Supplementary Material
Deep Graph-based Spatial Consistency for Robust Non-rigid Point Cloud Registration

Zheng Qin[1] Hao Yu[2] Changjian Wang[1] Yuxing Peng[1] Kai Xu[1*]
[1]National University of Defense Technology [2]Technical University of Munich

This supplementary material provides the implementation details of GraphSCNet and the baselines (Appx. A), the details of the metrics (Appx. B), more experiments and analysis (Appx. C), the details of N-ICP to estimate the warping function (Appx. D), and discusses the limitations of our method (Appx. E).

A. Implementation Details

Network architecture. In the initial feature embedding, we use a three-layer MLP with (256, 256, 256) channels to project the correspondence embedding to a high-dimension representation. Group normalization [12] and LeakyReLU are used after each layer in the MLP.

Unless otherwise noted, we use 3 correspondence embedding modules to generate the spatial-consistency-aware features, while each contains 2 SCA-SA modules. All layers in the models have $d = 256$ feature channels. The node coverage is $\sigma_n = 0.08$m. For each correspondence, we use $k = 6$ neighboring nodes to construct the graph. And the distance tolerance when computing spatial consistency is $\sigma_d = 0.08$m.

At last, we adopt another three-layer MLP with (128, 64, 1) channels to classify each correspondences. Group normalization [12] and LeakyReLU are used after the first two layers in the MLP, and sigmoid activation is applied after the last layer. We select the correspondences whose confidence scores are above $\tau_s = 0.4$ as inliers and the others are removed as outliers.

Baselines. For the baseline models PointCN [9] and PointDSC [1], the initial feature embedding and the classification head are the same as aforementioned. In PointCN, we replace the correspondence embedding modules with 6 MLP blocks, each of which consists of two linear layers with residual connection. In PointDSC, we use 6 SC-NonLocal [1] modules to learn the correspondence features. Due to memory limit, we randomly sample 2048 input correspondences in PointDSC. The architectures of different models are illustrated in Fig. 1. All the layers in the baseline models have 256 feature channels as in GraphSCNet.

Training and testing. We implement and evaluate our method with PyTorch [10] on an NVIDIA 2080Ti GPU. The models are trained with Adam optimizer [4] for 40 epochs. The batch size is 1 and the weight decay is 10^{-6}. The learning rate starts from 10^{-4} and decays exponentially by 0.05 after each epoch. During training, we regard the correspondences as inliers if their residuals under the ground-truth deformation are below $\tau = 0.04$m, and outliers otherwise. For data augmentation, we adopt a relatively weak data augmentation as in [13] with a random rotation sampled from $[0, 10^\circ]$ and a random translation sampled from $\mathcal{N}(0, 0.05)$.

In the experiments on 4DMatch, as the training data has been used to train the correspondence extractor, the putative correspondences on the training set are almost all inliers. In this case, the training data cannot provide effective supervision to train an outlier rejection network. To solve this problem, we split the official validation sequences by 90%/10% for training/validation, respectively, and evaluate the models on the official testing squences.

