ERS: Faster LiDAR Point Cloud Registration for Connected Vehicles

Yu Zhao*, Jinrui Zhou*, Mingjun Xiao*, Jie Wu† and He Sun*
*School of Computer Science and Technology, University of Science and Technology of China, P.R. China
†‡Department of Computer and Information Sciences, Temple University, USA
* {zhaoyu0624@mail., zzkevin@mail., xiaomj@, hesun@mail.}ustc.edu.cn, † jiewu@temple.edu

***Abstract*—The registration of LiDAR point clouds between Connected Vehicles (CVs) is the basis for vision tasks such as multi-vehicle cooperative perception and high-precision scene reconstruction. Due to the huge amount of LiDAR point cloud data and the complexity of computation, the registration has high transmission latency and execution time. Fast and accurate point cloud registration between CVs is urgently required by safe driving applications. To achieve faster registration, we propose partitioning overlapping point clouds for registration in advance. Meanwhile, for higher accuracy, we incorporate more geometric features during the registration. This paper demonstrates the Echo Registration System (ERS) developed to increase the speed of existing point cloud registration methods. ERS employs an initial positioning based on point cloud distributions, a density-consistent partition strategy, and a virtual geometric feature fusion. We deploy ERS to accelerate multiple state-of-the-art registration methods. The comprehensive results show that ERS improves the overall running time of current state-of-the-art baselines by 5.7× with 42.1% registration recall gains.**

***Index Terms*—Point Cloud, Registration, Connected Vehicles**

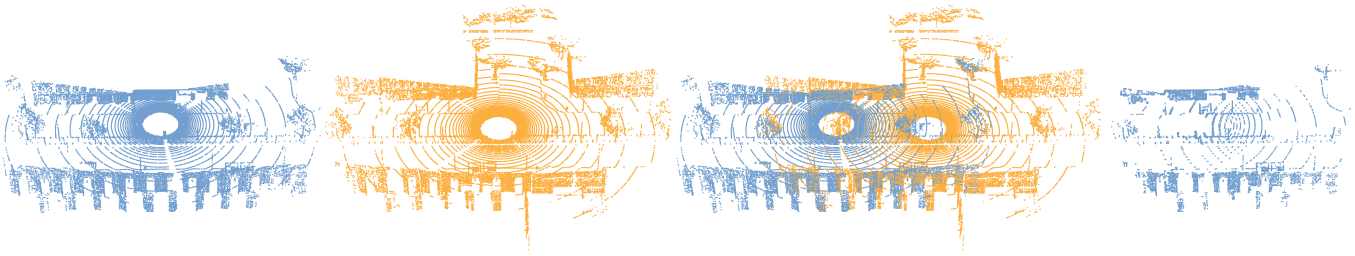
I. Introduction

Point cloud registration facilitates precise 3D scene reconstruction and serves as a foundational element for multi-vehicle cooperative driving [1], contributing to the enhancement of safety in driving applications [2], [3]. Take two Connected Vehicles (CVs) as an instance, we can merge 3D scans of the same scene into a more complete 3D point cloud. Fig. 1(a) shows the source point cloud generated by source vehicle, and Fig. 1(b) shows the target point cloud generated by target vehicle. Through point cloud registration, we transform the target point cloud, and then integrate these point clouds, as shown in Fig. 1(c). In this case, fast and accurate point cloud registration is imperative to facilitate subsequent tasks, including cooperative perception and path planning.

However, point cloud registration between CVs poses more challenges compared to the registration of a single vehicle. In LiDAR SLAM [4], registration is typically carried out on point clouds captured by a single vehicle at adjacent moments. In such scenarios, the point clouds have large overlapping areas and low noise levels, rendering the registration process relatively straightforward. However, when dealing with registration between CVs, the conditions are reversed, with smaller overlap regions and higher noise levels, posing challenges. This results in increased computational overhead and decreased accuracy during the registration process. Despite fundamental works such as KPConv [5] and MinKowski Engine [6] which were developed to expedite the computation of 3D points, point cloud registration between CVs is still a computationally intensive task. Additionally, registration between CVs involves transmitting point clouds. Considering the widely used Velodyne LiDAR HDL-64E, which captures over 100K points in a single scan [7], this volume of data results in significant transmission latency. Nonetheless, existing approaches for point cloud feature extraction involve extracting high-dimensional features or utilizing the cross-attention mechanism, which cannot reduce the transmitted data. Overall, point cloud registration between CVs faces challenges in meeting real-time requirements and requires improvements in accuracy. These challenges are exacerbated by the limited availability of onboard computing resources.

In response to the aforementioned challenges, we consider partitioning overlapping point clouds for registration, thereby reducing end-to-end running time, which includes execution time and transmission latency. Currently, several methods identify overlapping regions between point clouds and subsequently align these regions [8], [9]. However, these approaches still necessitate the representation of all raw point clouds, without effectively reducing the volume of point clouds. Hence, the first challenge lies in devising a solution to identify potential overlapping regions without the need to transmit and process entire raw point clouds. Additionally, noise still persists in the identified overlapping regions, impacting both speed and accuracy of the registration. Therefore, it becomes imperative to devise a partition strategy which includes a comprehensive evaluation of diverse points aiming to identify and retain points that make positive contributions to the registration process. While partitioning the point cloud serves to decrease running time and noise, it may result in diminished accuracy owing to the reduced information in special cases. Therefore, we also need to consider how to improve accuracy.

To achieve faster and more accurate registration, we propose the Echo Registration System (**ERS**), a framework to accelerate LiDAR point cloud registration between CVs. Echo represents the information returned to target vehicle during the registration, which determines the point cloud transmitted by the target vehicle. This framework is conceived as a plug-and-play extension to existing point cloud registration methods, demanding minimal modifications at the application level. ERS is implemented in three integral components. To identify over-



(a) Source point cloud. (b) Target point cloud. (c) Point cloud after registration. (d) Overlapping points of the source point cloud.

Fig. 1: The source point cloud and target point cloud can be merged into a unified point cloud in the same coordinate system through registration. In this process, only the overlapping points can be effective to registration.

lapping regions, we aim to approximate the relative positions between CVs which we achieve through a lightweight initial positioning based on point cloud distributions. Afterward, ERS formulates a density-consistent partition strategy, which is then transmitted as an echo to the target vehicle. Third, to ensure accuracy, ERS employs the virtual geometric feature fusion. The evaluations of ERS on current state-of-the-art point cloud registration methods are conducted to validate its efficiency.

The main contributions of the article are as follows.

- To the best of our knowledge, we are the first to introduce an accelerated point cloud registration framework between CVs, which strategically partitions LiDAR point clouds, aiming to mitigate end-to-end running time.
- To ascertain valid points within possible overlapping regions, we have developed an innovative initial positioning based on point cloud distributions and a density-consistent partition strategy.
- To ensure the accuracy of registration, we integrate virtual geometric features into these point clouds enhancing correspondences between the two point clouds.
- We implemented ERS on four state-of-the-art point cloud registration methods and compared their performance with the raw methods without incorporating ERS. The results demonstrate that ERS effectively reduces end-to-end running time and enhances registration accuracy.

II. Background and Motivation

In this section, we present the formulation of the point cloud registration problem and the solution involving a transformation matrix. This process encompasses correspondence search and transformation estimation. We then delve into examinations of how various points influence the registration.

