RegFormer: An Efficient Projection-Aware Transformer Network for Large-Scale Point Cloud Registration (Supplementary Materials)

Jiuming Liu[1], Guangming Wang[1,4], Zhe Liu[2*], Chaokang Jiang[3], Marc Pollefeys[4,5], Hesheng Wang[1*]
[1]Department of Automation, Key Laboratory of System Control
and Information Processing of Ministry of Education, Shanghai Jiao Tong University
[2] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[3] China University of Mining and Technology [4] ETH Zürich [5] Microsoft
{liujiuming,wangguangming,liuzhesjtu,wanghesheng}@sjtu.edu.cn
ts20060079a31@cumt.edu.cn marc.pollefeys@inf.ethz.ch

1. Overview

In the supplementary materials, we first point out the differences between our RegFormer and previous registration works in Section 2. Also, more network details are shown including both the network architecture and parameters in Section 3. Finally, extensive qualitative and quantitative experiment results in Section 4 are given to demonstrate the superiority of our RegFormer. Moreover, we evaluate our RegFormer on the LiDAR odometry task. The state-of-the-art performance on the odometry task proves that our network has strong generalization capability.

2. Comparison with Previous Pipelines

As illustrated in Fig. 1 A), most point cloud registration works follow a two-stage paradigm, which first extracts explicit correspondences by learning discriminative local descriptors [4] or keypoint detection [1]. Then, estimators (eg. SVD) are utilized to regress the transformation through these correspondences. The second stage is commonly figured out as an optimization problem:

$$R, t = \underset{R,t}{argmin} \sum_{i} \|Rx_i + t - y_i)\|_2, \qquad (1)$$

where x_i, y_i are the i-th pair of putative correspondence coordinates. $\|\cdot\|_2$ indicates the L_2 Norm. Different from these previous works, our network inputs projected point cloud patches into the transformer with a global perspective as in Fig. 1 B). Our RegFormer gets rid of the explicit correspondence establishment. Also, the final pose is directly generated from learned powerful geometric association features between two frames. Without any post-processing

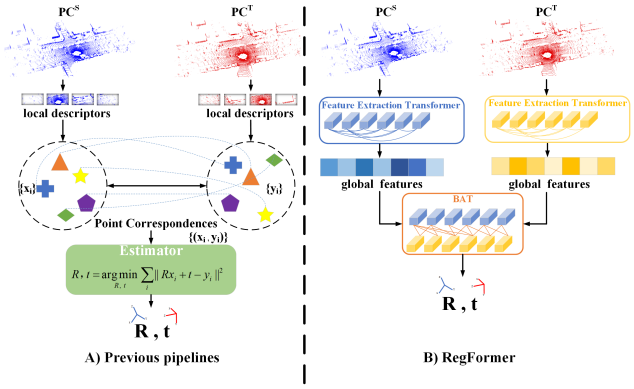

Figure 1: Comparison with previous pipelines. Most registration works follow a two-stage paradigm as the left figure A). They first extract explicit correspondences by learning discriminative local descriptors, and then leverage estimators to regress the transformation. Differently, our network directly inputs projected point cloud patches into the transformer with a global perspective B).

choice, our network can filter outliers effectively since the global understanding of feature maps is helpful to recognize the dynamic motions and localize the occluded objects.

3. Network Architecture Details

In this section, we give a comprehensive description of the architectural details of the whole network.

3.1. Network Configurations

In RegFormer, we first leverage three successive feature extraction transformer stages to extract global features. Then, a bijective association transformer is designed to correlate two point cloud frames and generate initial motion embed-

*Corresponding Authors. The first two authors contributed equally.

Module	Layer	N	Sample rate	Channels	MLP width	Heads
Feature Extraction Transformer	Set conv layer for I^S and I^T in stage 1	2	$[H/4 \times W/8]$	16	[16,64,16]	2
	Set conv layer for PC_1^S and PC_1^T in stage 2	2	$[H/8 \times W/16]$	32	[32,64,32]	4
	Set conv layer for PC_2^S and PC_2^T in stage 3	6	$[H/16 \times W/32]$	64	[64,256,64]	8
Bijective Association Transformer	Cross-attention layer in stage 3	6	—	64	[64,256,64]	8

Table 1: **Detailed network parameters in Transformer.** N denotes the number of transformer blocks. MLP width indicates the expansion layer of MLP in feed-forward networks. The variables in the table are defined the same as in the main manuscript.

dings. Finally, the transformation is recovered from these embeddings and refined iteratively.

