the resolution of a point cloud. It reflects "how clear the vehicle can see".

*6.2.2 RRE, RTE and binary success rate.* We adopt the metrics defined in [26] and used in [15, 16, 21] to compare the performance with other registration algorithms. They include the relative rotation error (RRE) and the relative translational error (RTE), which measure the point cloud registration errors between the estimated transformation $T_E$ and the ground-truth transformation $T_T$. RRE is defined as:

$$\begin{aligned} E_R &= \textstyle\sum_{i=1}^{3} \mid \text{angle } (i) \mid \\ \text{angle } &= F\left(R_T^{-1} R_E\right) \end{aligned} \quad (6)$$

where $R_T$ and $R_E$ are the rotation matrices of the ground-truth transformation and the estimated transformation, respectively. $F(\cdot)$ transforms a rotation matrix to three Euler angles. RRE is the sum of the absolute differences in three Euler angles. RTE is defined as: $E_T = \|t_T - t_E\|_2$, where $t_T$ and $t_E$ are the translation vectors of the ground-truth transformation and the estimated transformation, respectively. Besides, we define the registration result as a 'success' when the RTE is below a predefined threshold as in [21].

*6.2.3 Effective key point ratio.* The effective key point ratio is defined to measure the effectiveness of our saliency point extractor

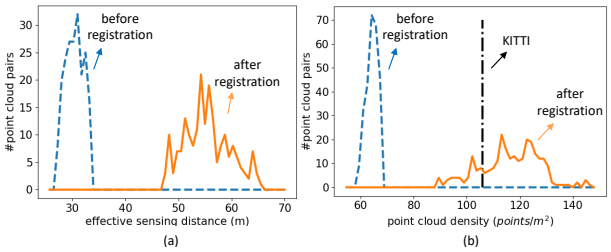



**Figure 11: Distributions of effective sensing distance and point cloud density before and after registration.**

and other key point extraction methods[2]. The criteria of whether a key point $p_I^i$ is effective is as follows:

$$\min \left\| T_T\left(p_I^i\right) - p_V^j \right\|_2 < threshold, p_V^j \in \mathcal{S}_\mathcal{V} \tag{7}$$

where $p_I^i$ and $p_V^j$ are the key points in the infrastructure point cloud and the vehicle point cloud, respectively. and $\mathcal{S}_\mathcal{V}$ is the set of key points in the vehicle point cloud. $T_T(\cdot)$ transforms a point from infrastructure to vehicle by the ground-truth transformation $T_T$. Key point $p_I^i$ in the infrastructure point cloud is defined as "effective" when there is a key point in the vehicle point cloud that is close enough (within some threshold distance) to it after the ground-truth transformation aligns the vehicle-infrastructure point cloud pair.

The effective key point ratio reflects the quality of the extracted key points and how much the key point extraction method contributes to a good registration result.

## 6.3 Application-level Results

We evaluate how much application-level perception improvement *VI-Eye* can bring to autonomous vehicles using the real traffic dataset. This evaluation supports our claims that: (i) *VI-Eye* extends the vehicle's effective sensing distance to areas that it could not perceive before the vehicle-infrastructure point cloud registration; (ii) For those areas that the vehicle can perceive on its own before registration, *VI-Eye* greatly increases the point cloud density even to the level surpassing the top-end LiDARs.

*6.3.1 Increase of the effective sensing distance.* For each vehicle-infrastructure point cloud pair, we calculate the average effective sensing distances of the vehicle before and after registration using Equation (5), where $d$ and $threshold$ are set to 1m and 30 points. As the length of the extended effective sensing distance varies with the relative position between the vehicle and the infrastructure, we measure the average improvement for easy group and plot the distributions in hard group before and after registration in Fig. 11 (a). In the easy group, *VI-Eye* extends the vehicle's effective sensing distance from an average of about 30m to around 52m, i.e., an 73% improvement. Fig. 11 (a) shows that *VI-Eye* significantly improves vehicle's effective sensing distance from $27-33$m to $47-65$m. In the best case, the effective sensing distance of the vehicle can be more than doubled.

[2]We use key points and saliency points interchangeably in the rest of this section.