B. Metrics

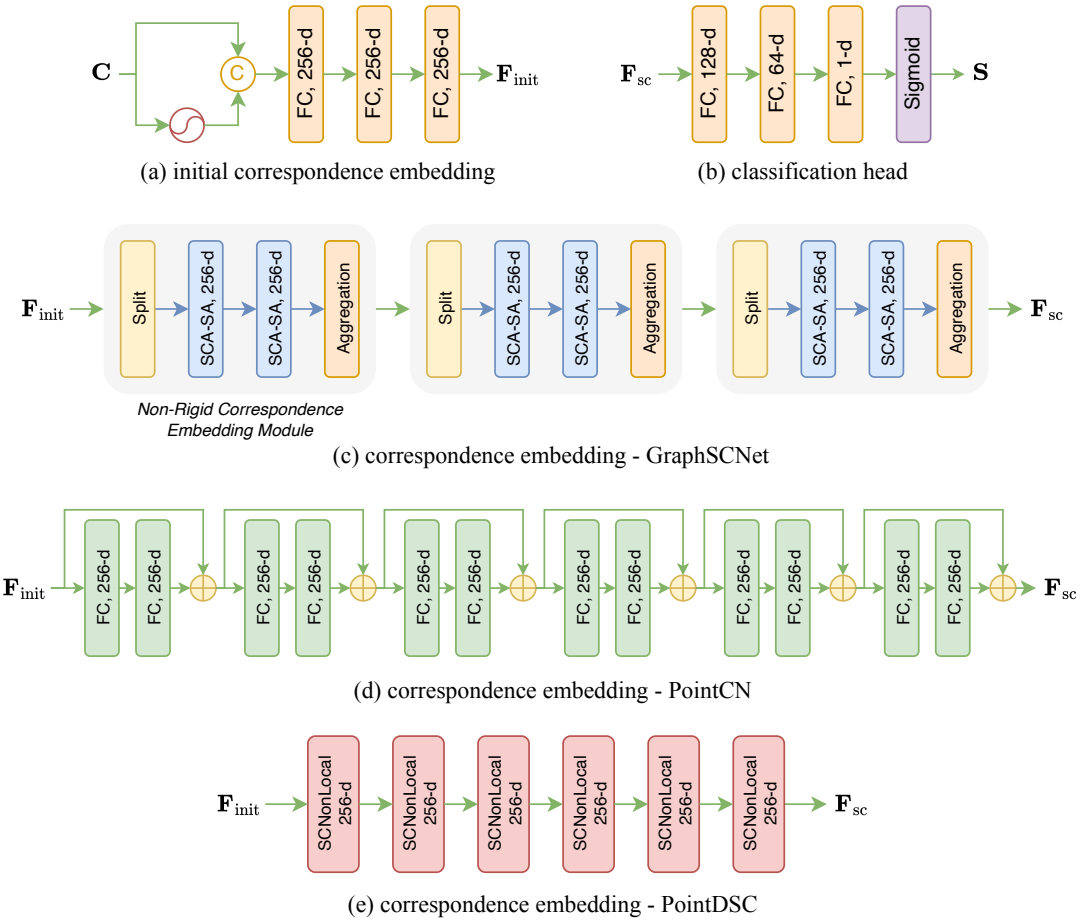
Following [7], we mainly evaluate our method using 4 metrics: 3D End Point Error, 3D Accuracy Strict, 3D Accuracy Relaxed and Outlier Ratio.

3D End Point Error (EPE) measures the average error over all warped points under the estimated and the ground-truth warping functions $\mathcal{W}(\cdot)$ and $\mathcal{W}^*(\cdot)$:

$$\texttt{EPE} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}_i \in \mathcal{P}} \|\mathcal{W}(\mathbf{p}_i) - \mathcal{W}^*(\mathbf{p}_i)\|_2. \tag{1}$$

3D Accuracy Strict (AccS) and *3D Accuracy Relaxed* (AccR) measure the fractions of points whose EPEs are below a EPE threshold or relative errors are below a relative error threshold. For AccS, the EPE threshold is 2.5cm and the relative error threshold is 2.5%. For AccR, the EPE threshold is 5cm and the relative error threshold is 5%. The

*Corresponding author: kevin.kai.xu@gmail.com.



(a) initial correspondence embedding

(b) classification head

(c) correspondence embedding - GraphSCNet

(d) correspondence embedding - PointCN

(e) correspondence embedding - PointDSC

Figure 1. Network architecture.

relative error is computed as:

$$\texttt{RE}(\mathbf{p}_i) = \frac{\|\mathcal{W}(\mathbf{p}_i) - \mathcal{W}^*(\mathbf{p}_i)\|_2}{\|\mathcal{W}^*(\mathbf{p}_i) - \mathbf{p}_i\|_2}. \tag{2}$$

And the 3D accuracy is defined as:

$$\texttt{AccS} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}_i \in \mathcal{P}} [\![\texttt{EPE}(\mathbf{p}_i) < 2.5\text{cm} \vee \texttt{RE}(\mathbf{p}_i) < 2.5\%]\!], \tag{3}$$

$$\texttt{AccR} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}_i \in \mathcal{P}} [\![\texttt{EPE}(\mathbf{p}_i) < 5\text{cm} \vee \texttt{RE}(\mathbf{p}_i) < 5\%]\!], \tag{4}$$

where $[\![\cdot]\!]$ is the Inversion bracket.