A. Formulation of the point cloud registration problem

Point cloud registration aligns point clouds by estimating a rigid transformation matrix T. This matrix consists of R and t, with $R \in \mathcal{SO}(3)$ denoting the rotation matrix and $t \in \mathbb{R}^3$ denoting the translation vector. $\mathcal{P} \in \mathbb{R}^{M\times 3}$ and $\mathcal{Q} \in \mathbb{R}^{N\times 3}$ represent source and target point clouds, and we denote p_i ($i \in [1, M]$) and q_j ($j \in [1, N]$) as raw vectors from matrices $\mathcal{P}$ and $\mathcal{Q}$ respectively. In these point clouds, p_i and q_j are the coordinates of the i_{th} and j_{th} points, which both include x, y and z.

Thus, the point cloud registration problem can be formulated as:

$$\underset{R\in\mathcal{SO}(3),t\in\mathbb{R}^3}{min} \left\| \widetilde{\mathcal{P}} - (R\widetilde{\mathcal{Q}} + t) \right\|_2, \tag{1}$$

where $\widetilde{\mathcal{P}}$ and $\widetilde{\mathcal{Q}}$ are matrices consisting of corresponding points selected from the point clouds $\mathcal{P}$ and $\mathcal{Q}$. However, the correspondences between the two point clouds are unknown and must be determined through the correspondence search. We use the mapping function $\mathcal{F}$ to denote the features extracted from points. If p_i and q_j are a pair of corresponding points, $\mathcal{F}(p_i)$ is similar to $\mathcal{F}(q_j)$, i.e. $\|\mathcal{F}(p_i) - \mathcal{F}(q_j)\| < f_{thr}$, where f_{thr} is a threshold for feature distances.

The transformation estimation can be solved using particular algorithms, with the Random Sample Consensus (RANSAC) algorithm being most commonly employed. The RANSAC algorithm estimates a transformation matrix by randomly sampling a subset of corresponding point pairs. This matrix is then used to assess the remaining data, with a point considered as an inlier if it conforms to the estimated transformation matrix. The identified inlier points are subsequently employed to re-estimate a transformation matrix, which is further utilized to validate the remaining points. This iterative process is repeated to ultimately obtain an optimal transformation matrix. RANSAC is extensively employed in transformation estimation due to its effectiveness in handling noise within the data. However, the outliers augment the number of iterations needed by the RANSAC algorithm, resulting in increased excution time and decreased accuracy.

B. A Motivation Study

Impact of different types of point clouds. Here, we will mainly discuss three types of point clouds, ground point clouds, non-ground point clouds, and point clouds of counterpart vehicles, which we refer to as *counterpart point clouds*, as shown in Fig. 2. The counterpart vehicle represents a vehicle that interacts with the self vehicle in the current registration. In this figure, the counterpart point cloud denotes the point cloud of target vehicle, as observed by the source vehicle. Upon analyzing the point cloud, it becomes evident

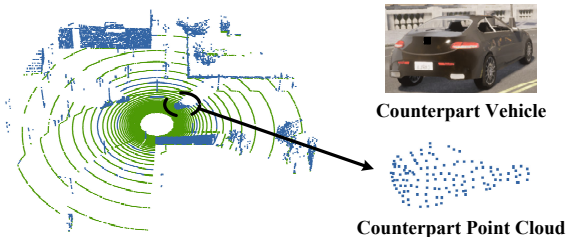


Fig. 2: This point cloud is split into the ground point cloud (green) and the non-ground point cloud (blue). The non-ground point cloud comprises the counterpart point cloud.

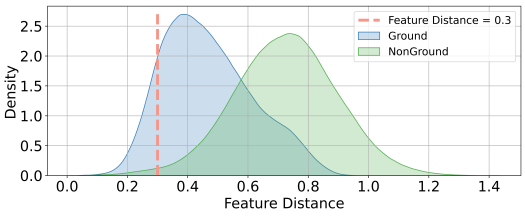
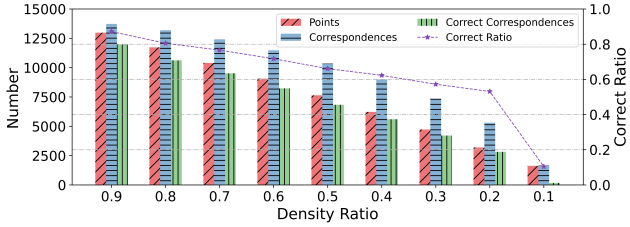
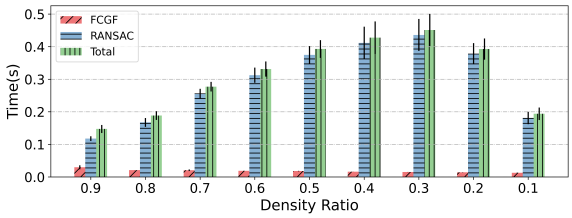


Fig. 3: Distribution of feature distances for ground and non-ground point clouds.



(a) The number of correspondences and correct correspondences.



(b) The execution time.

Fig. 4: Impact of density ratio between target and source point clouds on registration.

that the majority of points are distributed on the ground. For instance, in the case of the source point cloud in Fig. 2, about 76.15% of the points are ground points. As LiDAR generates different scans without obstructions, the ground is detected, resulting in concentric circles composed of points. The distance between two adjacent arcs increases with the distance from the center of the circle. Each ground point cloud appears to behave similarly. Consequently, similarity in the ground point clouds obtained from different locations may result in false correspondences, thereby increasing running time and decreasing accuracy.

To verify the similarity of ground point features, we present the distributions of Euclidean distances from the ground point cloud features and the non-ground point cloud features, which are estracted by FCGF [10], as illustrated in Fig. 3. The Euclidean distance disparities between ground point cloud features are significantly smaller than those of non-ground point clouds. In other words, the features of ground points exhibit greater similarity to each other, resulting in a higher likelihood of false matches during correspondence search. When analyzing feature distances among ground point clouds, approximately 12.15% of the points fall below the feature distance threshold of 0.3 set by the RANSAC algorithm. In contrast, in the non-ground point cloud, this percentage is only 1.06%. While the ground point clouds exhibit significant similarity, they contribute positively to determining the pitch angle, roll angle, and translation of the vehicle along the z-axis direction. In this paper, we specify that a right-handed coordinate system is used and that the x-axis is directed in the forward direction of the vehicle.

In Fig. 2, the counterpart point cloud, representing the counterpart vehicle, struggles to find correspondences. If the counterpart point clouds are utilized effectively, the number of searched correspondences can be increased, thus improving the accuracy of the registration.

Impact of different densities. Varying densities of point clouds at the same location introduce differences in features. The more similar the densities between two point clouds in the same region, the more consistent their features will be.

We randomly sample the same source point cloud multiple times to generate target point clouds with different densities. Registration using the FCGF [10] and the RANSAC algorithm is then conducted to find the impact of varying target point cloud densities on registration. We define the *density ratio* as the ratio of the number of points in the target point cloud to the number of points in the source point cloud within a given region. In Fig. 4(a), we illustrate the impact of varying densities in the target point cloud on the number of correspondences. It is unsurprising to observe a decrease in the number of correspondences as the density ratio decreases. Noteworthy is the more rapid decline in the number of correct correspondences. The *correct ratio* is defined as the ratio of the number of correct correspondences to the total number of correspondences. When the density ratio is 0.3, the correct ratio drops to 57.2%. At a density ratio of 0.1, the correct ratio decreases to around 10%. In Fig. 4(b), we compare the impact of various density ratios on the execution time. For a density ratio of 0.9, the RANSAC algorithm exhibits the shortest execution time, as it reaches the termination condition earlier. Additionally, the total time increases as the density ratio decreases. Even though the execution time decreases at low density ratios, it remains higher than the time at the density ratio of 0.9. In conclusion, at low density ratios, the extracted features differ more, potentially exerting a negative influence on point cloud registration.