Feature extraction transformer: For the feature extraction part of our RegFormer, we establish a three-stage successive downsampling transformer following most vision transformer architectures. Notably, the sample rate of our RegFormer is slightly different from the vision transformer architecture. As to images, the input height is usually equal to the width. However, due to the raw 3D data structure, projected pseudo images are not square in shape. Instead, they have a strip-like shape where widths are much larger than heights since LiDAR sensors can cover 360 degrees in the horizontal direction but a limited field in the vertical direction. Thus, we set the downsampling rate as $[4, 2, 2]$ for the height and $[8, 2, 2]$ for the width of initial feature maps as depicted in Table 1. Also, feature channels become twice after each patch merging layer. We add the attention heads accordingly for better modeling capability. The MLP expansion is set as 4 in each transformer block.

Bijetive association transformer: Apart from the sample rate, other parameters in the cross-attention block of BAT are the same as the ones in stage 3 of the feature extraction transformer. When it comes to the all-to-all point gathering module, each point in one frame correlates with all K points in the other frame as in Table 2, where K is 224 in layer 3. Then, initial motion embeddings and transformations are generated from correlated features.

Interative refinement with PWC structure: In the main manuscript, we have briefly clarified the PWC refinement structure [9], where Pyramid, Warp, and Cost-volume are applied to recover the precise transformation. In Table 2, we show more details about the PWC structure. First, the source point cloud is warped by the transformation from the upper layers. The warped source point cloud is closer to the target one. Then, we calculate the attentive cost volume [12] between warped source point cloud features and original target point cloud features. Finally, features from the cost volume layer together with upsampled features will go through a shared MLP for generating residual motion embeddings.

The PWC (Pyramid, Warping, and Cost volume) refinement is extremely crucial for generating the precise transformation. As in Table 5 of the main manuscript, the all-to-all strategy in the coarsest layer still has 30% errors because of the lowest resolution with sparse points. Then, the source point cloud will be iteratively warped closer to target one by the estimated coarse transformation. Therefore, this warping operation makes the following local point association in finer layers effective enough to estimate large residual transformation iteratively (15.7%, 16.9%, 3.0% improvement respectively in three iterations).

3.2. Cross-Attention Mechanism in BAT

In this section, we show more detailed descriptions of the cross-attention mechanism in our Bijective Association Transformer (BAT). This layer fully correlates points within the same frame and also between different frames as depicted in Algorithm 1.

To be more specific, given unconditioned features of two point cloud frames F_3^S, F_3^T and their masks M_3^S, M_3^T in stage 3 of the feature extraction transformer layer, cyclic shift and window partition are employed on both point clouds and their corresponding masks. Take the source point cloud as an example, F_3^S and M_3^S are first fed into a PW-MSA module. The same goes for the target point cloud, which enables each point to interact with other points in the same frame as:

$$\begin{aligned}\tilde{F}^S &= PW\text{-}MSA(F_3^S) \\ &= Attention(F_3^S \times W^Q, F_3^S \times W^K, F_3^S \times W^V) \\ &= Attention(Q_3^S, K_3^S, V_3^S) \\ &= softmax(\frac{Q_3^S K_3^S}{\sqrt{d_{head}}} + M_3^S + Bias)V_3^S, \end{aligned} \tag{2}$$

$$\begin{aligned}\tilde{F}^T &= PW\text{-}MSA(F_3^T) \\ &= Attention(F_3^T \times W^Q, F_3^T \times W^K, F_3^T \times W^V) \\ &= Attention(Q_3^S, K_3^S, V_3^S) \\ &= softmax(\frac{Q_3^S K_3^S}{\sqrt{d_{head}}} + M_3^T + Bias)V_3^S, \end{aligned} \tag{3}$$

where W^Q, W^K, and W^V are learned projected functions. $Bias$ is the relative position encoding operation.

Then, output features $\tilde{F}^S$ of PW-MSA will associate with $\tilde{F}^T$ for ego-motion estimation. Specifically, $\tilde{F}^S$ is linearly

Module	Layer type		K	Sample rate	MLP width
Generation of Initial Transformation	All-to-all point gathering		224	1	[128,64,64], [128,64]
	Shared MLP for FE		—	1	[128,64]
	FC1 for q_3, FC2 for t_3		—	1	[4], [3]
Iterative Refinement with PWC Structure	Pose Warp-Refinement for q_2,t_2	Attentive cost volume	4, 6	1	[128,64,64], [128,64]
		Set upconv	8	2×2	[128,64], [64]
		Shared MLP for FE_2	—	1	[128,64]
		FC1 for q_2, FC2 for t_2	—	1	[4], [3]
	Pose Warp-Refinement for q_1,t_1	Attentive cost volume	4, 6	1	[128,64,64], [128,64]
		Set upconv	8	2×2	[128,64], [64]
		Shared MLP for FE_1	—	1	[128,64]
		FC1 for q_1, FC2 for t_1	—	1	[4], [3]
	Pose Warp-Refinement for q_0,t_0	Attentive cost volume	4, 6	1	[128,64,64], [128,64]
		Set upconv	8	4×8	[128,64], [64]
		Shared MLP for FE_0	—	1	[128,64]
		FC1 for q_0, FC2 for t_0	—	1	[4], [3]