*6.3.2 Increase of the point cloud density.* According to the definition in Section 6.2.1, we calculate the point cloud density of the original effective sensing area before and after the registration. We plot the distributions of point cloud density in easy group before and after registration in Fig. 11 (b), which shows that *VI-Eye* significantly improves the point cloud density of the original effective sensing area from $59-67$ $points/m^2$ to $88-147$ $points/m^2$, i.e., an average 97% improvement. For the hard group, *VI-Eye* nearly doubles the point cloud density from an average of around 65 $points/m^2$ to an average of around 124 $points/m^2$.

To have a more intuitive understanding on the improvement of point cloud density after registration, we also calculate the average point cloud density of the sequence "00" in the KITTI dataset, where the point clouds are collected by Velodyne HDL-64E LiDAR, one of the most expensive top-end LiDARs on the market. It can be seen from Fig. 11 (b) that *VI-Eye* boosted the point cloud density of the original effective sensing range even higher than that of Velodyne HDL-64E LiDAR [3].

In summary, from the application-level perspective, *VI-Eye* can enable significantly longer sensing range and more refined perception for autonomous vehicles.

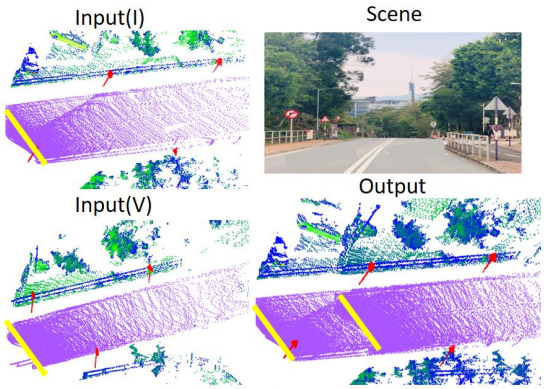
## 6.4 Results of Campus Traffic Dataset

We present extensive performance evaluation by comparing with four point cloud registration algorithms. (i) ICP [10], the most widely used point cloud registration algorithm; (ii) NDT [12], an algorithm based on probability density; (iii) The feature based algorithms, SAC-IA [38] and FGR [57]. The implementations of ICP, NDT, and SAC-IA are based on the Point Cloud Library (PCL) [39], and the implementation of FGR is from Open3D [58]. we adjusted the parameters of the baselines to optimize the best performance in our datasets.

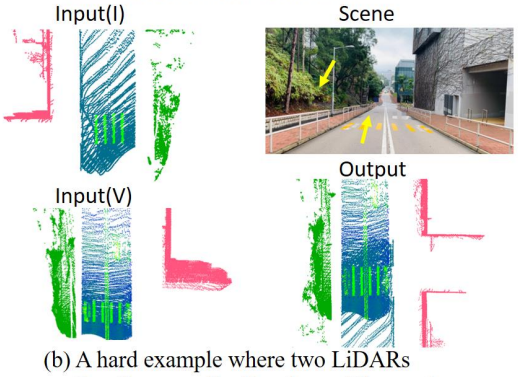
As introduced in Section 6.1, we divide the real traffic dataset into an easy group and a hard group. We first show two typical examples where *VI-Eye* successfully registered cloud pairs with centimeter-level accuracy while all baselines performed poorly. Fig. 12 (a) shows an easy example, where LiDARs on vehicle and infrastructure face the same direction of the road but are separated by a certain distance. In this case, all the baselines tend to align the two starts (yellow lines) of two point clouds, which leads to registration errors of several meters. Fig. 12 (b) shows a hard example, where LiDARs on vehicle and infrastructure face the opposite directions of the road (indicated by yellow arrows). In this case, the infrastructure point cloud needs a large rotation (more than 100°) to be aligned with the vehicle point cloud. However, all the baselines fail on this large rotation and result in registration errors of more than 20m.

Table 4 shows the experiment results on this real traffic dataset, which shows the success rate and the average RRE, RTE scores and the run time of the successful registrations. Results with RTE lower than 2m are viewed as 'success', which is consistent with the requirements of autonomous driving navigation [30]. For RRE, *VI-Eye* is at most one-fifth of other baselines. For RTE, *VI-Eye* is 19.34% to 0.61% of other baselines. For run time, *VI-Eye* is $2-165$ times faster than other baselines. Moreover, *VI-Eye* improves the

[3]The market price of LiDAR used in our experiments is about $800, 1% of that of Velodyne HDL-64E LiDAR.




(a) An easy example where two LiDARs face the same directions of the road.




(b) A hard example where two LiDARs face the opposite directions of the road.