Design Challenges. Designing the ERS presents several key challenges. The primary challenge involves identifying overlapping regions between two point clouds. To determine

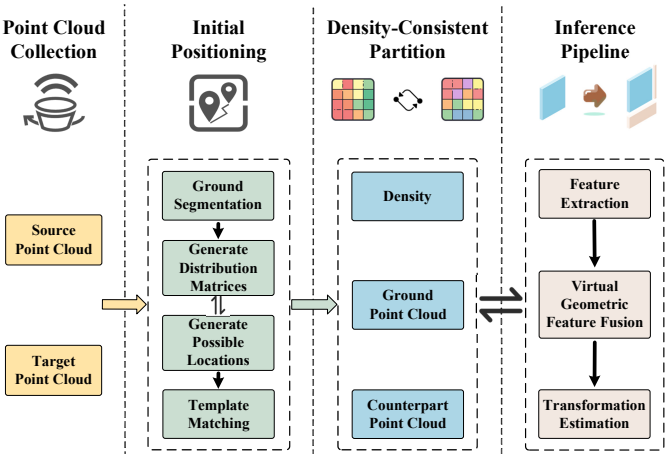


Fig. 5: ERS overview.

the overlapping regions, this can be done by determining the approximate relative positions of the CVs in advance. Ideally, GPS positioning can be leveraged to determine relative positions between CVs. However, in the case of consumer-grade sensors operating in multi-path environments like urban canyons, assuming an offset distance of 5 to 20 meters is a realistic assumption [11]. This assumption, however, introduces difficulty in determining the approximate relative positions of vehicles to the extent that overlapping regions become indeterminable. And existing point cloud registration methods typically identify overlapping regions based on multiple downsamplings and features extracted from the raw point cloud, introducing additional overhead. Hence, ERS must include a lightweight method for identifying possible overlapping regions.

The second challenge involves effectively utilizing different types of point clouds to determine the partition strategy. While ground point cloud features exhibit similarity, they are essential for the registration. Furthermore, even in regions where two non-ground point clouds overlap, there may be redundant points to generate noise. Thus, ERS needs to incorporate a partition strategy to prioritize points crucial for registration. Additionally, ensuring registration accuracy while partitioning partial point clouds poses another challenge. The simplification of the point cloud inevitably decreases the number of inliers in the RANSAC algorithm. How to effectively utilize the counterpart point cloud to increase the number of correspondences during registration is another challenge.

III. System Architecture

ERS aims to tackle the design challenges through the following steps:

(1) Initial positioning based on point cloud distributions. ERS transforms the initial positioning problem into a template matching problem by aligning the point cloud distribution matrices. This alignment helps determine the initial transformation matrix. The approach is lightweight and proves effective for approximate positioning.

(2) Density-consistent partition strategy. Given the initial transformation matrix, ERS identifies correspondences between regions in the source and target point clouds. The partition decision is determined based on the density-consistent principle. Moreover, ERS simplifies the ground point cloud while preserving observations about the counterpart vehicle.

(3) Virtual geometric feature fusion. During the registration process, ERS further integrates virtual geometric features with those features extracted from the partitioned point cloud. These geometric features primarily originate from the vehicle's external shape, obtained through methods such as CAD modeling or 3D sensor data collection.

The main workflow is illustrated in Fig. 5. Initially, the vehicle segments the point cloud into ground and non-ground components. Notably, the non-ground point cloud is segmented into low and high point clouds. ERS further divides these point clouds into regions, each represented as a mesh. Subsequently, the system counts the number of points within each mesh to generate three matrices reflecting the point cloud distributions. The target vehicle compresses and transmits these distribution matrices to the source vehicle. By utilizing the distribution matrices of the ground point cloud, possible locations of the counterpart vehicle are generated. Following this, a two-layer template matching is executed using the distribution matrices of the non-ground point cloud. ERS identifies the position and rotation angle with the highest correlation, obtaining the initial transformation matrix.

Given the initial transformation matrix, ERS identifies the overlapping regions of the source and target point clouds. To address the unique challenges posed by the point cloud registration, the partition strategy must adhere to the following principles: (1) refine overlapping regions to further reduce the amount of data transmitted; (2) retain the specific counterpart point cloud; (3) simplify the ground point cloud with similar features. In response, ERS excutes a density-consistent partition strategy. The generated result is returned to target vehicle as an echo. Subsequently, target vehicle transmits the requested partial point clouds to the source vehicle based on the echo.

After receiving the point cloud transmitted by target vehicle, the deployed point cloud registration methods extract features from the point cloud. To enhance the accuracy of the registration, ERS introduces virtual geometric feature fusion. We construct geometric features by collecting the point cloud of the vehicle's external shape in advance. Integrating these geometric features as virtual points into the extracted features effectively increases the number of correct correspondences. The transformation estimation of these aggregated features is performed to obtain the transformation matrix.

IV. Methodology

This section details how ERS accelerate point cloud registration and ensures its accuracy. We introduce a lightweight initial positioning based on point cloud distributions, describe the density-consistent partition strategy, and detail how to construct and integrate virtual geometric features.

A. Initial Positioning based on Point Cloud Distributions

We construct the distribution matrices based on point clouds, present the generation of possible locations, and detail the template matching.

1) Generate Distribution Matrices: For the point cloud $\mathcal{P}$ captured by the vehicle's LiDAR, ERS executes ground segmentation to obtain ground point cloud $\mathcal{G}$ and non-ground point cloud $\hat{\mathcal{G}}$, where $\mathcal{G}$ and $\hat{\mathcal{G}}$ also satisfy $\mathcal{G} \cup \hat{\mathcal{G}} = \mathcal{P}$. Furthermore, based on the values of the points in the z-axis, we partition the non-ground point cloud into non-ground low point cloud $\hat{\mathcal{G}}^l$ and non-ground high point cloud $\hat{\mathcal{G}}^h$:

$$\begin{aligned} \hat{\mathcal{G}}^l &= \{p_k \in \mathcal{P} \mid z(p_k) < z_{init} + z_{thr}\}, \\ \hat{\mathcal{G}}^h &= \{p_k \in \mathcal{P} \mid z(p_k) \geq z_{init} + z_{thr}\}, \end{aligned} \tag{2}$$

where $z(\cdot)$ returns the z value of a point, z_{init} denotes the height of the LiDAR and z_{thr} donotes a defined threshold. This partition is motivated by the fact that non-ground high point clouds are generally more stable, while low point clouds are susceptible to factors such as pedestrians and vehicles. Subsequently, we divide the point cloud into meshes based on the (x, y) values of the points, and count the number of points in each mesh. Similarly to the classification representation of the point cloud, we denote three matrices from the ground point cloud, the non-ground low point cloud, and the non-ground high point cloud as M, $\hat{M}^l$, and $\hat{M}^h$, respectively. As the number of points observed by LiDAR decreases with distance, ERS employs larger area for long-distance meshes to better represent the distribution of ground point clouds. Dividing according to the mesh makes it easier to transform these matrices. Since all three matrices are sparse, target vehicles compress them to reduce the amount of data transmitted.