Table 2: **Detailed network parameters in pose generation and refinement.** K points are selected in the K Nearest Neighbors (KNN) of the all-to-all point gathering layer, set upconv layer, and attentive cost volume layer. MLP width means the number of output channels for each layer of MLP. The variables in the table are defined the same as in the main manuscript.

projected as $query$ ($\tilde{Q}^S$), $\tilde{F}^T$ is linearly projected as key ($\tilde{K}^T$) and $value$ ($\tilde{V}^T$). We calculate attention weights by feeding them into the same PW-MSA block as:

$$\begin{aligned}\tilde{F}_i^S &= Attention(\tilde{F}^S \times W^Q, \tilde{F}^T \times W^K, \tilde{F}^T \times W^V) \\ &= Attention(\tilde{Q}^S, \tilde{K}^S, \tilde{V}^S) \\ &= softmax(\frac{\tilde{Q}^S \tilde{K}^S}{\sqrt{d_{head}}} + M_3^S + Bias)\tilde{V}^S, \end{aligned} \tag{4}$$

$$\begin{aligned}\tilde{F}_i^T &= Attention(\tilde{F}_3^T \times W^Q, \tilde{F}_3^S \times W^K, \tilde{F}_3^S \times W^V) \\ &= Attention(\tilde{Q}^S, \tilde{K}^S, \tilde{V}^S) \\ &= softmax(\frac{\tilde{Q}^S \tilde{K}^S}{\sqrt{d_{head}}} + M_3^T + Bias)\tilde{V}^S. \end{aligned} \tag{5}$$

This step resembles the encoder-decoder conjunction in a vanilla transformer, which enables the network to learn relative position transformation between two frames. All the above processes will be repeated once in the following Point Shift Window-based Self Attention block (PSW-WSA). Then, we reverse window partition and shifting operations. Finally, correlated features $\tilde{F}_L^S$, $\tilde{F}_L^T$ are output as input for the all-to-all point gathering module.

4. Additional Experiment

4.1. Evaluation Metrics

We follow protocols of DGR [3] to evaluate our RegFormer with three metrics: (1) Relative Translation Error (RTE), the Euclidean distance between predicted and ground-truth translation vectors. (2) Relative Rotation Error (RRE), the geodesic distance between estimated and ground-truth rotation parameters. (3) Registration Recall (RR), the average success ratio of the registration. Registration is successful when RRE and RTE are within a certain threshold. RRE and RTE can be calculated as:

$$RRE = arccos\frac{Tr({R_{pred}}^T R_{gt} - 1)}{2}, \tag{6}$$

$$RTE = \|t_{gt} - t_{pred}\|_2, \tag{7}$$

where R_{pred} and t_{pred} are predicted rotation and translation vectors. R_{gt} and t_{gt} are the ground truth ones. Note that failed registrations can lead to unreliable error metrics, we only calculate RRE and RTE for successful registrations [3, 5, 4, 1].

4.2. Generalization ability on LiDAR Odometry

For evaluating the generalization ability of RegFormer, we also apply our large-scale registration network to the LiDAR odometry, which is a downstream task of point cloud registration. From Table 3, we can see our RegFormer can outperform all recent learning-based methods including LONet [6], SelfVoxelLO [13], RSLO [14] et al. by a large margin. Compared with classical methods, our network has a 25.8% lower translation error and a 25.6% lower rotation error. Notably, loop closure and mapping are not applied in our registration network, so we only compare with LOAM [15] and SUMA [2] without mapping and loop closure for a fair comparison. Experiment results show the strong generalization ability of RegFormer.

