44th IEEE International Conference on
Distributed Computing Systems

ERS: Faster LiDAR Point Cloud Registration for Connected Vehicles

Yu Zhao[1], Jinrui Zhou[1], Mingjun Xiao[1], Jie Wu[2], and He Sun[1]

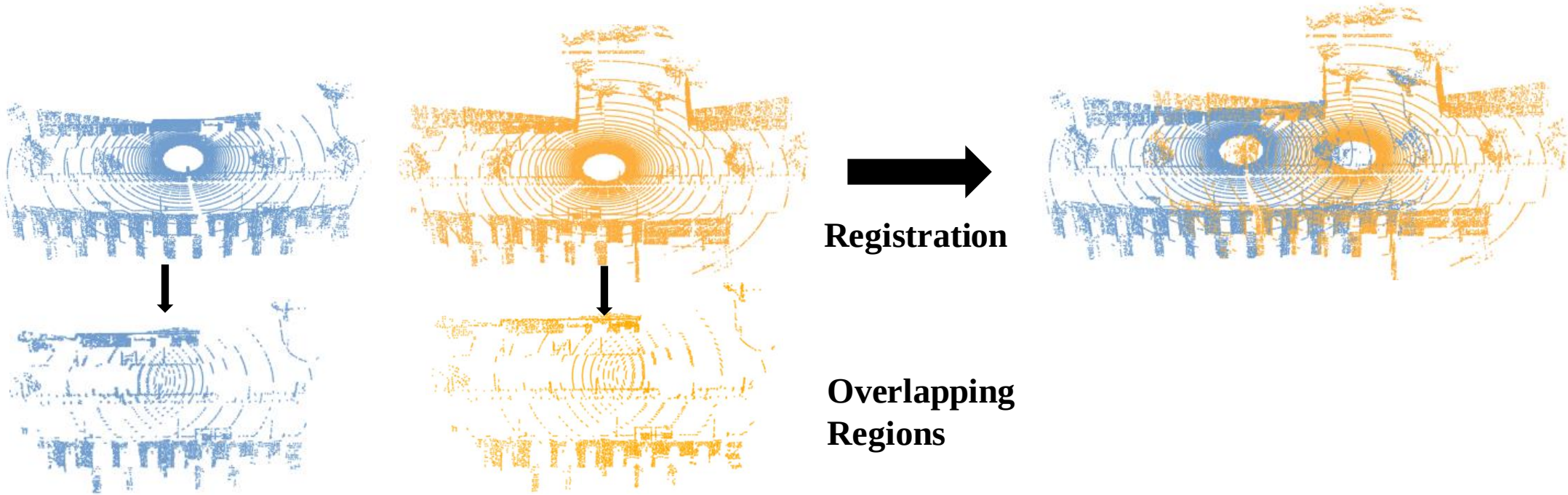
[1]University of Science and Technology of China
[2]Temple University

Outline

- Background & Motivation
- System Architecture
- Methodology
- Evaluation & Conclusion

Background

- Point Cloud Registration for Connected Vehicles
  - Extend the connected vehicles' views
  - Circumvent the resource-intensive procedures
  - Exhibit adaptability to planning without HD maps
- Registration Challenges
  - Small overlapping regions
  - Computationally intensive task
  - Significant transmission latency due to large data volumes



Background

- **Formulation of the Point Cloud Registration Problem:**

$$\min_{R \in SO(3), t \in R^3} \| \tilde{P} - (R\tilde{Q} + t) \|_2$$

$\tilde{\mathcal{P}}$ & $\tilde{Q}$ are matrices consisting of corresponding points selected from the point clouds $\mathcal{P}$ & Q .
$\mathcal{P} \in \mathbb{R}^{M \times 3}$ and $Q \in \mathbb{R}^{M \times 3}$ represent the source and target point clouds.

R & t : $R \in SO(3)$ denotes the rotation matrix and $t \in R^3$ denotes translation vector.

Transformation Matrix: $\begin{bmatrix} R & t^T \\ 0_{1\times 3} & 1 \end{bmatrix}$

- **Correspondence Search**:

$$\| \mathcal{F}(p_i) - \mathcal{F}(q_j) \| < f_{thr}$$

$\mathcal{F}$ denotes the features extracted from points.
p_i and q_j are a pair of corresponding points.
f_{thr} is a threshold for feature distances.

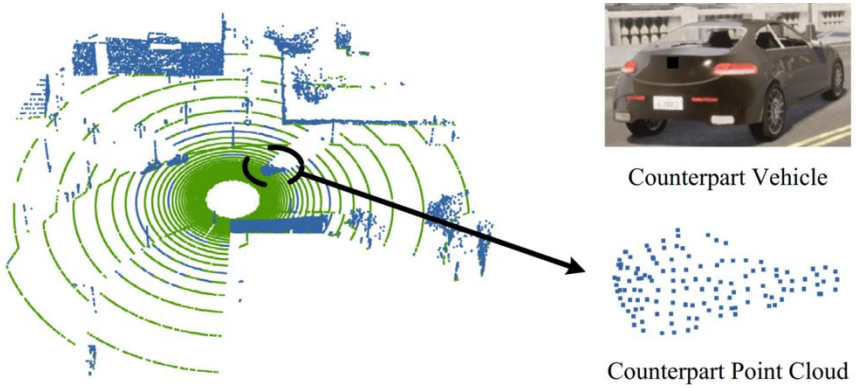
- **Transformation Estimation**:

The transformation estimation can be solved using particular algorithms, with the Random Sample Consensus (RANSAC) algorithm being most commonly used.