2) Generate Possible Locations: The distribution matrices are received, and ERS further processes them. Based on our observations, we find that vehicles in the perceptual range create gaps in the ground. Consequently, possible locations are determined by identifying gaps in the ground distribution matrix. As depicted in Fig. 6, owing to the inherent sparseness of LiDAR for long-range observation points, numerous gaps will be present at long distances. Moreover, ground gaps created by vehicles may surpass the vehicles' coverage regions significantly. To address this, for the ground distribution matrix, we devise a center-oriented Gaussian blur, which is one-dimensional. Let's consider a mesh E in the ground distribution matrix, and the center of the distribution matrix O. The center-oriented Gaussian blur fills the points on the line EO from the outside inward. Thus, the purpose of the blur is to fill in ground gaps at long distances and reduce the extent of vehicle occlusion gaps.

ERS further narrows down the number of possible locations based on the non-ground distribution matrices. An obstruction by a non-ground object must exist in the line between the possible location and the center. Therefore, we connect each gap to the center point. On the side close to the gap, ERS detects the presence of non-ground low objects. If non-ground low point cloud occlusion occurs, the ground gap can be considered

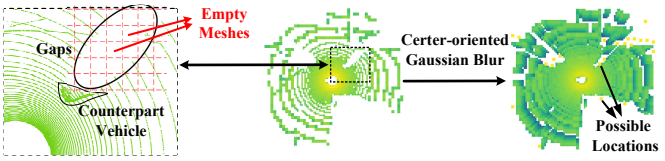


Fig. 6: Left: the counterpart vehicle generates a larger gap than its coverage area. Middle: the ground point cloud distribution matrix, where numerous gaps exist. Right: the ground matrix after Gaussian blur and the obtained possible locations of the counterpart vehicle.

a possible location. This enables further filtering of possible locations. Assuming that n possible positions are generated, the set can be expressed as $\mathcal{C} = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$. The sets of possible locations of the source and target point clouds are denoted as $\mathcal{C}_s$ and $\mathcal{C}_t$. The subscripts s and t represent the source and target point clouds, respectively, and the same applies in the following.

3) Template Matching: For each possible location, ERS calculates the distance between it and the center, and then maintains two distance sets: $\mathcal{D}_s = \{d_s^1, d_s^2, \ldots, d_s^{|\mathcal{C}_s|}\}$ and $\mathcal{D}_t = \{d_t^1, d_t^2, \ldots, d_t^{|\mathcal{C}_t|}\}$. ERS seeks pairs of points in the two sets with similar distances to the center point, indicating that the difference in distance is less than a specific threshold d_{thr}. If a pair of points is found, i.e., $|d_s^i - d_t^j| < d_{thr}$, we calculate the rotation angle and translation vector between them based on these two possible locations (x_s^i, y_s^i) and (x_t^j, y_t^j) as:

$$\begin{aligned} \delta_{ij} &= arctan(-y_s^i, -x_s^i) - arctan(y_t^j, x_t^j), \\ t_{ij} &= ((x_s^i - x_t^j)/2, (y_s^i - y_t^j)/2). \end{aligned} \tag{3}$$

Rotating the matrices of the target point cloud enables the coordinates of the pair of possible locations to be approximately symmetric about the center. Subsequently, by translating the matrices of the target point cloud, the coordinates of this pair of points become close to the center of their counterpart matrices, respectively. This implies that the possible location of the source point cloud is in proximity to the center of the target point cloud, and the possible location of the target point cloud is close to the center of the source point cloud. The rotation of the matrix approximates the rotation of the point cloud by randomly generating a specific number of points in each mesh. These points undergo rotation, and then the number of points within each mesh is recorded to generate a new matrix.

After obtaining the K pairs of correspondences among the possible locations, we implement two layers template matching on the non-ground point cloud matrices. The first layer of template matching is carried out on the distribution matrices $\hat{M}_s^h$ and $\hat{M}_t^h$. First, as illustrated in Fig. 7, we compute the initial transformation based on the correspondence of possible points and transform the distribution matrices $\hat{M}_s^h$ and $\hat{M}_t^h$ to $\hat{M}_s^{h'}$ and $\hat{M}_t^{h'}$, respectively. At this stage, two-dimensional Gaussian blur is applied to both matrices before each template matching to mitigate the impact of the initial transformation

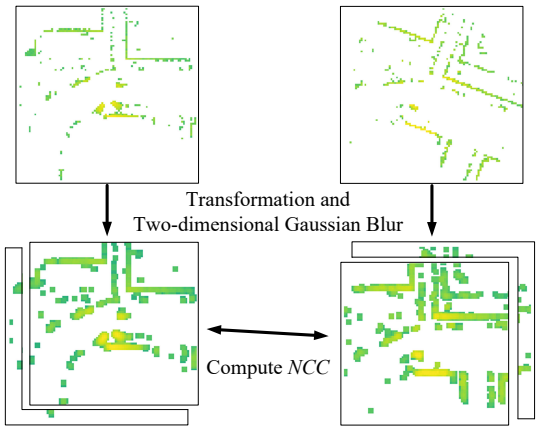


Fig. 7: Upper left: the distribution matrix $\hat{M}_s^l$. Upper right: the distribution matrix $\hat{M}_t^l$. Lower left: the distribution matrix $\hat{M}_s^{l'}$. Lower right: the distribution matrix $\hat{M}_t^{l'}$. The separated matrices represent the overlapping regions. Non-ground point clouds share the same process.

error and to enhance the matching. Matching is exclusively carried out on overlapping regions. The mesh values undergo binarization to alleviate the influence of point density. ERS calculates the Normalized Cross Correlation (NCC) between the two matrices in overlapping regions as follows:

$$NCC(\hat{M}_s^{h'}, \hat{M}_t^{h'}) = \frac{\sum_{x,y}(\hat{M}_s^{h'}(x,y) \cdot \hat{M}_t^{h'}(x,y))}{\sqrt{\sum_{x,y}\hat{M}_s^{h'}(x,y)^2 \cdot \sum_{x,y}\hat{M}_t^{h'}(x,y)^2}}. \tag{4}$$

ERS then selects the top-k entries with the highest NCC values. Subsequently, utilizing the distribution matrices of the non-ground low point cloud, the second layer of template matching is applied to these pairs of possible locations. Considering the potential errors in the initial transformation matrix, ERS extends one rotation angle to obtain several similar values through sampling at regular intervals, from which the best matching angle is selected to refine the initial transformation. The non-ground low matrix is then transformed and subjected to Gaussian blur for matching. Based on the cumulative NCC from the two matching processes, ERS identifies the possible location with the highest value. Further details on the template matching process are provided in Algorithm 1.

Algorithm 1: Template Matching.

