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

Jiuming Liu¹, Guangming Wang¹,⁴, Zhe Liu², Chaokang Jiang³, Marc Pollefeys⁴,⁵, Hesheng Wang¹
¹Department of Automation, Key Laboratory of System Control and Information Processing of Ministry of Education, Shanghai Jiao Tong University
²MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
³China University of Mining and Technology
⁴ETH Zürich
⁵Microsoft
{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 (e.g. SVD) are utilized to regress the transformation through these correspondences. The second stage is commonly figured out as an optimization problem:

\[
R, t = \arg\min_{R, t} \sum_i \|Rx_i + t - y_i\|_2,
\]

(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 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-
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.

<table>
<thead>
<tr>
<th>Module</th>
<th>Layer</th>
<th>$N$</th>
<th>Sample rate</th>
<th>Channels</th>
<th>MLP width</th>
<th>Heads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Extraction</td>
<td>Set conv layer for $I^S$ and $I^T$ in stage 1</td>
<td>2</td>
<td>$[H/4 \times W/8]$</td>
<td>16</td>
<td>[16,64,16]</td>
<td>2</td>
</tr>
<tr>
<td>Transformer</td>
<td>Set conv layer for $PC^S_1$ and $PC^T_1$ in stage 2</td>
<td>2</td>
<td>$[H/8 \times W/16]$</td>
<td>32</td>
<td>[32,64,32]</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Set conv layer for $PC^S_2$ and $PC^T_2$ in stage 3</td>
<td>6</td>
<td>$[H/16 \times W/32]$</td>
<td>64</td>
<td>[64,256,64]</td>
<td>8</td>
</tr>
<tr>
<td>Bijective Association</td>
<td>Cross-attention layer in stage 3</td>
<td>6</td>
<td>—</td>
<td>64</td>
<td>[64,256,64]</td>
<td>8</td>
</tr>
</tbody>
</table>

**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.

**Bijective 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.

**Iterative 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 layer to the lower layer. 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 $F^S_3$, $F^T_3$ and their masks $M^S_3$, $M^T_3$ 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, $F^S_3$ and $M^S_3$ 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:

$$
\tilde{F}^S = PW-MSA(F^S_3) = Attention(F^S_3 \times W^Q, F^S_3 \times W^K, F^S_3 \times W^V) = Attention(Q^S_3, K^S_3, V^S_3) = softmax(Q^S_3 K^S_3^{\top} + M^S_3 + Bias)V^S_3,  
$$

(2)

$$
\tilde{F}^T = PW-MSA(F^T_3) = Attention(F^T_3 \times W^Q, F^T_3 \times W^K, F^T_3 \times W^V) = Attention(Q^T_3, K^T_3, V^T_3) = softmax(Q^T_3 K^T_3^{\top} + M^T_3 + Bias)V^T_3,  
$$

(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
**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:

\[
RRE = \arccos \frac{Tr(R_{\text{pred}}^T R_{\text{gt}} - 1)}{2},
\]

\[
RTE = \|t_{\text{gt}} - t_{\text{pred}}\|_2,
\]

where \(R_{\text{pred}}\) and \(t_{\text{pred}}\) are predicted rotation and translation vectors. \(R_{\text{gt}}\) and \(t_{\text{gt}}\) 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 LOAM [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 network.
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.

Table 3:

<table>
<thead>
<tr>
<th>Method</th>
<th>00'</th>
<th>01'</th>
<th>02'</th>
<th>03'</th>
<th>04'</th>
<th>05'</th>
<th>06'</th>
<th>07'</th>
<th>08'</th>
<th>09'</th>
<th>10'</th>
<th>Mean on 07-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICP-prepos</td>
<td>6.08</td>
<td>2.09</td>
<td>1.11</td>
<td>2.58</td>
<td>6.21</td>
<td>3.39</td>
<td>11.07</td>
<td>4.05</td>
<td>6.64</td>
<td>4.02</td>
<td>1.97</td>
<td>1.05</td>
</tr>
<tr>
<td>ICP-pos [7]</td>
<td>3.00</td>
<td>1.71</td>
<td>15.53</td>
<td>25.58</td>
<td>9.00</td>
<td>2.54</td>
<td>2.72</td>
<td>1.63</td>
<td>2.96</td>
<td>2.58</td>
<td>2.29</td>
<td>1.06</td>
</tr>
<tr>
<td>GCP [7]</td>
<td>1.29</td>
<td>0.64</td>
<td>4.39</td>
<td>0.99</td>
<td>2.53</td>
<td>0.77</td>
<td>1.68</td>
<td>1.08</td>
<td>3.76</td>
<td>1.07</td>
<td>1.02</td>
<td>0.34</td>
</tr>
<tr>
<td>CLS [10]</td>
<td>2.11</td>
<td>0.95</td>
<td>4.22</td>
<td>1.05</td>
<td>2.29</td>
<td>0.86</td>
<td>1.63</td>
<td>1.09</td>
<td>1.59</td>
<td>0.71</td>
<td>1.98</td>
<td>0.92</td>
</tr>
<tr>
<td>LOAM w/o mapping [15]</td>
<td>15.99</td>
<td>6.25</td>
<td>3.43</td>
<td>0.93</td>
<td>0.84</td>
<td>3.68</td>
<td>18.13</td>
<td>9.91</td>
<td>5.99</td>
<td>4.57</td>
<td>11.16</td>
<td>4.10</td>
</tr>
<tr>
<td>LeGo-LOAM [8]</td>
<td>2.17</td>
<td>1.05</td>
<td>13.4</td>
<td>1.02</td>
<td>2.17</td>
<td>1.01</td>
<td>2.34</td>
<td>1.18</td>
<td>1.27</td>
<td>1.01</td>
<td>1.28</td>
<td>0.74</td>
</tr>
<tr>
<td>SiMa w/o mapping [2]</td>
<td>2.93</td>
<td>0.92</td>
<td>2.09</td>
<td>0.93</td>
<td>4.05</td>
<td>1.22</td>
<td>2.50</td>
<td>0.79</td>
<td>1.43</td>
<td>0.75</td>
<td>1.19</td>
<td>1.06</td>
</tr>
<tr>
<td>PUMA [11]</td>
<td>1.46</td>
<td>0.60</td>
<td>1.78</td>
<td>1.00</td>
<td>1.86</td>
<td>0.72</td>
<td>1.60</td>
<td>1.10</td>
<td>1.65</td>
<td>0.92</td>
<td>1.20</td>
<td>0.43</td>
</tr>
<tr>
<td>Zhou et al. [16]</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
</tr>
<tr>
<td>LDNet [6]</td>
<td>1.47</td>
<td>0.73</td>
<td>2.36</td>
<td>0.47</td>
<td>1.52</td>
<td>0.71</td>
<td>1.88</td>
<td>0.66</td>
<td>0.51</td>
<td>0.65</td>
<td>1.64</td>
<td>0.60</td>
</tr>
<tr>
<td>iSegSegLO [13]</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
</tr>
<tr>
<td>RSOLO w/o mapping [14]</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
<td>NG</td>
</tr>
<tr>
<td>Ours</td>
<td>0.88</td>
<td>0.43</td>
<td>1.37</td>
<td>0.46</td>
<td>0.91</td>
<td>0.35</td>
<td>1.63</td>
<td>0.73</td>
<td>0.36</td>
<td>0.21</td>
<td>0.64</td>
<td>0.43</td>
</tr>
</tbody>
</table>

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

References


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.