**Figure 12: Two typical registration results of *VI-Eye*. We crop unimportant points for better visualization.**

**Table 4: Results of different registration algorithms on the real traffic dataset.**

| | methods | RRE(°) | RTE(cm) | time(s) | success rate |
|---|---|---|---|---|---|
| easy group | ICP | 7.13 | 75.74 | 3.98 | 11.48% |
| | NDT | 12.62 | 122.17 | 0.48 | 9.26% |
| | SAC-IA | 15.89 | 81.83 | 25.73 | 11.11% |
| | FGR | 22.80 | 150.21 | 1.18 | 21.85% |
| | ***VI-Eye*** | **2.01** | **14.65** | **0.22** | **95.93%** |
| hard group | ICP | 181.34 | 2700.25 | 4.09 | 0% |
| | NDT | 187.15 | 2740.52 | 0.94 | 0% |
| | SAC-IA | 142.57 | 2650.46 | 31.4 | 0% |
| | FGR | 225.88 | 2720.44 | 1.36 | 0% |
| | ***VI-Eye*** | **2.08** | **16.69** | **0.19** | **94.00%** |

success rate by at least 4.5 times. In the hard group, the RTE of these four baselines is as high as 27m. We can also see that, *VI-Eye*'s run time on the hard group is slightly shorter than that on the easy group. This is reasonable because in the easy group, overlap ratio is larger, which results in multiple matching saliency point pairs between the infrastructure and vehicle. This further leads to large search space for the registration module. On the other hand, the overlap ratio is small for the hard group, results in less consistent saliency points between the vehicle-infrastructure point cloud pairs. Thanks to our extension of the *two-tier label verification*, most of those inconsistent saliency points can be easily removed, thereby greatly reducing the search space of the registration. We show some example registration results in Fig. 13, which shows that *VI-Eye* can successfully align the vehicle-infrastructure point cloud pair where all baselines fail. Another example of registration results in Fig. 15 shows that *VI-Eye* achieves more refined registration compared to the best performing baseline ICP.

## 6.5 Results of Indoor Simulated Traffic Dataset

We also compare *VI-Eye* with the four baselines on the indoor simulated traffic dataset. The threshold of the registration success is set to 1m (same as [21]). Fig. 14 shows the qualitative registration results on this dataset, which shows that *VI-Eye* achieves a more accurate registration result (see the anastomosis of walls). The performance of *VI-Eye* and other baselines are shown in Table 5.

**Table 5: Results of different registration algorithms on the indoor simulated traffic dataset**

| | methods | RRE(°) | RTE(cm) | time(s) | success rate |
|---|---|---|---|---|---|
| easy group | ICP | 18.48 | 75.72 | 2.12 | 33.02% |
| | NDT | 21.07 | 93.24 | 1.37 | 26.05% |
| | SAC-IA | 32.11 | 19.22 | 47.41 | 40.00% |
| | FGR | 36.12 | 90.51 | 0.85 | 13.49% |
| | ***VI-Eye*** | **2.55** | **8.62** | **0.23** | **94.88%** |
| hard group | ICP | 206.86 | 755.66 | 1.92 | 0% |
| | NDT | 194.03 | 739.11 | 1.19 | 0% |
| | SAC-IA | 239.40 | 686.44 | 49.31 | 0% |
| | FGR | 217.70 | 744.36 | 0.69 | 0% |
| | ***VI-Eye*** | **3.26** | **9.02** | **0.20** | **89.13%** |

Although the point cloud pairs in the easy group have a large overlap ratio, the four baselines do not yield good performance. For RRE, other methods results in at least 18.48°. In contrast, the error of *VI-Eye* is at least one order of magnitude smaller. For RTE, *VI-Eye* reduces the error by more than half compared to the best performing baseline. For run time, *VI-Eye* is about a quarter of the fastest baseline. The run time of NDT and FGR is significantly shorter than the other two baselines. That's because they do not require computing point to point correspondence (like ICP) or iterative sampling (like SAC-IA). For success rate, all the four baselines are less than half. The reason is that there are inherently large differences in scanning heights and scanning perspectives between the vehicle-infrastructure point cloud pairs. In the hard group, the overlap ratio is small or the rotation is large. We can see that all four baselines fail to align any vehicle-infrastructure point cloud pairs. *VI-Eye* shows a high success rate in both two groups. It can achieve a centimeter-level registration accuracy within around 0.2s.