Input: Distribution matrices: $\hat{M}_s^h$, $\hat{M}_s^l$, $\hat{M}_t^h$, $\hat{M}_t^l$; Possible locations sets: $\mathcal{C}_s$, $\mathcal{C}_t$; Distance sets: $\mathcal{D}_s$, $\mathcal{D}_t$;
Output: Initial rotation angle, δ; Initial translation, t;

STEP-1: Find Possible Locations Correspondences;
Initialize $\mathcal{C}_{st}$;
for $i = 1$ *to* $|\mathcal{D}_s|$ **do**
  for $j = 1$ *to* $|\mathcal{D}_t|$ **do**
    if $|d_s^i - d_t^j| < d_{thr}$ **then**
      Insert $(\mathcal{C}_s[i], \mathcal{C}_t[j])$ into $\mathcal{C}_{st}$;
STEP-2: First Layer Template Matching;
Initialize NCC set $\mathcal{N}1$ and $\mathcal{N}2$;
Perform Gaussian blur on $\hat{M}_s^h$ and $\hat{M}_s^l$;
for $k = 1$ *to* $|\mathcal{C}_{st}|$ **do**
  Compute δ_k and t_k by $\mathcal{C}_{st}[k]$;
  Transform and perform Gaussian blur to take $\hat{M}_s^h$, $\hat{M}_t^h$ to $\hat{M}_s^{h'}$, $\hat{M}_t^{h'}$;
  Insert $NCC(\hat{M}_s^{h'}, \hat{M}_t^{h'})$ into $\mathcal{N}1$;
Select NCC from $\mathcal{N}1$ as $\mathcal{N}1'$ and corresponding coordinates from $\mathcal{C}_{st}$ as $\mathcal{C}_{st}'$ via top-k selection;
STEP-3: Second Layer Template Matching;
for $k = 1$ *to* $|\mathcal{C}_{st}'|$ **do**
  Compute δ_k^0 and t_k by $\mathcal{C}_{st}'[k]$;
  Extend δ_k^0 to $[\delta_k^0, \delta_k^1, \delta_k^2, ..., \delta_k^{w-1}]$;
  for δ_k *in* $[\delta_k^0, \delta_k^1, \delta_k^2, ..., \delta_k^{w-1}]$ **do**
    Transform and perform Gaussian blur to take $\hat{M}_s^l$, $\hat{M}_t^l$ to $\hat{M}_s^{l'}$, $\hat{M}_t^{l'}$;
    Insert $\mathcal{N}1'[k] + NCC(\hat{M}_s^{l'}, \hat{M}_t^{l'})$ into $\mathcal{N}2$;
Select the highest NCC from $\mathcal{N}2$ and corresponding δ and t;
Return: δ, t;

B. Density-consistent Partition

After obtaining the initial transformation, ERS determines the partition strategy, taking into account three factors: density, counterpart point cloud, and ground point cloud. We will now delve into each of these aspects.

Density. ERS efficiently partitions point clouds based on density to reduce computational and transmission overhead. The strategy is developed using the distribution matrices of the point cloud, with the density matrix $\hat{M}^l$ and the density matrix $\hat{M}^h$ merged into a unified non-ground distribution matrix denoted as $\hat{M}$. Due to inherent errors in the initial transformation, $\hat{M}$ adopts a larger mesh length. Following this, the matrix $\hat{M}_t$ of the target point cloud is rotated and translated based on the initial transformation. The corresponding meshes with density ratios between $1/dt$ and dt are then selected by comparing $\hat{M}_s$ and $\hat{M}_t$:

$$1/dt \leq \ \hat{M}_t(i,j)/\hat{M}_s(i,j) \leq dt, \tag{5}$$

where dt denotes the density threshold. This part of point cloud is referred to as the *partitioned point cloud*.

Counterpart point cloud. Observations of the vehicle generating the target point cloud will be present in the source point cloud, and vice versa. As LiDAR cannot observe the appearance of its self vehicle, the counterpart point cloud is unable to establish the necessary correspondences, rendering it a redundant part of the registration process. However, given the stable nature of the external shape of the vehicle, effective utilization of this point cloud has the potential to increase the number of correspondences, particularly the correct ones. In the partition strategy, the counterpart point cloud is typically filtered out due to the mismatch in density. Depending on the

selected possible location, ERS retains the counterpart point cloud by reserving a specific range of the possible location.

Ground point cloud. As ground points exhibit similar features, ERS selectively retains only the normal and height, i.e., the mean value along the z-axis of the ground point cloud. Following the approach outlined in [12], we calculate the covariance matrix of the ground point cloud, denoted as $A \in \mathbb{R}^{3\times3}$. The three eigenvectors are computed as follows:

$$A\ v_\alpha = \lambda_\alpha\ v_\alpha, \tag{6}$$

where $\alpha = 1, 2, 3$, and assume $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The eigenvector v_3 corresponding to the smallest eigenvalue is employed to represent the normal vector of the ground. This normal vector is utilized to initiate the rotation of the point cloud in the x-axis and y-axis, corresponding to the pitch and roll angles of the vehicle. Additionally, the ground height z^g is embedded as a feature in the points, allowing for the filtration of points with significant differences in height.

ERS establishes an echo transmission back to the target vehicle, implementing the aforementioned strategy. The target vehicle simplifes point cloud based on the echo and transmit it to the source vehicle. Redundant points that are not valid for the registration would not be transmitted. Density-consistent partition significantly reduces transmission latency compared to transmitting all raw point clouds, and ensures the acquisition of reliable correspondences in the registration.

C. Virtual Geometric Feature Fusion

In this subsection, we elaborate on enhancing the accuracy of the registration by augmenting the quantity and quality of correspondences through the utilization of the counterpart point cloud and height information.

Geometric feature construction. First, it is imperative to build geometric point clouds representing the external shape of vehicles. This can be achieved through either CAD modeling or LiDAR data acquisition. Here, we focus on outlining the procedure for constructing geometric features from LiDAR-acquired point clouds in advance. Specifically, we gather point clouds using LiDAR from various positions surrounding a vehicle, designating these LiDAR devices as *reference LiDARs*. Simultaneously, we record the transformation matrices that describe the relationships between these reference LiDARs and the LiDAR installed on the vehicle. These transformation matrices enable the conversion of the collected point clouds to a coordinate system centered on the vehicle's LiDAR. Subsequently, these transformed point clouds are regarded as observed by the LiDAR installed on the vehicle, rather than by the reference LiDARs. These points are designated as the virtual geometric point cloud $\mathcal{V}$. Next, features are extracted from the virtual geometric point cloud, resulting in the virtual geometric features $\mathcal{V}^f = \mathcal{F}(\mathcal{V})$, as illustrated in Fig. 8.

Virtual geometric feature fusion. ERS requires performing virtual geometric feature fusion, which involves introducing correspondences to the counterpart point cloud. The target vehicle transmits the virtual geometric point cloud to the source vehicle, and then features are extracted as virtual geometric features. Subsequently, these features are concatenated with the features from the partitioned point cloud. Taking the partitioned point cloud $\dot{\mathcal{P}}$ as an example, the concatenation procedure to generate integrated point cloud $\dot{\mathcal{P}}'$ is as follows:

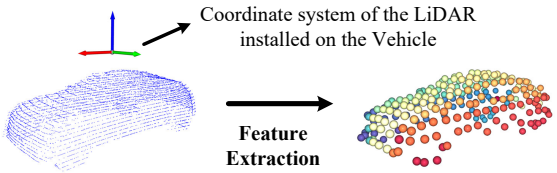


Fig. 8: Virtual geometric point cloud (left) and virtual geometric features (right) through FCGF.

$$\begin{aligned} \dot{\mathcal{P}}' &= Concat(\dot{\mathcal{P}}, \mathcal{V}_s), \\ \mathcal{F}(\dot{\mathcal{P}}') &= Concat(\mathcal{F}(\dot{\mathcal{P}}), \mathcal{V}_s^f). \end{aligned} \tag{7}$$

Instead of integrating the entire virtual point cloud, we trim it based on the initial positions obtained to avoid adding mismatched points. The concatenation effectively increases the number of correspondences between the two point clouds, as shown in Fig. 9.