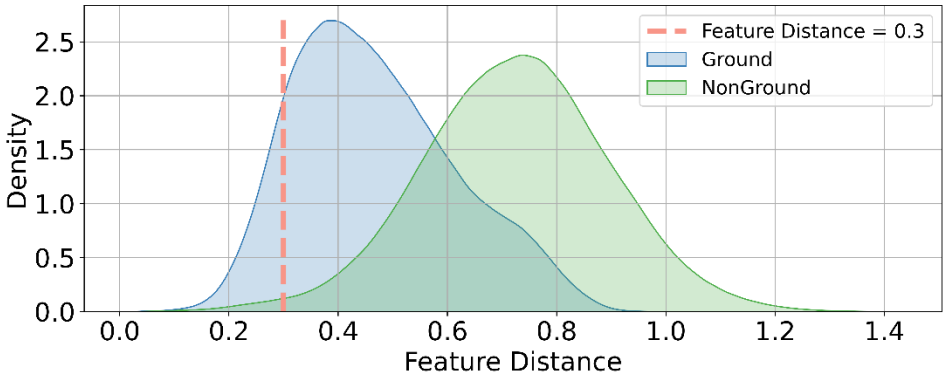
Motivation

Impact of different types of point clouds

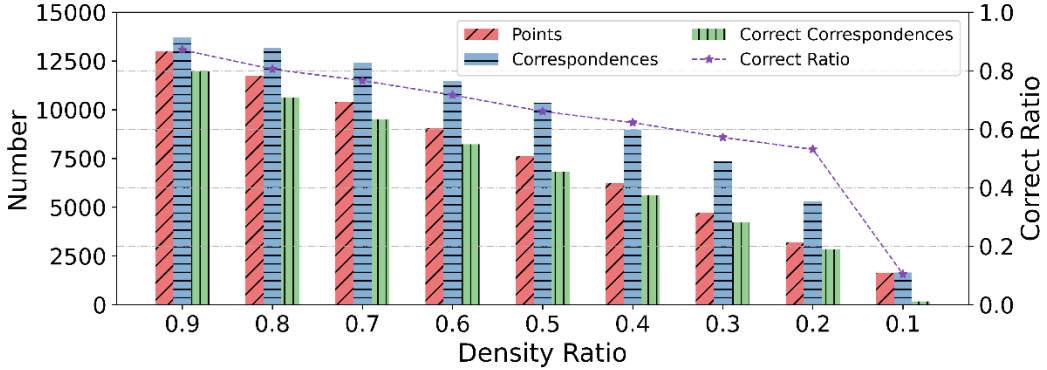
- Counterpart point cloud → can not find correspondence

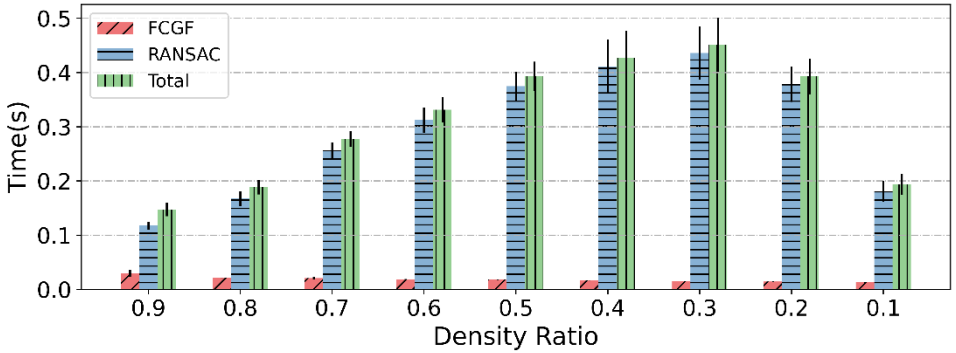


- Ground points → majority of the points & concentric circles → behave similarly



- Non-ground points → different densities → different extracted features → negative influence on point cloud registration