4.3. Qualitative Visualization

In this section, we will visualize more qualitative experiment results to demonstrate the registration accuracy of our

	Method	00*		01*		02*		03*		04*		05*		06*		07†		08†		09†		10†		Mean on 07-10	
		t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Classic	ICP-po2po	6.88	2.99	11.21	2.58	8.21	3.39	11.07	5.05	6.64	4.02	3.97	1.93	1.95	1.59	5.17	3.35	10.04	4.93	6.93	2.89	8.91	4.74	7.359	3.407
	ICP-po2pl	3.80	1.73	13.53	2.58	9.00	2.74	2.72	1.63	2.96	2.58	2.29	1.08	1.77	1.00	1.55	1.42	4.42	2.14	3.95	1.71	6.13	2.60	4.735	1.932
	GICP [7]	1.29	0.64	4.39	0.91	2.53	0.77	1.68	1.08	3.76	1.07	1.02	0.54	0.92	0.46	0.64	0.45	1.58	0.75	1.97	0.77	**1.31**	**0.62**	1.921	0.733
	CLS [10]	2.11	0.95	4.22	1.05	2.29	0.86	1.63	1.09	1.59	0.71	1.98	0.92	0.92	0.46	1.04	0.73	2.14	1.05	1.95	0.92	3.46	1.28	2.148	0.995
	LOAM w/o mapping [15]	15.99	6.25	3.43	0.93	9.40	3.68	18.18	9.91	9.59	4.57	9.16	4.10	8.91	4.63	10.87	6.76	12.72	5.77	8.10	4.30	12.67	8.79	10.820	5.426
	LeGO-LOAM [8]	2.17	1.05	13.4	1.02	2.17	1.01	2.34	1.18	1.27	1.01	1.28	0.74	1.06	0.63	1.12	0.81	1.99	0.94	1.97	0.98	2.21	0.92	2.49	1.00
	SuMa w/o mapping [2]	2.93	0.92	2.09	0.93	4.05	1.22	2.30	0.79	1.43	0.75	11.9	1.06	1.46	0.79	1.75	1.17	2.53	0.96	1.92	0.78	1.81	0.97	2.93	0.92
	PUMA [11]	1.46	0.68	3.38	1.00	1.86	0.72	1.60	1.10	1.63	0.92	1.20	0.61	0.88	0.42	0.72	0.55	1.44	0.61	1.51	0.66	1.38	0.84	1.55	0.74
DL-based	Zhou et al. [16]	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	21.3	6.65	21.9	2.91	18.8	3.21	14.3	3.30	19.10	4.02
	LO-Net [6]	1.47	0.72	1.36	0.47	1.52	0.71	**1.03**	**0.66**	0.51	0.65	1.04	0.69	**0.71**	0.50	1.70	0.89	2.12	0.77	1.37	0.58	1.80	0.93	1.330	0.688
	SelfVoxeLO [13]	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	2.51	1.15	2.65	1.00	2.86	1.17	3.22	1.26	2.81	1.15
	RSLO w/o mapping [14]	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	NG	2.37	1.15	2.14	0.92	2.61	1.05	2.33	0.94	2.36	1.02
	Ours	**0.88**	**0.41**	**1.17**	**0.46**	**0.91**	**0.35**	**1.03**	0.73	**0.36**	**0.21**	**0.64**	**0.43**	0.74	**0.33**	**0.57**	**0.44**	**1.34**	**0.52**	**1.09**	**0.54**	1.46	0.71	**1.115**	**0.553**

Table 3: **Comparison with the state-of-the-art on the odometry task.** t_{rel}, r_{rel} indicate the average translation RMSE (%) and rotation RMSE (°/100m) respectively on all subsequences in the length of $100, 200, ..., 800m$. '*' means the training sequence, '†' means the testing ones. 'NG' means results are not given. The best result for each sequence is **bold.**

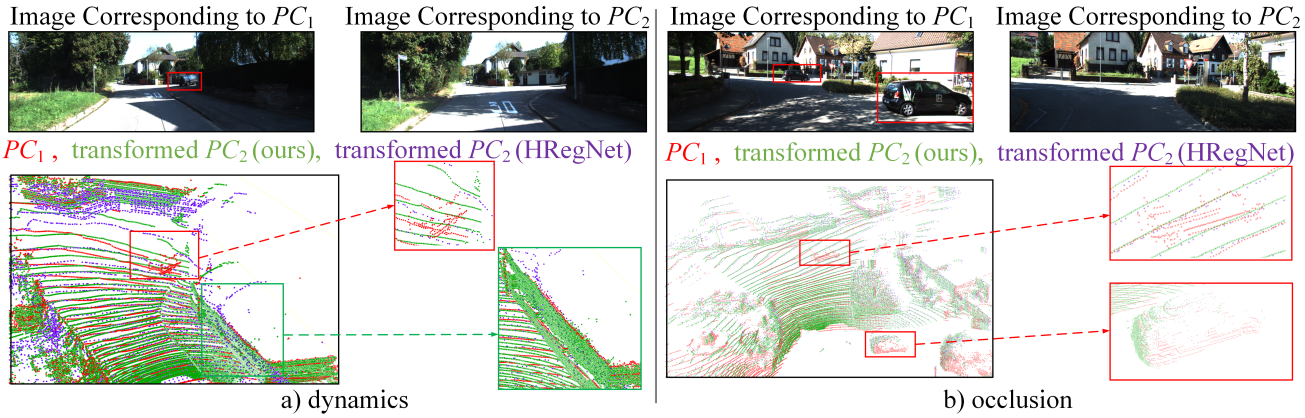

Figure 2: Visualization of traffic flow scenes. Our transformed PC_2 is highly overlapped with PC_1 on trees and buildings (static objects), while has nearly no overlap on moving cars (dynamic or occluded objects). The transformed PC_2 from HRegNet has large registration errors compared with ours.