Another observation is that the performances of the four baseline methods in both groups are better on the indoor simulated dataset, especially for RTE and success rate. This is expected because the campus dataset contains longer roads and more complex and diverse features than the indoor dataset, which greatly reduced the accuracy of existing registration methods. On the contrary, our method achieves better success rate on the campus dataset than

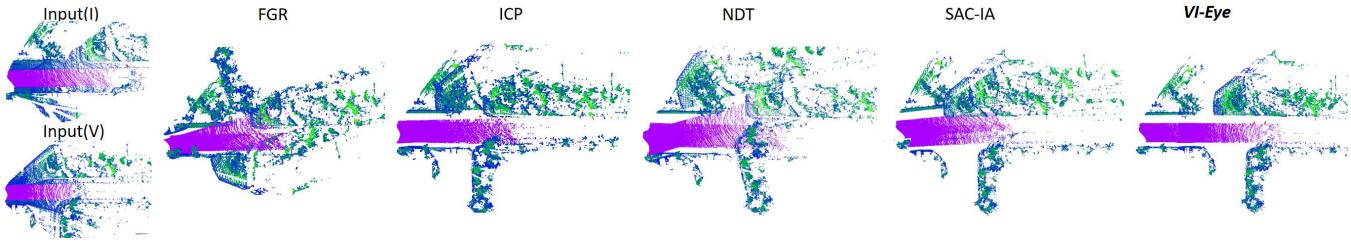



**Figure 13: A bird-eye view of the registration results in the real traffic scene. We crop unimportant points in results and only leave the points of the road (purple), curbs ans trees (blue and green) for better visualization.**

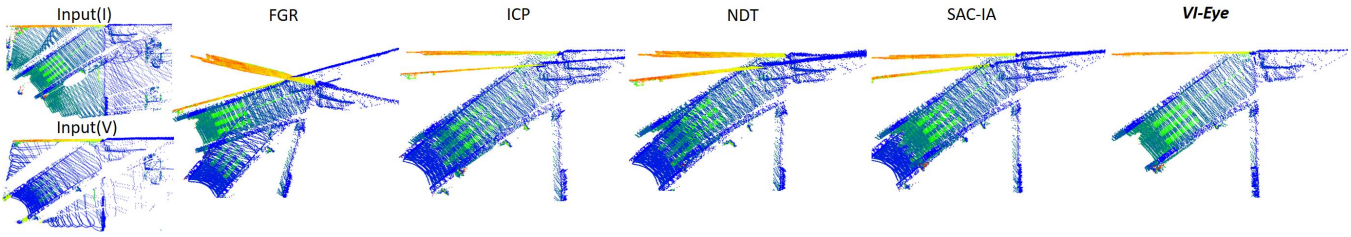



**Figure 14: A bird-eye view of the registration results on the indoor testbed. We crop unimportant points in results and only leave the points of the road and wall for better visualization.**

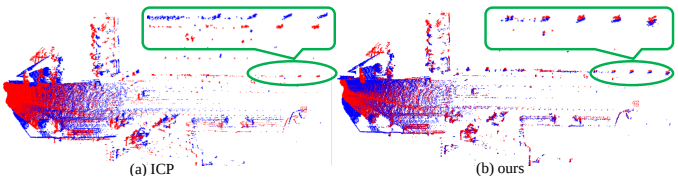



**Figure 15: A closer look at the qualitative results on the real traffic scene.**

**Table 6: Comparison with different key point extraction algorithms.**

| methods | #key points in point cloud (I) | #key points in point cloud (V) | #effective key point pairs | effective key point ratio |
|---|---|---|---|---|
| Harris3D | 145.65 | 142.72 | 30.69 | 21.60% |
| ISS3D | 904.09 | 962.78 | 294.25 | 33.09% |
| SIFT3D | 532.94 | 400.06 | 197.94 | 49.57% |
| ***VI-Eye*** | **21.94** | **19.78** | **17.56** | **71.76%** |

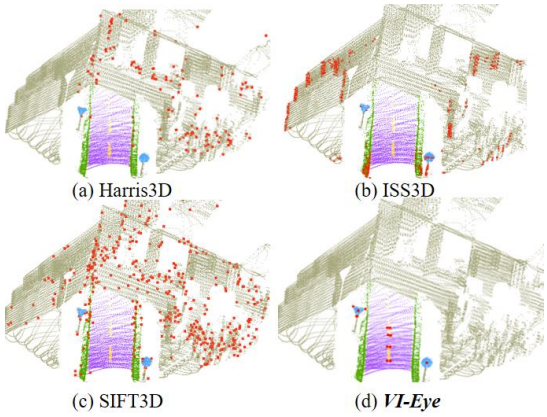



**Figure 16: Visualization of key points extracted by baseline methods and our saliency point extractor.**

the indoor dataset. This is because, to stress test *VI-Eye* we set up more scenes that are challenging for *VI-Eye* but hardly affect other baselines, such as the scenes with multiple closely located semantic objects or only one kind of semantic objects.