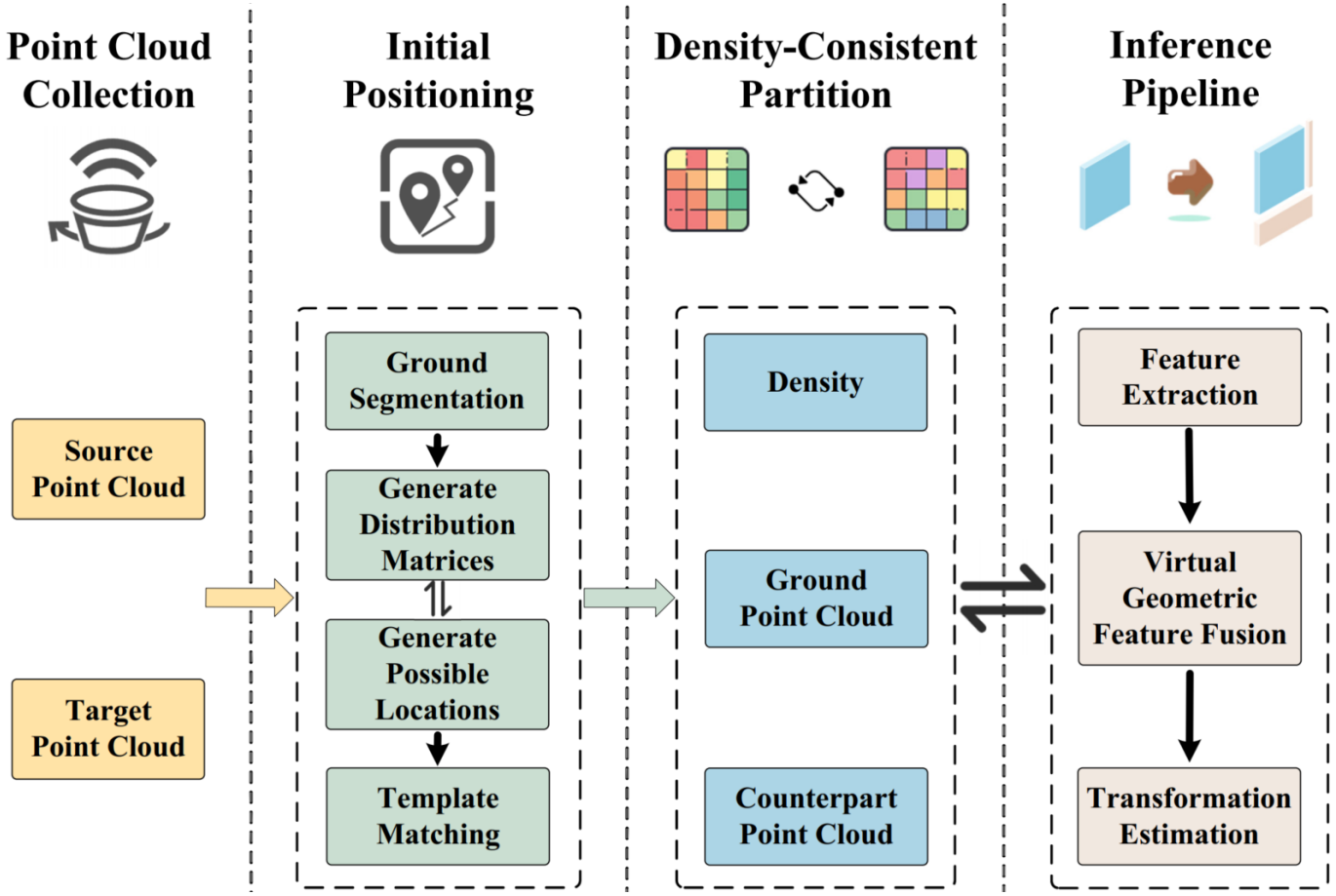




Outline

- Background & Motivation
- System Architecture
- Methodology
- Evaluation & Conclusion

System Architecture



Workflow

- **Initial positioning based on point cloud distributions:** convert point clouds into distribution matrices, and perform template matching to determine the initial transformation.
- **Density- consistent partition strategy:** identify correspondences between regions in source and target point cloud, simplify the ground point cloud while preserving observations about the counterpart vehicle.
- **Inference Pipeline:** integrate virtual geometric features with those features extracted from the partitioned point cloud.

Outline

- Background & Motivation
- Related Work & Problem Formulation
- Methodology
- Evaluation & Conclusion

Initial Positioning

➢ Four Main Steps

Step 1

Execute Ground Segmentation to obtain ground point cloud $\mathcal{G}$, non-ground low point cloud $\hat{\mathcal{G}}^l$, and non-ground high point cloud $\hat{\mathcal{G}}^h$.

Step 2

Divide these point clouds to Generate Distribution Matrices, and denote matrices as M, $\hat{M}^l$ and $\hat{M}^h$ from point clouds $\hat{\mathcal{G}}$, $\hat{\mathcal{G}}^l$, and $\hat{\mathcal{G}}^h$, respectively.

Step 3

Generate Possible Locations by identifying gaps in the ground distribution matrix, and further filter possible locations.

Step 4

Seek pairs of points in two possible locations' sets, and implement Template Matching to obtain initial transformation matrix.