RegFormer.

Filtering outliers in dense traffic flow. Our RegFormer has the capability to filter dynamics and occlusion with the transformer's global modeling. We visualize this benefit in dense traffic flows. As in Fig. 2 a), the car turning right is viewed as a dynamic object and harmful to the regression of transformation, while trees and buildings (static objects) are beneficial to the overall pose regression. From the visualization result, it is obvious that transformed PC_2 is almost overlapped with PC_1 on trees and buildings but has nearly no points shared with the dynamic car. Similarly, our RegFormer can also identify the locations of occluded cars in Fig. 2 b).

Registration results. First of all, we display a series of registration samples in Fig. 3. Point clouds colored red and green are input point pairs respectively. Transformed point clouds by poses from HregNet are colored pink in the second column. Transformed point clouds by our predicted poses and the ground truth ones are colored blue and purple. From the figures, we can see our predicted point clouds are almost overlapped with the ground truth ones in outdoor large-scale scenes. Also, a better performance of our method proves that transformer is more suitable for large-scale point cloud registration than CNN-based methods, eg. HRegNet.

Registration errors. Furthermore, we show more samples with registration error analysis in Fig. 4 and Fig. 5. The first row indicates the ground truth and predicted point clouds. Error vectors are indicated separately with the Bird's Eye View (BEV) in the second row and with the side view in the third row. In the main manuscript, we have already found the error distribution characteristics of the horizontal direction. In this section, we mainly summarize in which direction the registration errors are larger, with respect to the vertical and horizontal directions. As illustrated in Fig. 4 and Fig. 5, errors commonly come from the horizontal direction since the ground is a huge plane that offers sufficient information to localize the vertical position variation. In Fig. 5 e), it is more obvious that there are relatively larger errors in the horizontal direction with the Bird's Eye View (BEV), but almost no errors in the vertical direction where the ground

Algorithm 1 The cross-attention block in BAT.

Input: Uncorrelated point cloud features F_3^S, F_3^T and their corresponding masks M_3^S, M_3^T of two consecutive frames from stage 3 of the feature extraction transformer.
Parameter: Number of transformer blocks: L.
Output: Correlated point cloud features $\tilde{F}_L^S$, $\tilde{F}_L^T$.

Initialize $i = 0$. (Parameter i is used for indicating whether the cyclic shift is employed in the transformer.)
while $i <= L$ **do**
  Each point associates with other points in the same frame as:
  $\tilde{F}^S = PW\text{-}MSA(LN(Q_3^S, K_3^S, V_3^S), M_3^S) + F_3^S$.
  $\tilde{F}^T = PW\text{-}MSA(LN(Q_3^T, K_3^T, V_3^T), M_3^T) + F_3^T$.
  if $i\%2 == 0$ **then**
    Points associate with points from the other frame (without shift) as:
    $\tilde{F}_i^S = PW\text{-}MSA(LN(\tilde{Q}^S, \tilde{K}^T, \tilde{V}^T), M_3^S) + \tilde{F}^S$.
    $\tilde{F}_i^T = PW\text{-}MSA(LN(\tilde{Q}^T, \tilde{K}^S, \tilde{V}^S), M_3^T) + \tilde{F}^T$.
  else
    Points associate with points from the other frame (with shift) as:
    $\tilde{F}_i^S = PSW\text{-}MSA(LN(\tilde{Q}^S, \tilde{K}^T, \tilde{V}^T), M_3^S) + \tilde{F}^S$.
    $\tilde{F}_i^T = PSW\text{-}MSA(LN(\tilde{Q}^T, \tilde{K}^S, \tilde{V}^S), M_3^T) + \tilde{F}^T$.
  end if
  $i += 1$.
end while
return $\tilde{F}_L^S$, $\tilde{F}_L^T$

truth point cloud (yellow) has almost covered the predicted one (blue).

References

[1] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6359–6367, 2020.

[2] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*, volume 2018, page 59, 2018.

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

[4] 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.