## 6.6 Performance of Saliency Point Extractor

We now evaluate the performance of saliency point extractor, one of the key modules in *VI-Eye*, by comparing with three widely used 3D keypoint extraction algorithms: Harris3D [44], ISS3D [55], and SIFT3D [24], which are implemented using PCL [39]. We compare with these method using two metrics: the average number of extracted key points in each point cloud and the effective key point ratio. The threshold in Equation (7) is set to 0.3m. Table 6 shows the results for the indoor simulated traffic dataset. We can see that the three baselines extracted over 100 or even hundreds of key points in vehicle and infrastructure point clouds, while only less than half of them are effective for registration. These baselines use the features such as the curvature and normal of the local point cloud to extract corner points or points on the edges. Although such an approach can be effective for CAD models [48], it performs poorly when applied to point clouds of complex traffic scenes, as these methods extract too many key points with similar features.

In contrast, *VI-Eye* extracts significantly fewer saliency points but achieves an effective key point ratio of more than 70%. We visualize the key points extracted by the three baselines and our saliency point extractor in Fig. 16. Key points extracted by the three baselines are scattered randomly in the scene, while the key points extracted by *VI-Eye* are highly interpretable: all geometric vertices or centers of the semantic objects to be used in registration.

**Table 7: Average run time of different modules of *VI-Eye* on the real traffic dataset**

| | modules | run time (ms) | proportion (%) |
|---|---|---|---|
| easy group | semantic segmenter | 139.56 | 64.02 |
| | saliency point extractor | 28.22 | 12.95 |
| | ground registration | 14.11 | 6.47 |
| | registration | 36.11 | 16.57 |
| hard group | semantic segmenter | 139.03 | 72.18 |
| | saliency point extractor | 22.83 | 11.85 |
| | ground registration | 11.42 | 5.93 |
| | registration | 19.33 | 10.04 |

## 6.7 Performance of Registration

In this subsection, we evaluate the performance of registration module. To verify that the proposed two extensions indeed accelerate the RANSAC algorithm, we conduct an ablation experiment on the campus dataset. Two metrics used to evaluate the registration performance include the average registration time and average number of iterations. We implement a baseline by replacing the registration module with the original RANSAC algorithm and leave all other modules unchanged.

Our registration module significantly reduces the average number of iterations from 4712 to 1547, which shows the effectiveness of the *two-tier label verification* that only matches saliency point pairs with consistent semantic label and primitive label. Another finding is that the reduction in average registration time is more drastic than the reduction in the number of iterations, i.e., from 1.74s to 0.21s. This shows the effectiveness of *early check*, which terminates iterations that are doomed to fail early to reduce unnecessary calculations, thus reduces the average time required for each iteration. The ablation experiments show that the proposed two extensions play an important role in achieving real-time registration.

## 6.8 Analysis of Run Time

To explore the possibility of further reducing the run time of *VI-Eye*, we measure the average run time of different modules of *VI-Eye* and their proportion of total run time in Table 7. The experiment is conducted on the real traffic dataset using a desktop PC with an Intel i7 2.90GHz CPU and an NVIDIA RTX2060Super GPU. We can find that the semantic segmenter's run time accounts for more than half of the overall run time. Unfortunately, it's challenging to reduce such delay since the run time of the state-of-the-art point cloud-based segmentation algorithms are around 0.1s to 1s according to KITTI benchmark [26]. The saliency point extractor and registration module have the second or third longest run time due to the point-wise and pixel-wise operations in the saliency point extractor and iterations in the registration module. Another observation is that the registration module's run time in the easy group is nearly twice that in the hard group, which is consistent with the results in Section 6.4. Our current implementation of *VI-Eye* is based on Python. The run time of the three modules can be further reduced by code optimization, e.g., by implementing them in C++, which is left for future work.