However, the fusion inevitably increases the computational overhead of correspondence search, as well as the number of mismatched correspondences. ERS further optimizes the registration by extending the feature dimension. For instance, FCGF generates 32-dimensional features, and ERS expands them to 33 dimensions. ERS assigns a fixed value to the additional dimension incorporated into the features of counterpart point cloud and the virtual geometric point cloud, while the extended dimension of the features from the partitioned point cloud are related to the height values, which are $z(p_k)$ and $z_(q_k) + z_s^g - z_t^g$ for the source and target point clouds, respectively. These height values differ significantly from the fixed value. This dimension extension enhances the feature distance between the overlapping point cloud and the others during correspondence searching, thereby effectively reducing false matches.

V. System Implementation and Evaluation

We implemented a prototype of ERS and conducted an evaluation using collected LiDAR point cloud data. The evaluations primarily focus on assessing the impact of ERS on the accuracy and running time of point cloud registration. Our results indicate that ERS effectively accelerates point cloud registration methods between CVs and improves the accuracy significantly.

A. Implementation and Experiment Setup

Implementation. ERS consists of over 4K lines of code written in C++. We deployed ERS on the laptops with AMD R7 4800H CPU, GeForce RTX 2060 GPU, and 16GB of DDR4 3200Mhz RAM as resource-limited vehicles. To assess the data exchange between CVs, we employed LTE-V2X for vehicle-to-vehicle communication, utilizing QPSK as the

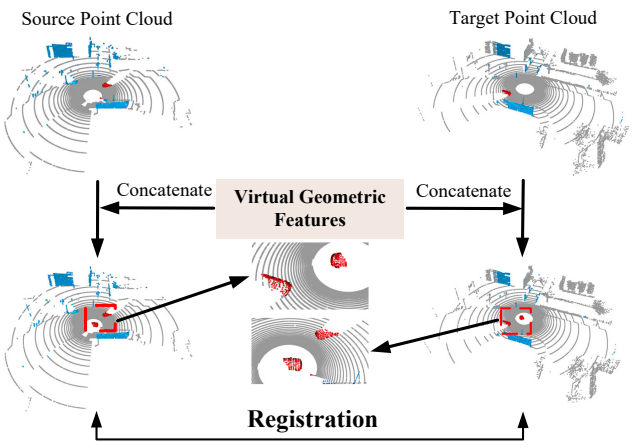


Fig. 9: The partitioned point cloud (blue), the virtual geometric point cloud (red) and the counterpart point cloud (red) are integrated for registration. Redundant point clouds are shown in grey.

modulation scheme [13]. The measured data rate averages around 7.0 Mbps. Ground segmentation on the vehicle side is executed using Patchworkpp [14]. Point cloud processing and the implementation of the RANSAC algorithm are carried out using Open3D [15].

Data collection. The reasons for collecting a new evaluation dataset are as follows. First, existing autonomous driving datasets that are widely used to train point cloud registration are only ego-vehicle datasets and can not truly represent multi-vehicle registration scenes [16], [17]. Moreover, existing datasets for cooperative driving [18], [19] lack the necessary prior information required for virtual geometric feature fusion. As a solution, we used the CARLA simulator [20] to gather LiDAR point clouds from multiple vehicles in the same scene and to collect virtual geometry point clouds of different vehicles. This evaluation dataset contains more than 700 different scenes with at least 3 moving vehicles in each scene. The distances between vehicles are greater than 10 m.

Deployed on different point cloud registration methods. ERS is designed and implemented as a framework for accelerating different point cloud registration methods. Our objective is to provide support for diverse point cloud registration methods, encompassing both feature extraction and end-to-end methods. Moreover, the computational overheads incurred during inference cannot be too large to ensure deployability on resource-constrained devices. Consequently, the specific methods we deployed ERS on include FPFH [21], FCGF [10], Predator [8], and GeoTransformer [9]. FPFH, FCGF, and Predator use the RANSAC-50K algorithm to solve the transformation matrices, while GeoTransformer uses the proposed Local-to-Global registration.

B. Evaluation Metrices

In this paper, our evaluation encompasses the assessment of running time and accuracy. The running time includes the execution time of each component and transmission latency, while accuracy is measured by various metrics, including Feature Matching Recall (FMR), Relative Translation Error (RTE), Relative Rotation Error (RRE), and Registration Recall (RR) [22].

FMR quantifies the percentage of correctly corresponding pairs of points that can confidently recover the transformation. It can be expressed as follows:

$$FMR = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}\Big(\big[\frac{1}{|\Omega_k|} \sum_{(i,j)\in\Omega_k} \mathbb{1}(||T^* p_i - q_j|| < \psi_1) \big] > \psi_2 \Big), \tag{8}$$

where K is the number of point cloud pairs, Ω_k is the set of correspondences between a point cloud pair k, p and q are 3D coordinates from the first and second point cloud, and T^* is the ground-truth transformation matrix. ψ_1 is the inlier distance threshold, and ψ_2 is the inlier ratio threshold.

RTE is the Euclidean distance between estimated and ground-truth translation vectors, denoted by t and t^*, respectively. RRE evaluates the rotation error between the estimated rotation matrix R and the ground-truth rotation matrix R^*. Mathematically, they are defined as:

$$\begin{aligned} RTE &= ||t - t^*||, \\ RRE &= arccos((Tr(R^{\mathrm{T}} R^*) - 1)/2). \end{aligned} \tag{9}$$

RR is the percentage of point cloud pairs whose translation and rotation errors are smaller than a certain threshold, e.g., $RTE <= 2m$ and $RRE <= 5^\circ$.

C. Impact of density threshold

First, we assess the impact of different density thresholds on the execution time and accuracy of the registration, as illustrated in Fig. 10. Six density thresholds (1.5, 2.0, 2.5, 3.0, 4.0, and 5.0) are considered in the evaluation. The execution time for each method increases as the density threshold increases, eventually approaching the execution time without ERS. Generally, RR reach their highest values at lower density thresholds and then gradually decrease. Each registration method extracts different features from the points and thus has different density requirements. For example, FPFH achieves the highest RR at the lowest density threshold, whereas GeoTransformer achieves the highest RR at the density threshold of 3.0. Taking into account the execution time and accuracy, we select appropriate density thresholds for these registration methods, which are 1.5, 2.0, 1.5, and 2.0, respectively.

D. Performance of Accuracy

We evaluate the accuracy improvement that ERS also brings to four different registration methods. We verify the efficiency of the initial positioning.

Fig. 11 illustrates the sensitivity of Feature Matching Recall (FMR) to inlier distance and inlier ratio for the four methods before and after deploying ERS. As anticipated, the deployment of ERS effectively enhances the quality of corresponding

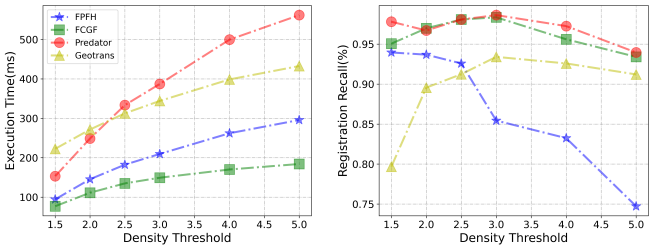


Fig. 10: Impact of different density thresholds on registration execution time and accuracy.

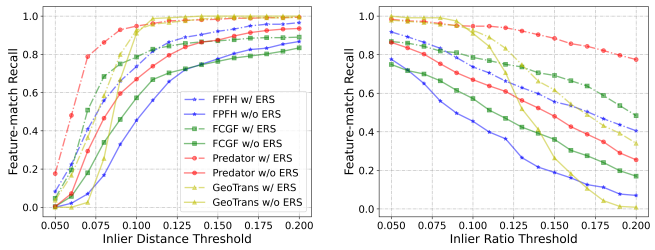


Fig. 11: FMR with respect to inlier distance ψ_1 (left) and inlier ratio ψ_2 (right).