[5] Junha Lee, Seungwook Kim, Minsu Cho, and Jaesik Park. Deep hough voting for robust global registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15994–16003, 2021.

[6] Qing Li, Shaoyang Chen, Cheng Wang, Xin Li, Chenglu Wen, Ming Cheng, and Jonathan Li. Lo-net: Deep real-time lidar odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8473–8482, 2019.

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

[8] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[9] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[10] Martin Velas, Michal Spanel, and Adam Herout. Collar line segments for fast odometry estimation from velodyne point clouds. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4486–4495. IEEE, 2016.

[11] Ignacio Vizzo, Xieyuanli Chen, Nived Chebrolu, Jens Behley, and Cyrill Stachniss. Poisson surface reconstruction for lidar odometry and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5624–5630. IEEE, 2021.

[12] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. Hierarchical attention learning of scene flow in 3d point clouds. *IEEE Transactions on Image Processing*, 30:5168–5181, 2021.

[13] Yan Xu, Zhaoyang Huang, Kwan-Yee Lin, Xinge Zhu, Jianping Shi, Hujun Bao, Guofeng Zhang, and Hongsheng Li. Selfvoxelo: Self-supervised lidar odometry with voxel-based deep neural networks, 2020.

[14] Yan Xu, Junyi Lin, Jianping Shi, Guofeng Zhang, Xiaogang Wang, and Hongsheng Li. Robust self-supervised lidar odometry via representative structure discovery and 3d inherent error modeling. *IEEE Robotics and Automation Letters*, 7(2):1651–1658, 2022.

[15] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.

[16] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017.