# 7 DISCUSSION

**Minimum Number of Saliency Points.** As a key advantage, *VI-Eye* is able to register vehicle-infrastructure point cloud pairs with small overlaps. The performance of *VI-Eye* depends on the number of sharing saliency points between the vehicle-infrastructure point clouds. According to Section 4.5, in principle, 3 saliency point pairs is the minimum requirement for *VI-Eye* to work. Note that a single sharing semantic object may have 1-5 saliency points (according to Section 4.3, depending on the shape and completion of sharing objects). Therefore, *VI-Eye* has a low requirement on the number of sharing objects between the infrastructure and the vehicle.

**Communication and Computation Overhead.** The communication overhead of *VI-Eye* depends on which fusion mode is used. For abstract information fusion, *VI-Eye* requires little bandwidth (about tens of KB/s) since only saliency points need to be transmitted. For raw data fusion, both infrastructure point clouds and saliency points need to be transmitted. The bandwidth required is about 12MB/s (for Livox Horizon LiDAR), which can potentially be further reduced using data compression algorithms since large amount of points are temporally redundant.

As discussed in Section 4, *VI-Eye*'s computation overhead is much lower than those of the baselines, which use all points in the point cloud to calculate the transform between the infrastructure and the vehicle. *VI-Eye* only uses a small part of the point cloud to calculate the transform. Specifically, the saliency point extractor module and the ground registration module are based on the point cloud of certain semantic objects, and the registration module only requires very few saliency points. Therefore, *VI-Eye* has the shortest run time compared with the baselines (Table 4 and Table 5).

# 8 CONCLUSION AND FUTURE WORK

In this paper, we present *VI-Eye*, the first system that aligns vehicle-infrastructure point cloud pairs at centimeter accuracy in real-time, which enables a broad range of on-vehicle autonomous driving applications. Evaluations on the two self-collected datasets show that *VI-Eye* outperforms state-of-the-art baselines in accuracy, robustness and efficiency.

We will address several limitation of *VI-Eye* in future work. First, the current design of *VI-Eye* only focuses on pairwise registration. However, it is not practical to register vehicle-infrastructure point cloud pairs for every frame considering the high run time overhead. A possible solution is to select key point cloud frames for registration while using the vehicle‘s odometry to update the transform in non-key point cloud frames. Second, the performance of *VI-Eye* is heavily dependent upon the accuracy of the saliency point extractor, which in turn relies on the performance of the semantic segmenter. In the future, we will extend the pure point cloud semantic segmentation to segmentation based on camera-LiDAR fusion since images allow for better detection and segmentation performance than point clouds.

# ACKNOWLEDGMENTS

This work is partially supported by Hong Kong Innovation and Technology Commission under grant GHP/126/19SZ.

# REFERENCES


[1] [n.d.]. Apollo. https://github.com/ApolloAuto/apollo.

[2] [n.d.]. Autoware. https://github.com/CPFL/Autoware.

[3] [n.d.]. Code for planning and design of urban road traffic. http://www.cacp.org.cn/zlbzgf/5172.jhtml.

[4] [n.d.]. Drop of LiDAR prices. https://compoundsemiconductor.net/article/111900/LiDAR_Dropping_Prices_And_Low_Volumes%7BfeatureExtra%7D.

[5] [n.d.]. F1TENTH. https://f1tenth.org/.

[6] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 2008. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH 2008 papers*. 1–10.

[7] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 2008. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH 2008 papers*. 1–10.

[8] Majd Alshawa. 2007. ICL: Iterative closest line A novel point cloud registration algorithm based on linear features. *Ekscentar* 10 (2007), 53–59.

[9] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. 2019. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7163–7172.

[10] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. International Society for Optics and Photonics, 586–606.

[11] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. International Society for Optics and Photonics, 586–606.

[12] Peter Biber and Wolfgang Straßer. 2003. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, Vol. 3. IEEE, 2743–2748.

[13] Gary Bradski and Adrian Kaehler. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

[14] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. 2002. The trimmed iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, Vol. 3. IEEE, 545–548.

[15] Christopher Choy, Wei Dong, and Vladlen Koltun. 2020. Deep global registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2514–2523.

[16] Christopher Choy, Jaesik Park, and Vladlen Koltun. 2019. Fully Convolutional Geometric Features. In *ICCV*.

[17] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 2018. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 195–205.

[18] Jianmin Dong, Yaxin Peng, Shihui Ying, and Zhiyu Hu. 2014. LieTrICP: An improvement of trimmed iterative closest point algorithm. *Neurocomputing* 140 (2014), 67–76.