TABLE I: The accuracy of the four registration methods is evaluated in three cases: without ERS deployment, with ERS deployment but no virtual geometric features (VGF), and with full ERS deployment.

Method	RTE(m)	STD(m)	REE(°)	STD(°)	RR(%)
FPFH w/o ERS	0.360	0.267	1.460	1.010	19.5
FPFH w/o VGF	0.335	0.297	1.325	0.950	86.5
FPFH w/ ERS	0.212	0.167	1.091	0.826	94.0
FCGF w/o ERS	0.327	0.408	0.812	0.717	58.2
FCGF w/o VGF	0.347	0.322	1.302	1.011	92.7
FCGF w/ ERS	0.213	0.227	0.615	0.602	97.0
Predator w/o ERS	0.198	0.166	0.779	0.640	63.2
Predator w/o VGF	0.350	0.279	1.567	1.102	77.5
Predator w/ ERS	0.141	0.146	0.581	0.471	97.8
GeoTrans w/o ERS	0.194	0.246	0.537	0.829	64.6
GeoTrans w/o VGF	0.292	0.312	1.242	1.112	85.9
GeoTrans w/ ERS	0.248	0.301	1.005	0.991	89.6

point pairs. For instance, with the inlier ratio and inlier distance fixed at 0.1 and 0.1m, respectively, the FMRs of FCGF, FPFH, and Predator see significant improvements after deploying ERS, reaching 73.7% (+28.2%), 78.6% (+21.5%), and 94.8% (+27.8%). GeoTransformer shows a slight increase (+1.6%) in FMR at this point but a significant improvement in the overall quality of the corresponding point pairs.

We then compare the registration results of the methods with and without ERS, as shown in Table I. All methods achieve higher RR after deploying ERS, achieving 94.5% (+75.0%), 97.0% (+38.8%), 97.8% (+34.6%), 89.6% (+25.0%). Besides GeoTransformer, all other methods achieve higher accuracy in terms of RTE and RRE. Here RTE and RRE are the average values when registration is successful. Deploying ERS solves a large number of complex scenarios that are difficult to successfully registration with the raw methods. In addition,

TABLE II: The initial positioning accuracy.

	RTE(m)	STD(m)	RRE(°)	STD(°)	Recall(%)
Criterion1	0.972	0.470	1.363	1.072	82.55
Criterion2	1.173	1.072	1.385	1.162	94.78

TABLE III: The mean NCC values for template matching.

Layer	Maximum	Sub-maximum
Layer1	0.723	0.510
Layer2	0.526	0.415

virtual geometric feature fusion demonstrates excellent performance, achieving higher RR. In the cases of FCGF and Predator registration, RTE and RRE without virtual geometry features are higher than with raw registration methods. After the integration of virtual geometric features, the accuracy is further enhanced, resulting in a reduction of RTE by 0.134 m and 0.211 m, respectively. Fig. 12 visualizes the registration results of the four methods with and without deploying ERS.

For the positions obtained from the initial positioning, we similarly compare them with the ground-truth values, calculating both RTE and RRE, as presented in Table II. Here, we only calculate the RTE and RRE about the translation and rotation in the plane. We set two different evaluation criteria to calculate Recall, where $Criterion1$ is $RTE <= 2m$ and $RRE <= 5°$, $Criterion2$ is $RTE <= 4m$ and $RRE <= 10°$. Hence, the initial positioning proves effective in discerning the relative positions of the vehicles using distribution matrices, with nearly 95% of the results exhibiting a positional offset of less than 4 meters. Furthermore, we quantify the NCC values resulting from the two-layer template matching, specifically emphasizing the comparison between the maximum and sub-maximum value in each layer, as depicted in Table III. The findings reveal a significant disparity between these two values, with the mean maximum NCC being 0.625, which is 35.0% greater than the mean sub-maximum NCC.

E. Performance of Running Time

We proceed to assess the efficiency of ERS concerning the running time of point cloud registration methods. This evaluation entails a comparison of the running time for each step with and without ERS. Notably, ERS operates independently of the specific registration methods and executes before registration. We separately record the execution time within ERS. The average time of the main steps is shown in Table IV. The average overall execution time of ERS is 31.15 ms. By compressing sparse distribution matrices, target vehicle needs to send data with an average size of 16.23 KB. Additionally, selected point clouds with an average size of 29.68 KB or 49.65 KB (density threshold as 1.5 / 2.0) are transmitted when they receive echoes. This results in a remarkable reduction in transmitted data compared to the raw LiDAR point cloud (2.0 MB), and a reduction of compared to the voxelized point cloud (1.2 MB). The actual measured average transmission latencies are 6.67 ms or 9.23 ms, while the latencies for transmitting

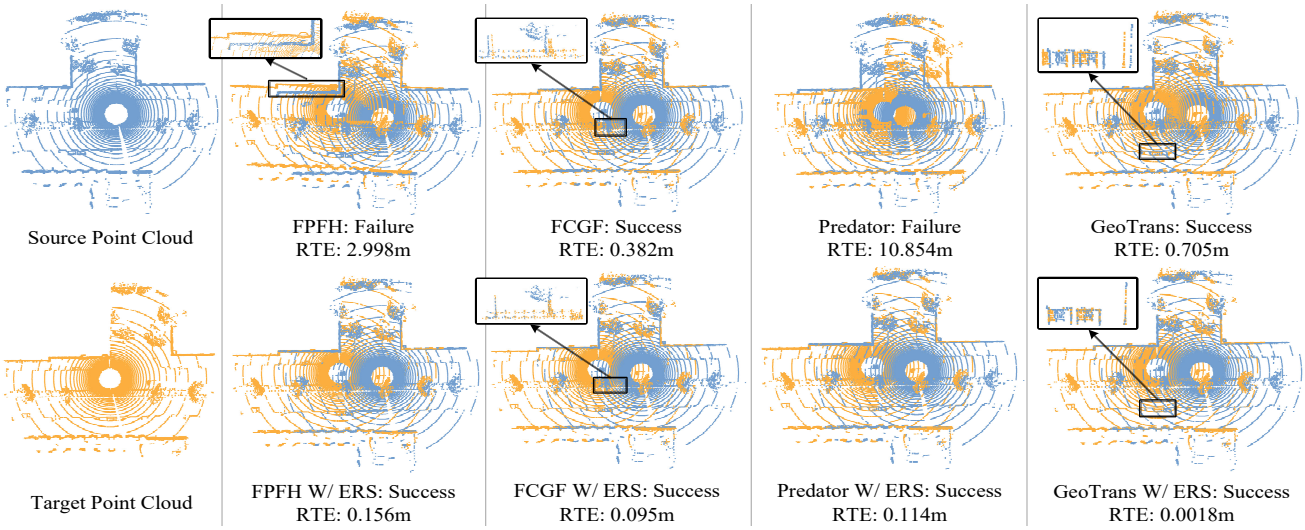


Fig. 12: Visualization of registration results W/ and W/O ERS.

TABLE IV: Average execution time for each step of ERS.

Steps	Times(ms)
Ground Segmentation	12.33
Generate Possible Locations	4.20
Template Matching	12.99
Density-Consistent Partition	1.63

the raw and the voxelized point cloud are 292.71 ms and 171.43 ms, respectively. ERS delivers a more than 18.5× data transmission saving.