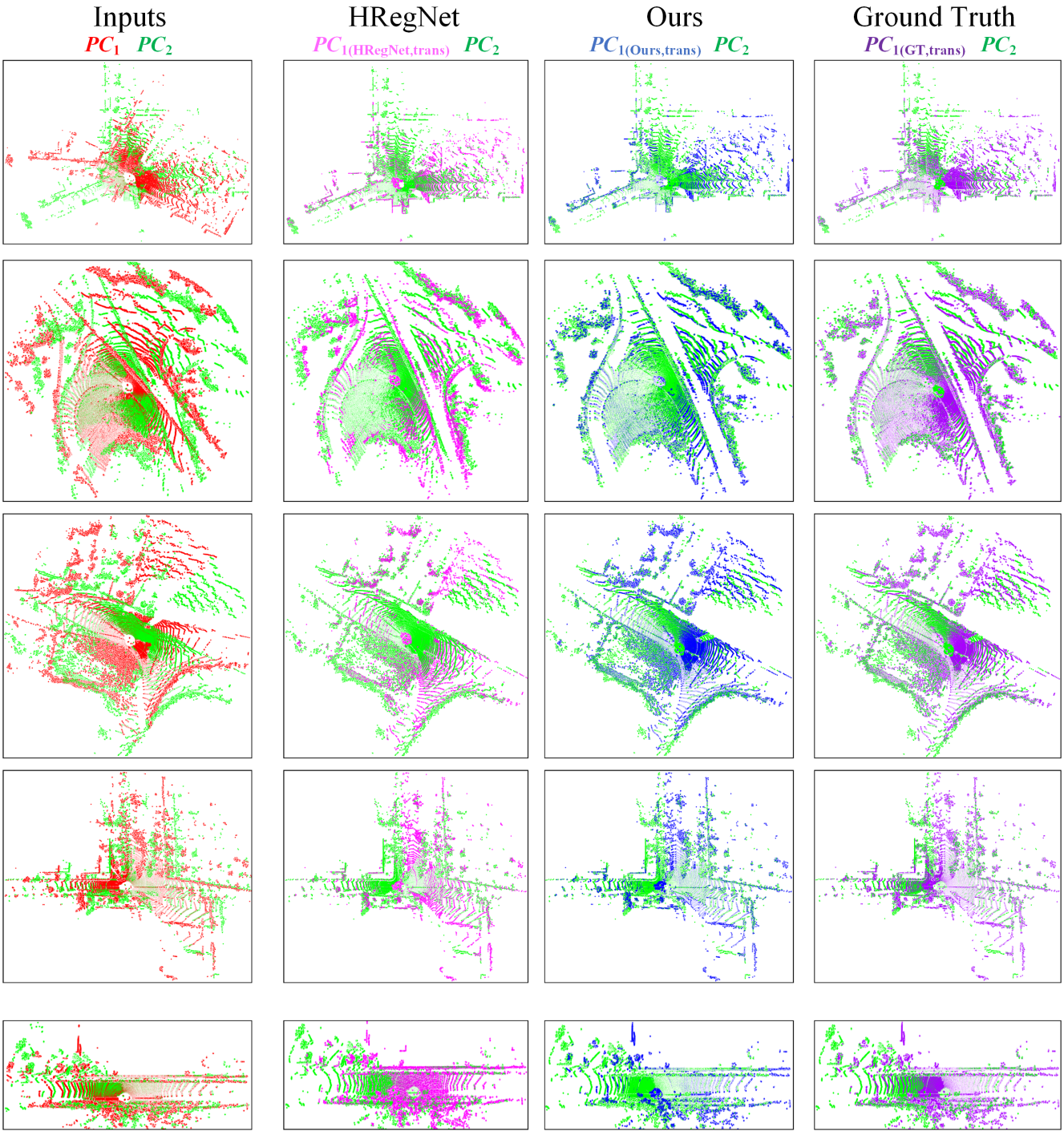

Figure 3: Qualitative visualization of our RegFormer. We display a series of registration samples of our RegFormer. Without any post-processing and local descriptors, our RegFormer can also achieve higher accuracy than HRegNet.

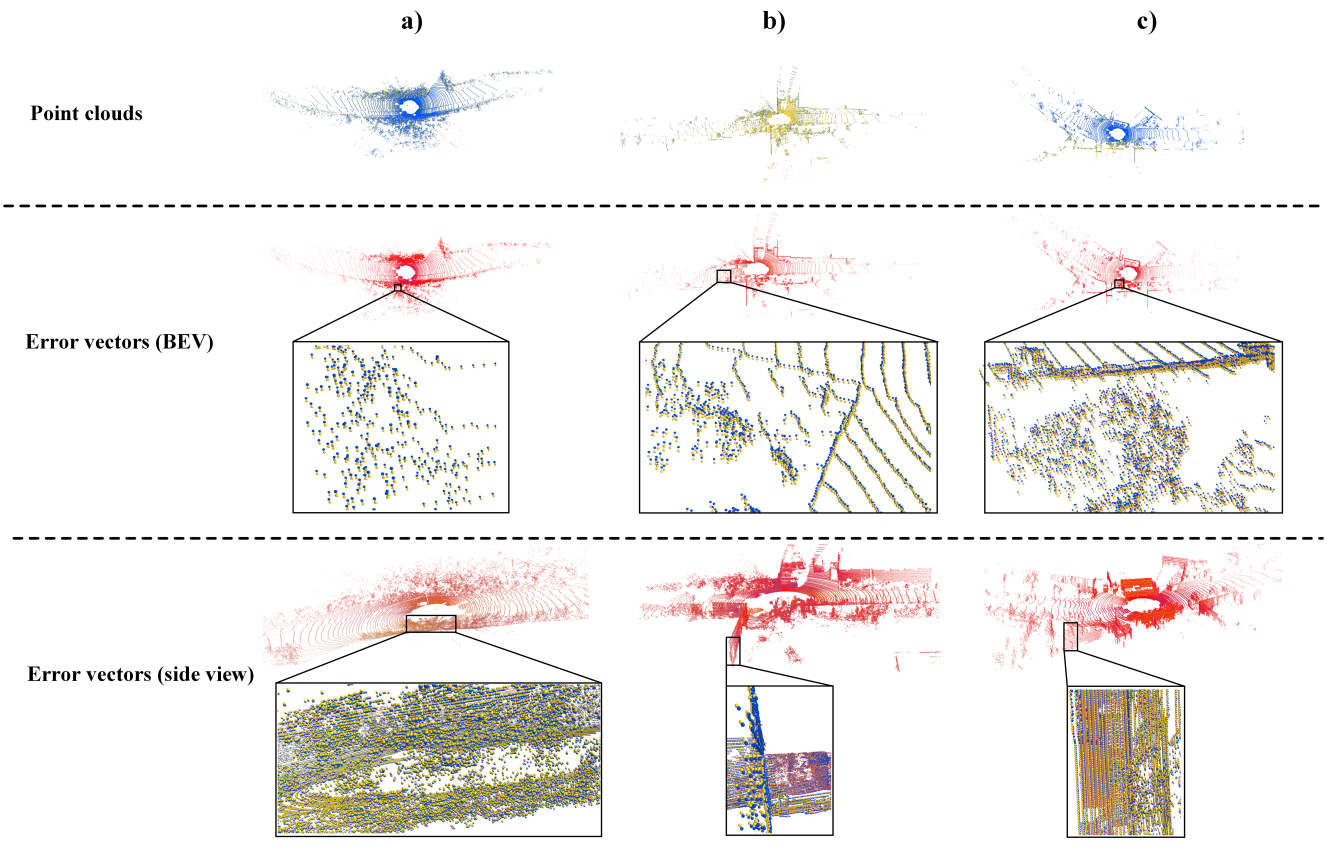

Figure 4: Visualization of registration errors (1). For the analysis of registration errors, we display more samples where error vectors (red) point from the predicted point cloud (blue) to the ground truth one (yellow). The first row indicates the ground truth and predicted point clouds. Error vectors are indicated separately with the Bird's Eye View (BEV) in the second row and with the side view in the third row.