➢ Ground Segmentation

Employ Patchworkpp to obtain ground point cloud $\mathcal{G}$ and non-ground point cloud $\hat{\mathcal{G}}$

Partition $\hat{\mathcal{G}}$ into non-ground low point cloud $\hat{\mathcal{G}}^l$ and non-ground high point cloud $\hat{\mathcal{G}}^h$

$$\hat{\mathcal{G}}^l = \{p_k \in \mathcal{P} \,|z(p_k) \leq z_{init} + z_{thr}\}$$
$$\hat{\mathcal{G}}^h = \{p_k \in \mathcal{P} \,|z(p_k) \geq z_{init} + z_{thr}\}$$

$z(\cdot)$ returns the z value of a point
z_{init} denotes the height of the LiDAR
z_{thr} denotes a defined threshold

Initial Positioning

➢ Generate Distribution Matrices

Core: Divide the point cloud into meshes based on the (x, y) values of the points.

Denote three Matrices as as M, $\widehat{M}^l$ and $\widehat{M}^h$.

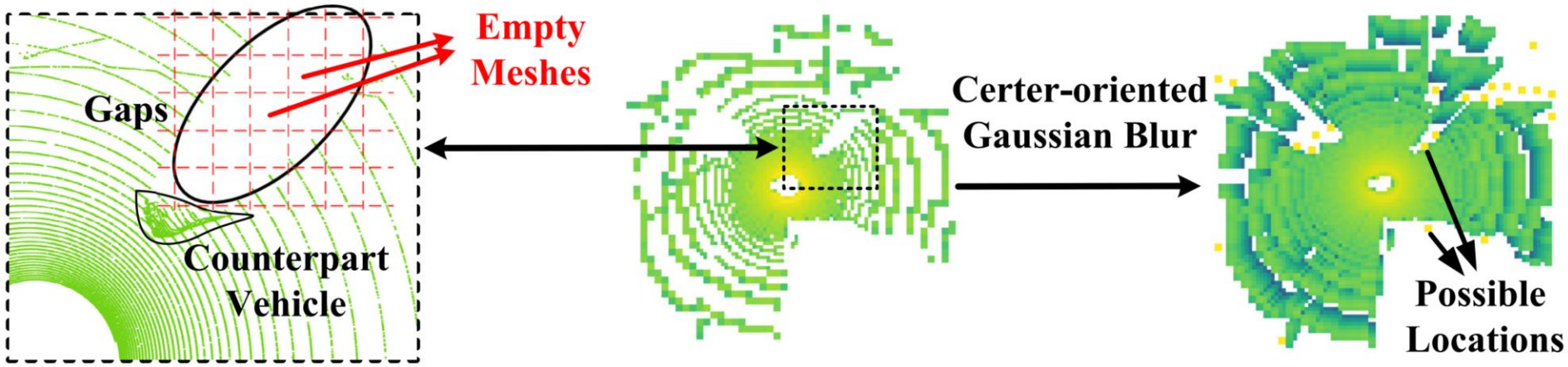
Employ larger area for long-distance meshes.

➢ Generate Possible Locations

Core: vehicles in the perceptual range create gaps in the ground → Identify gaps in ground distribution matrix M

Devise a center-oriented Gaussian blur to fill in ground gaps at long distances

Find obscured objects in front of gaps → Filter possible locations → $\mathcal{C} = \{(x^1, y^1), \ldots, (x^i, y^i), \ldots, (x^n, y^n)\}$



Initial Positioning

➢ **Template Matching**

$d_s^i = \| (x_s^i, y_s^i) \|_2$, the distance between possible location and center
$|\mathcal{C}_s|$, $|\mathcal{C}_t|$ denote the possible locations' sets from two vehicles

$$\mathcal{D}_s = \{d_s^1, \dots, d_s^i, \dots, d_s^{|\mathcal{C}_s|}\}$$

$$\mathcal{D}_t = \{d_t^1, \dots, d_t^j, \dots, d_t^{|\mathcal{C}_t|}\}$$

If $|d_s^i - d_t^j| < d_{thr}$, calculate the rotation angle and translation vector:

$$\delta_{ij} = arctan(-y_s^i, -x_s^i) - arctan(y_t^j, x_t^j)$$

$$t_{ij} = ((x_s^i - x_t^j)/2, (y_s^i - y_t^j)/2)$$

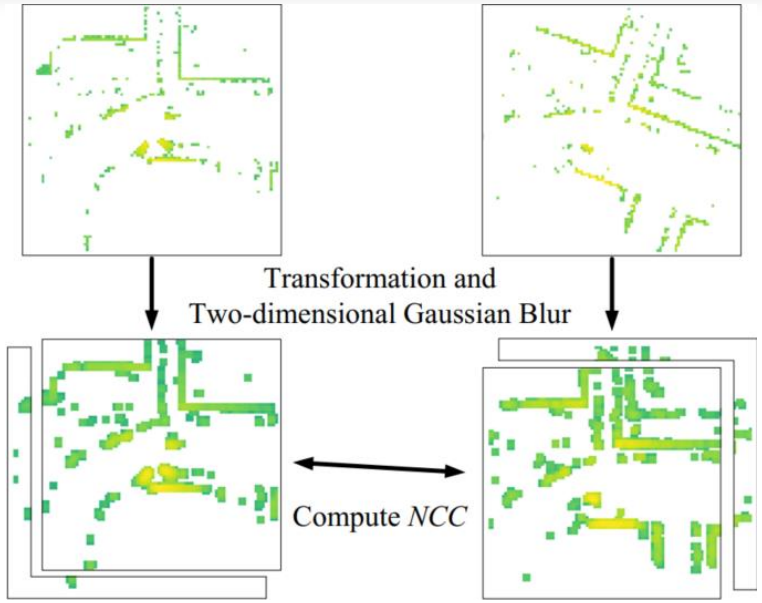
Rotation angle enables the pair of possible locations to be approximately symmetric about the center
Translation vector makes the possible locations are close to the center in corresponding matrices