While deploying ERS adds time to the execution process, it significantly reduces the overall execution time during subsequent registration, as indicated in Table V. FPFH and FCGF require feature extraction and transformation estimation, while Predator and Geotransformer necessitate down-sampling on point cloud to extract superpoints. After deploying ERS, the four registration methods are accelerated by 6.4×, 7.0×, 9.0× and 2.7×, respectively. In the cases of FCGF and Predator, the virtual geometric features fution effectively increases the correspondences so that the RANSAC algorithm in transformation estimation reaches the termination condition earlier, lending to reductions about 55 ms and 47 ms. For FPFH and GeoTransformer, the virtual geometry feature fusion result in a slight time increase of 16.7 ms and 4.9 ms, respectively. Assessing the overall running time requires consideration of both execution time and transmission latency. Compared to raw registration methods, ERS speeds up the overall running time by 5.7× on average, up to 8.0×. All methods achieve real-time performance (< 200 ms) [23] except GeoTransformer, which uses a special transformation estimation approach resulting in a slight improvement in this part.

VI. Related Work

Point Cloud Registration. Many works focus on how to effectively solve the point cloud registration problem, mainly

TABLE V: The execution times of the four registration methods are evaluated in three cases: without ERS deployment, with ERS deployment but no virtual geometric features (VGF), and with full ERS deployment.

Method	Down-Sampling(ms)	Extract Features(ms)	Transformation Estimation(ms)	Total Execution Time(ms)
FPFH w/o ERS	-	168.70	436.37	605.07
FPFH w/o VGF	-	23.38	55.30	77.68
FPFH w/ ERS	-	31.89	62.52	94.41
FCGF w/o ERS	-	64.51	669.68	734.19
FCGF w/o VGF	-	31.13	129.18	160.31
FCGF w/ ERS	-	31.33	74.20	105.53
Predator w/o ERS	329.47	298.08	740.91	1368.46
Predator w/o VGF	40.16	36.43	112.91	189.50
Predator w/ ERS	46.22	39.88	66.23	152.33
GeoTrans w/o ERS	354.23	221.12	143.81	719.16
GeoTrans w/o VGF	72.73	45.04	142.83	260.60
GeoTrans w/ ERS	74.69	47.13	143.67	265.49

investigating extracting features from the point cloud. Rusu et al. propose a 3D point cloud local feature descriptor for object detection and point cloud registration [21]. Choy et al. present a fully convolutional geometric feature network that produces high-quality features quickly [10]. SpinNet implements a neural feature extractor using 3D cylindrical convolutional neural layers [24]. Huang et al. use a graph neural network and cross-attention module to estimate the overlap region between two point clouds and thus extract features [8]. In addition, due to the widespread application of self-attention and cross-attention mechanisms, several works have employed transformer architecture without traditional optimization methods to achieve end-to-end registration [9], [25]. Unlike existing works, ERS utilizes the characteristics of LiDAR point clouds to filter them lightly, retaining only efficient point pairs to reduce running time.

Connected Vehicles. Connected vehicles contribute to an extended perception range, consequently enhancing reliability and safety [26], [27]. OPV2V aggregates information from multiple vehicles through a designed attentive intermediate

fusion pipeline [28]. EMP facilitates the sharing of raw point clouds via edge servers, employing a segmentation on various vehicles [29]. Autocast identifies common objects and transmits pertinent data based on the positional relationship among traffic participants and the temporal evolution of their trajectories [30]. He et al. determines the vehicle's location through the alignment of perceptual point clouds between the roadside infrastructure and the vehicle [23], [31].

VII. Conclusion

In this paper, we present ERS, a plug-and-play system to enhance the speed and accuracy of LiDAR point cloud registration between CVs. The system comprises three key components: initial positioning, density-consistent partition strategy, and virtual geometric feature fusion. The initial positioning leverages distribution matrices for lightweight vehicle positioning, the density-consistent partition strategy effectively filters points that may be redundant to registration, and the fusion of virtual geometric features further refines the accuracy of the registration. Results demonstrate that ERS improves the overall running time of current state-of-the-art baselines by 5.7× with 42.1% registration recall gains.

References

[1] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*, 2021.

[2] Quan Liu, Yunsong Zhou, Hongzi Zhu, Shan Chang, and Minyi Guo. Apr: Online distant point cloud registration through aggregated point cloud reconstruction. *arXiv preprint arXiv:2305.02893*, 2023.

[3] Baorui Ma, Yu-Shen Liu, Matthias Zwicker, and Zhizhong Han. Surface reconstruction from point clouds by learning predictive context priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6326–6337, 2022.

[4] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[5] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019.

[7] Yingjie Wang, Qiuyu Mao, Hanqi Zhu, Jiajun Deng, Yu Zhang, Jianmin Ji, Houqiang Li, and Yanyong Zhang. Multi-modal 3d object detection in autonomous driving: a survey. *International Journal of Computer Vision*, pages 1–31, 2023.

[8] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 4267–4276, 2021.

[9] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11143–11152, 2022.

[10] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8958–8966, 2019.

[11] Paul-Edouard Sarlin, Daniel DeTone, Tsun-Yi Yang, Armen Avetisyan, Julian Straub, Tomasz Malisiewicz, Samuel Rota Bulò, Richard Newcombe, Peter Kontschieder, and Vasileios Balntas. Orienternet: Visual localization in 2d public maps with neural matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21632–21642, 2023.

[12] Hyungtae Lim, Minho Oh, and Hyun Myung. Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor. *IEEE Robotics and Automation Letters*, 6(4):6458–6465, 2021.

[13] Ehsan Moradi-Pari, Danyang Tian, Mojtaba Bahramgiri, Samer Rajab, and Sue Bai. Dsrc versus lte-v2x: Empirical performance analysis of direct vehicular communication technologies. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):4889–4903, 2023.

[14] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13276–13283. IEEE, 2022.

[15] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

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

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

[18] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589, 2022.

[19] Tsinghua University Institue for AI Industry Research (AIR). Vehicle-infrastructure collaborative autonomous driving: Dair-v2x dataset, 2021.

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

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

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

[23] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. 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*, pages 573–586, 2021.

[24] Sheng Ao, Qingyong Hu, Bo Yang, Andrew Markham, and Yulan Guo. Spinnet: Learning a general surface descriptor for 3d point cloud registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11753–11762, 2021.

[25] Jiuming Liu, Guangming Wang, Zhe Liu, Chaokang Jiang, Marc Pollefeys, and Hesheng Wang. Regformer: An efficient projection-aware transformer network for large-scale point cloud registration. *arXiv preprint arXiv:2303.12384*, 2023.

[26] Julian Heinovski and Falko Dressler. Where to decide? centralized vs. distributed vehicle assignment for platoon formation. *arXiv preprint arXiv:2310.09580*, 2023.

[27] Qichang Liu, Haiying Shen, Tanmoy Sen, and Rui Ning. Time-series misalignment aware dnn adversarial attacks for connected autonomous vehicles. In *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 261–269, 2023.

[28] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589. IEEE, 2022.

[29] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. Emp: Edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558, 2021.

[30] Hang Qiu, Pohan Huang, Namo Asavisanu, Xiaochen Liu, Konstantinos Psounis, and Ramesh Govindan. Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. *arXiv preprint arXiv:2112.14947*, 2021.

[31] Yuze He, Chen Bian, Jingfei Xia, Shuyao Shi, Zhenyu Yan, Qun Song, and Guoliang Xing. Vi-map: Infrastructure-assisted real-time hd mapping for autonomous driving. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.