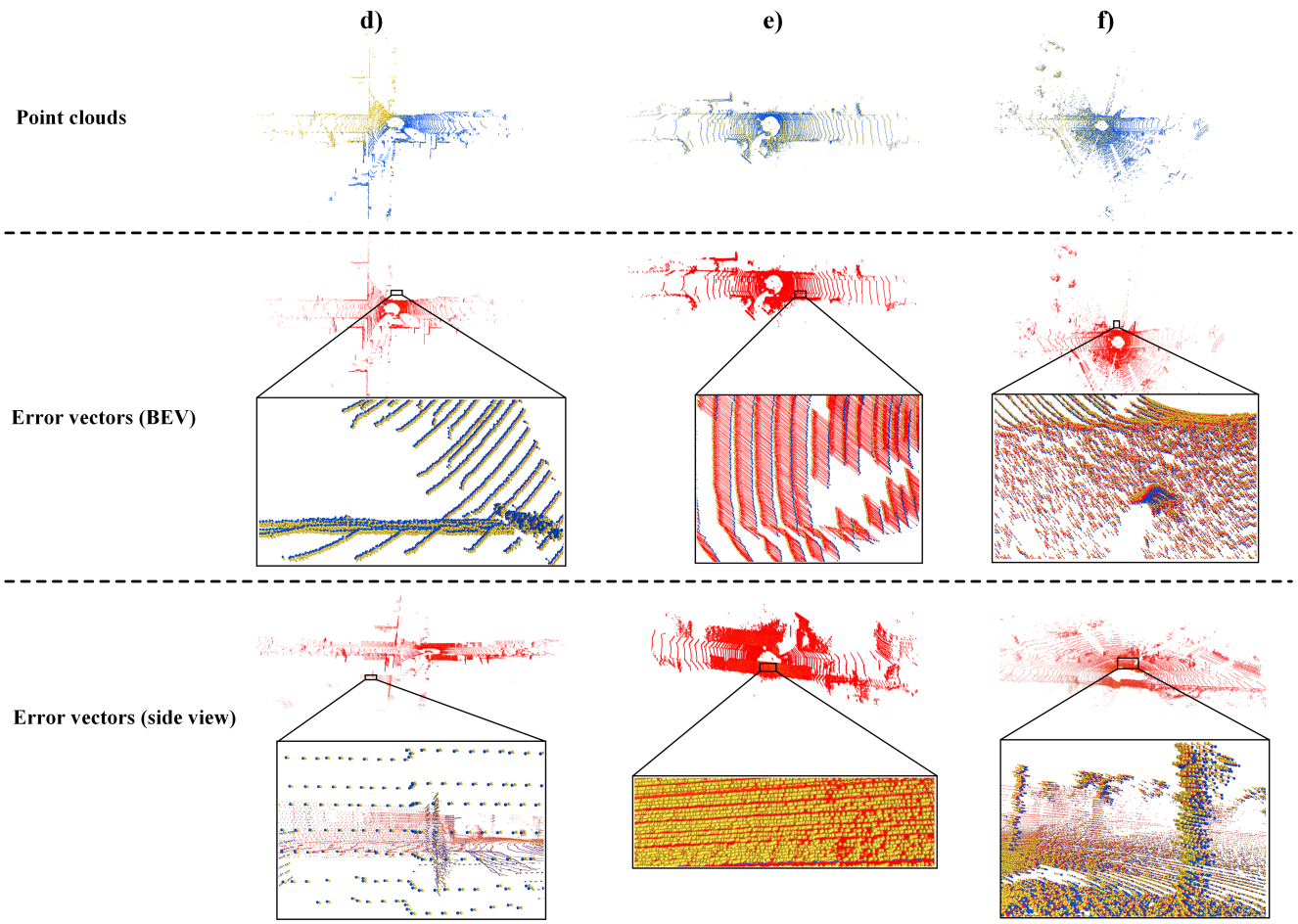

Figure 5: Visualization of registration errors (2). For the analysis of registration errors, we display more samples where error vectors (red) point from the predicted point cloud (blue) to the ground truth one (yellow). The first row indicates the ground truth and predicted point clouds. Error vectors are indicated separately with the Bird's Eye View (BEV) in the second row and with the side view in the third row.