Execute transformation and two-dimensional Gaussian Blur:

$\widehat{M}^h, \widehat{M}^l \rightarrow \widehat{M}^{h'}, \widehat{M}^{l'}$

Core: Execute Binarization and calculate the Normalized Cross Correlation (NCC) for Template Matching:

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



Initial Positioning

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;

Core: Find possible locations and execute two-layer template matching → initial transformation

- Find the corresponding pairs of points based on the distance to the center.
- Execute the first layer template matching on non-ground high point cloud.
- Select corresponding pairs of points via top-k selection.
- Extend the values of δ_k and t_k and execute the second layer template matching on non-ground low point cloud.
- Return the δ and t with highest *NCC*

Density-consistent Partition

➢ **Three factors for partition**

Factor 1: Density

Factor 2: Counterpart point cloud

Factor 3: Ground point cloud

➢ **Density**

- ✓ Merge the matrices $\widehat{M}^l$ and $\widehat{M}^h$ into a non-ground distributed matrix denoted as $\widehat{M}$.
- ✓ Adopt a larger mesh length to reduce the effect of initial transformation error.
- ✓ Select corresponding meshes with close density ratios :

$$1/dt \leq \widehat{M}_t(i,j)/\widehat{M}_s(i,j) \leq dt$$ ---→ dt denotes the density threshold

➢ **Counterpart point cloud**

- ✓ Be filter out due to the mismatch in density → Keep this part and utilize it

➢ **Ground point cloud**

- ✓ Exhibit similar features → calculate and retain the normal and height
- ✓ Normal vector could initiate the rotation in x-axis and y-axis → pitch and roll angles of the vehicle
- ✓ Height could be embedded as features

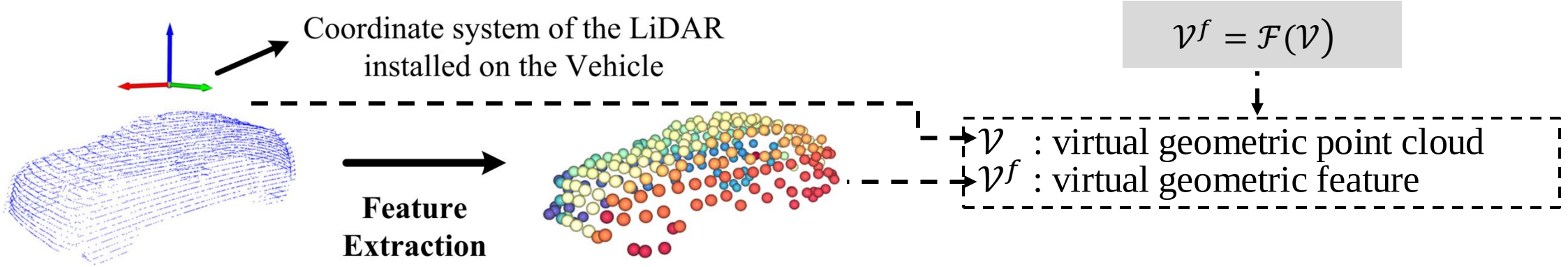
Inference Pipeline

Feature Extraction & Transformation Estimation are consistent with previous work.

➢ Virtual Geometric Feature Fusion

✓ Geometric Feature Construction:

- Gather point clouds using *reference LiDARs* from various positions surrounding a vehicle.
- Record transformation matrices between these *reference LiDARs* and the LiDAR installed on the vehicle.
- Transform these collected point clouds into a coordinate system.