[19] 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.

[20] Erik Einhorn and Horst-Michael Gross. 2015. Generic NDT mapping in dynamic environments and its application for lifelong SLAM. *Robotics and Autonomous Systems* 69 (2015), 28–39.

[21] G. Elbaz, T. Avraham, and A. Fischer. 2017. 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2472–2481. https://doi.org/10.1109/CVPR.2017.265

[22] 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 (June 1981), 381–395. https://doi.org/10.1145/358669.358692

[23] Andrew W Fitzgibbon. 2003. Robust registration of 2D and 3D point sets. *Image and vision computing* 21, 13-14 (2003), 1145–1153.

[24] A. Flint, A. Dick, and A. v. d. Hengel. 2007. Thrift: Local 3D Structure Recognition. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*. 182–188. https://doi.org/10.1109/DICTA.2007.4426794

[25] 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.

[26] 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.

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

[28] Pileun Kim, Jingdao Chen, and Yong K Cho. 2018. Automated point cloud registration using visual and planar features for construction environments. *Journal of Computing in Civil Engineering* 32, 2 (2018), 04017076.

[29] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. 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. https://doi.org/10.1109/JIOT.2018.2812300

[30] Jingnan Liu, Hangbin Wu, Chi Guo, Hongmin Zhang, Wenwei Zuo, and Cheng Yang. 2018. Progress and consideration of high precision road navigation map. *Strategic Study of Chinese Academy of Engineering* 20, 2 (2018), 99–105.

[31] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. 2019. DeepICP: An end-to-end deep neural network for 3D point cloud registration. *arXiv preprint arXiv:1905.04153* (2019).

[32] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 2017. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)* 36, 1 (2017), 3–15. https://doi.org/10.1177/0278364916679498 arXiv:http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+html

[33] Ellon Mendes, Pierrick Koch, and Simon Lacroix. 2016. ICP-based pose-graph SLAM. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 195–200.

[34] 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.

[35] François Pomerleau, Francis Colas, and Roland Siegwart. 2015. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics* 4, 1 (2015), 1–104.

[36] Alba Pujol-Miro, Josep R Casas, and Javier Ruiz-Hidalgo. 2019. Correspondence matching in unorganized 3D point clouds using Convolutional Neural Networks. *Image and Vision Computing* 83 (2019), 51–60.

[37] Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 145–152.

[38] 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.

[39] Radu Bogdan Rusu and Steve Cousins. 2011. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*. IEEE, 1–4.

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

[41] 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.

[42] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. 2009. Generalized-icp.. In *Robotics: science and systems*, Vol. 2. Seattle, WA, 435.

[43] 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.

[44] Ivan Sipiran and Benjamin Bustos. 2011. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer* 27, 11 (2011), 963–976.

[45] Masahiro Tomono. 2009. Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, 4306–4311.

[46] Yue Wang and Justin M Solomon. 2019. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3523–3532.

[47] Wikipedia contributors. 2021. Braking distance — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Braking_distance&oldid=1004144687 [Online; accessed 25-March-2021].

[48] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

[49] Jiaolong Yang, Hongdong Li, and Yunde Jia. 2013. Go-icp: Solving 3d registration efficiently and globally optimally. In *Proceedings of the IEEE International Conference on Computer Vision*. 1457–1464.

[50] Zi Jian Yew and Gim Hee Lee. 2018. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 607–623.

[51] Zi Jian Yew and Gim Hee Lee. 2020. Rpm-net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11824–11833.

[52] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 2017. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1802–1811.

[53] Zhenxin Zhang, Lan Sun, Ruofei Zhong, Dong Chen, Zhihua Xu, Cheng Wang, Cheng-Zhi Qin, Haili Sun, and Roujing Li. 2019. 3-D deep feature construction for mobile laser scanning point cloud registration. *IEEE Geoscience and Remote Sensing Letters* 16, 12 (2019), 1904–1908.

[54] 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.

[55] Yu Zhong. 2009. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 689–696.

[56] Lei Zhou, Siyu Zhu, Zixin Luo, Tianwei Shen, Runze Zhang, Mingmin Zhen, Tian Fang, and Long Quan. 2018. Learning and matching multi-view descriptors for registration of point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 505–522.

[57] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2016. Fast global registration. In *European Conference on Computer Vision*. Springer, 766–782.

[58] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018).