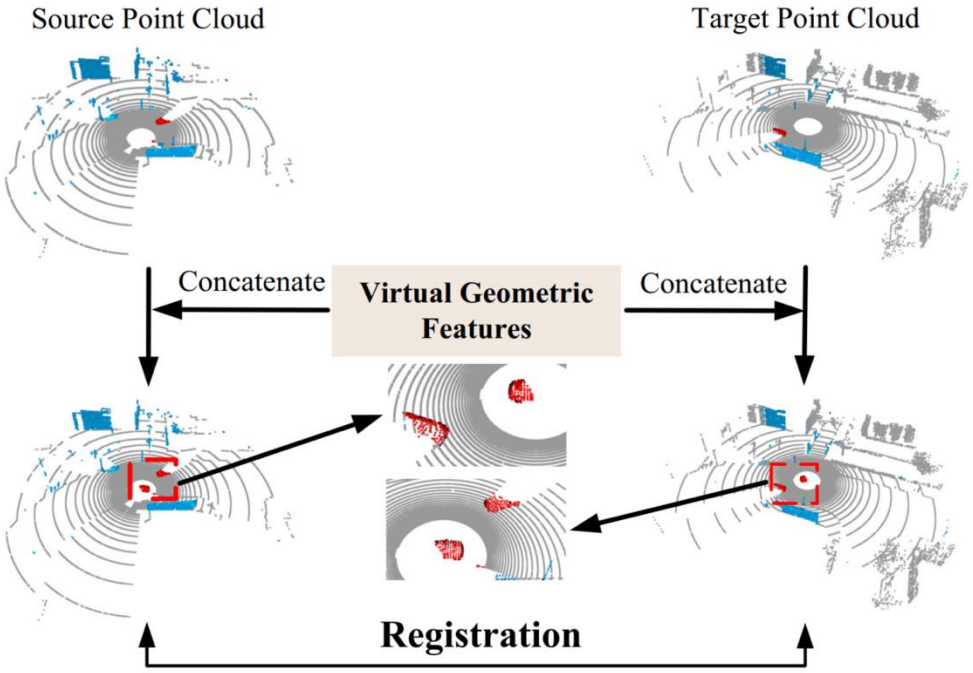
Inference Pipeline

➢ Virtual Geometric Feature Extraction

✓ Take the partitioned point cloud $\dot{\mathcal{P}}$ to generate integrated point cloud $\dot{\mathcal{P}}'$:

$$\dot{\mathcal{P}}' = Concat(\dot{\mathcal{P}}, \mathcal{V}_s)$$
$$\mathcal{F}(\dot{\mathcal{P}}') = Concat(\mathcal{F}(\dot{\mathcal{P}}), \mathcal{V}_s^f).$$

Instead of integrating the entire virtual point cloud, we trim it based on the initial transformation to avoid mismatched of correspondence.



✓ Assign fixed values into the features of counterpart point cloud and the virtual geometric point cloud.

✓ Assign values about ground height into the features of partitioned point cloud.

Outline

- Background & Motivation
- Related Work & Problem Formulation
- Basic Idea & Solution
- Evaluation & Conclusion

Experimental Settings

Dataset

- CARLA Simulator → gather LiDAR point clouds & collect virtual geometry point clouds of different vehicles.

Platform

- Employ ERS on laptops as resource-limited vehicles
- measured data rate around 7.0 Mbps

Compared Algorithms

Feature Extraction + Transformation Estimation:

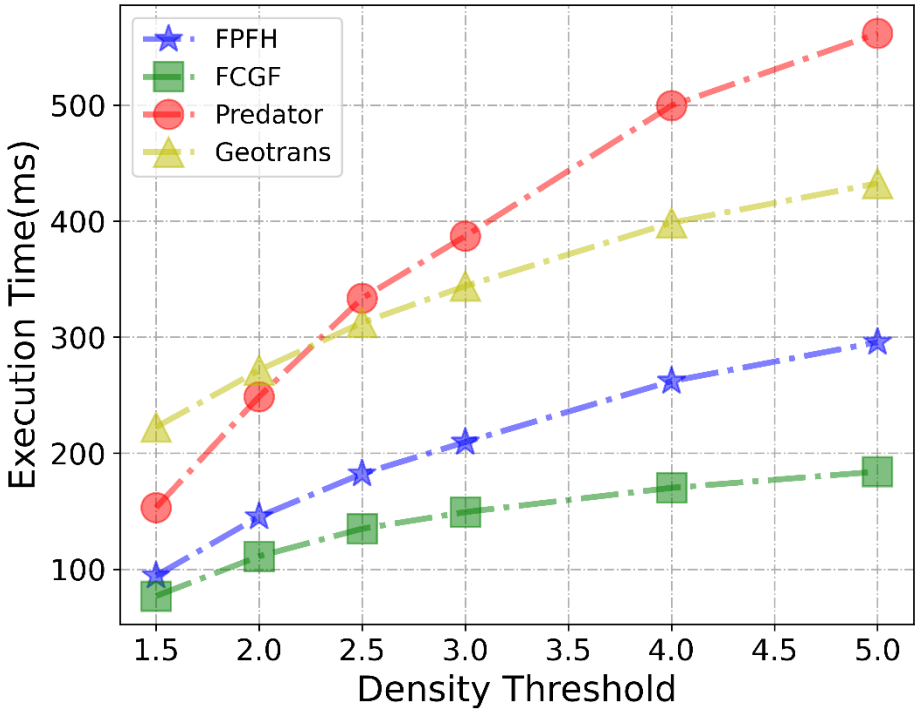
- FPFH + RANSAC
- FCGF + RANSAC
- Predator + RANSAC
- GeoTransformer (End-to-end method)

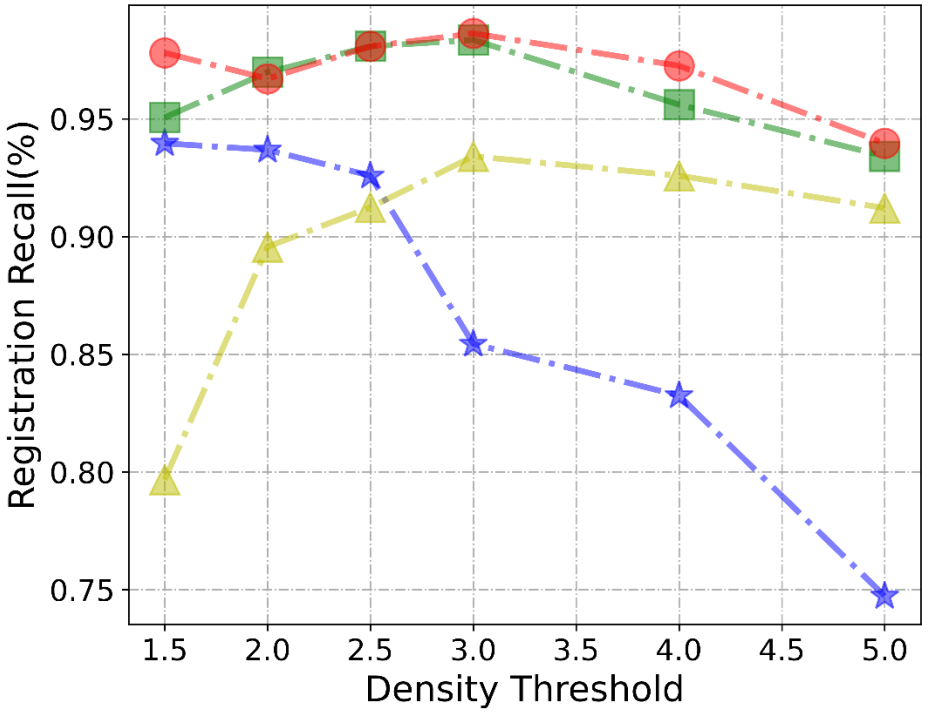
Evaluation Metrics

- Accuracy: Relative Rotation Error (RTE), Relative Translation Error (REE), Feature Matching Recall (FMR), Registration Recall (RR)
- Running Time: Execution time & Transmission latency

Evaluation

- Impact of different density thresholds on execution time and accuracy





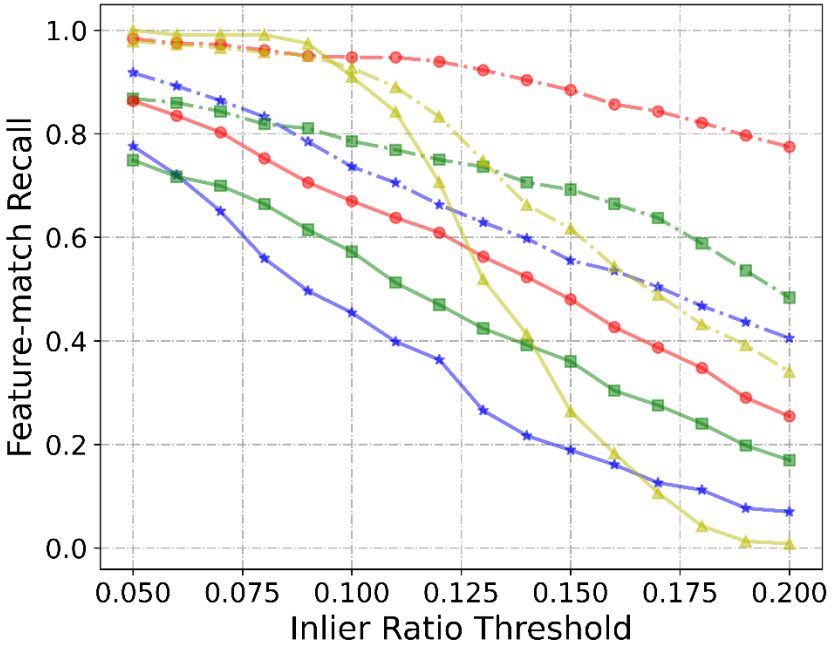
The execution time for each method increases as the density threshold increases.

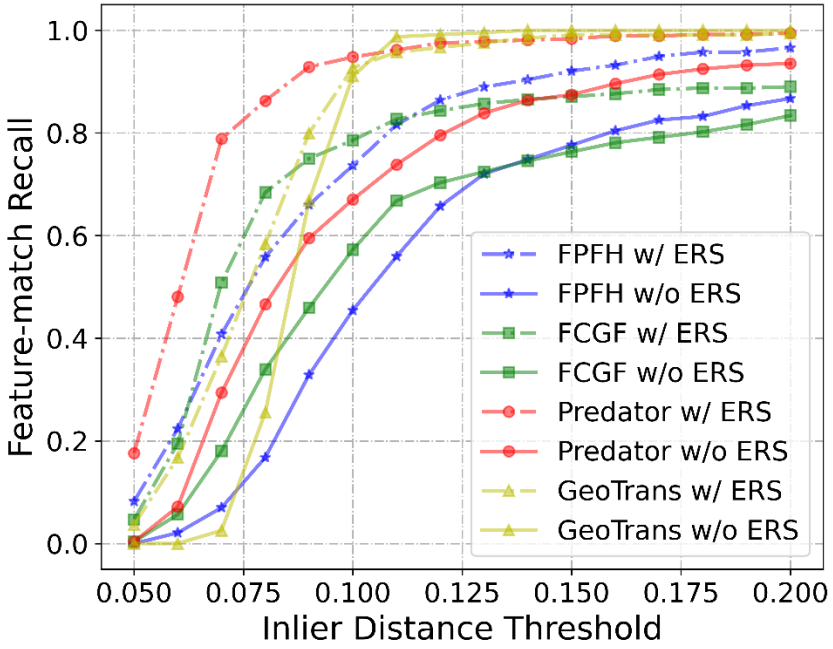
Registration Recall reach their highest values at lower density thresholds and then gradually decrease.

Taking into account the execution time and RR, select appropriate density thresholds for these registration methods.

Evaluation

- FMR with respect to inlier distance (left) and inlier ratio (right)





- ✓ ERS effectively enhances the quality of corresponding point pairs.
- ✓ With the inlier ratio and inlier distance fixed at 0.1 and 0.1 m, three methods see significant improvement after deploying ERS.
- ✓ GeoTransformer shows a slight increase at this point but a clear improvement in the overall quality.

Evaluation

- Accuracy under three cases:
  - without ERS employment,
  - with ERS deployment but no virtual geometric features (VGF),
  - 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

All methods achieve higher RR after deploying ERS.

Three methods achieve higher accuracy in terms of RTE and RRE.

RRE and RTE are the average values when registration is successful, deploying ERS solves a large number of complex scenarios. So the RTE and RRE of GeoTransformer becomes higher.

Evaluation

- 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

ERS operates independently of the specific registration methods.

The average overall execution time of ERS is 31.15ms.

- Average transmission latency

Raw LiDAR point cloud (2.0 MB) \ point cloud after voxelization(1.2 MB) →
Sparse distribution matrices (16.23 KB) +
Selected point clouds (29.68 KB or 49.65 KB with density threshold as 1.5/2.0)

ERS delivers a more than 18.5X data transmission saving.

Evaluation

- Execution times of the four registration methods

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

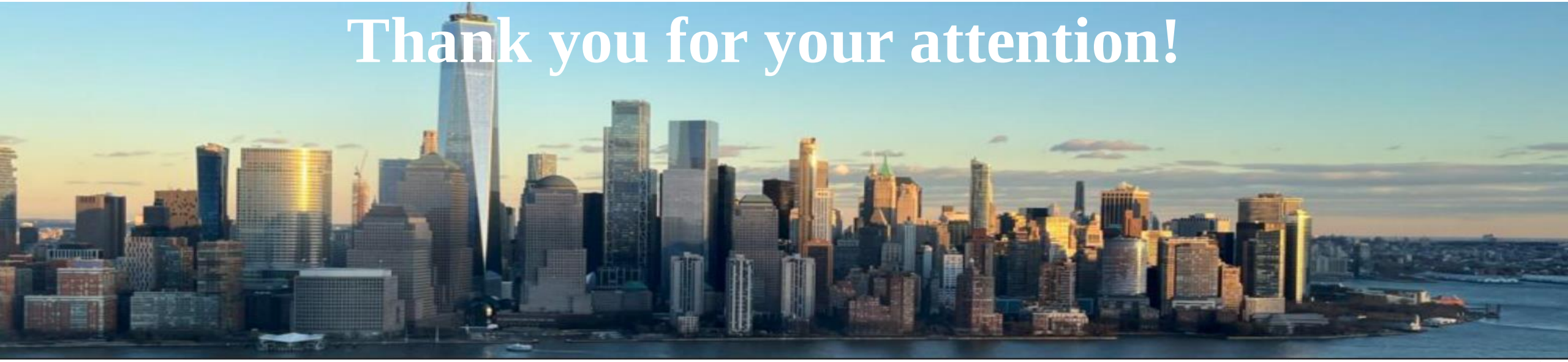
FGCF & Predator: VGF effectively increases the correspondences so that RANSAC reaches the termination condition earlier.
FPFH & GeoTransformer: VGF results in a slight time increase.

ERS speeds up the overall running time by 5.7X on average, up to 8.0X. All methods achieve real-time or near real-time performance.

Conclusion

- ✓ We present ERS, a plug-and-play system to enhance the speed and accuracy of LiDAR point cloud registration between CVs.
- ✓ Extensive simulations validate its great performance. Results demonstrate that ERS improves the overall running time of current state-of-art baselines by 5.7*X* with 42.1% registration recalls gains.

- ✓ To facilitate computing and transmission, we propose three key components to find overlapping regions: initial positioning, density-consistent partition strategy, and virtual geometric feature fusion.



Thank you for your attention!

Question?

Yu Zhao zhaoyu0624@mail.ustc.edu.cn