Outlier Ratio (OR) measures the fraction of points which are not successfully registered. Following [7], a point is regarded as a failure if its relative error is above 30%:

$$\texttt{OR} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}_i \in \mathcal{P}} [\![\texttt{RE}(\mathbf{p}_i) > 30\%]\!] \tag{5}$$

C. Additional Experiments

C.1. Evaluations on Low-Inlier-Ratio Cases

To evaluate the performance in low-inlier-ratio scenarios, we add random outliers into the correspondences from GeoTransformer, making the final inlier ratio less than 30%. In Tab. 1, PointDSC and PointCN fail to achieve reasonable registration results due to enormous outliers. In contrast, our method still achieves promising results, showing strong generality to low-inlier-ratio cases.

C.2. Evaluations on Large-Deformation Cases

Next, we investigate the performance when the deformations are large. As there is no off-the-shelf benchmarks with large deformations, we evaluate our method on the testing pairs whose mean residuals are above 15cm on 4DMatch. In Tab. 2, our method significantly outperforms the baselines, demonstrating its efficacy under large deformations.

Model	4DMatch				4DLoMatch			
	Prec	Recall	AccS	AccR	Prec	Recall	AccS	AccR
GraphSCNet	**91.9**	69.7	**54.5**	**66.5**	**81.8**	70.6	**26.9**	**38.6**
PointDSC	55.3	81.5	5.8	12.7	50.6	74.6	4.8	10.5
PointCN	44.8	**85.1**	3.1	10.4	42.3	**74.8**	3.4	8.9
w/o outlier rejection	29.3	100.0	0.5	1.9	25.7	100.0	1.0	2.9

Table 1. Evaluations on 4DMatch and 4DLoMatch with low inlier ratios.

Model	4DMatch				4DLoMatch			
	Prec	Recall	AccS	AccR	Prec	Recall	AccS	AccR
GraphSCNet	**89.1**	**93.8**	**57.1**	**71.5**	**77.8**	**78.4**	**28.6**	**42.7**
PointDSC	83.3	90.5	53.6	68.7	66.4	76.7	25.7	40.1
PointCN	80.0	87.5	51.1	67.0	62.4	75.5	23.3	38.4
w/o outlier rejection	76.4	100.0	51.7	68.3	56.2	100.0	23.5	38.9

Table 2. Evaluations on 4DMatch and 4DLoMatch with large deformations.

Model	4DMatch				4DLoMatch			
	Prec	Recall	AccS	AccR	Prec	Recall	AccS	AccR
(a.1) Euclidean	**92.2**	**96.9**	**72.3**	**84.4**	**82.6**	**86.8**	**41.0**	**58.3**
(a.2) Geodesic	91.2	96.4	71.1	83.5	80.7	85.6	39.8	57.4
(b.1) XYZ+Fourier	**92.2**	96.9	**72.3**	**84.4**	**82.6**	86.8	**41.0**	**58.3**
(b.2) XYZ	**92.2**	96.6	**72.3**	**84.4**	82.1	86.1	40.8	58.0
(b.3) Fourier	91.3	**97.2**	71.9	84.2	79.8	**87.1**	40.2	57.5
(c.1) w/ FL w/ CL*	**92.2**	**96.9**	**72.3**	**84.4**	**82.6**	**86.8**	**41.0**	**58.3**
(c.2) w/ FL w/o CL	90.5	96.4	71.3	83.7	77.5	84.4	38.5	55.8
(c.3) w/ BCE w/o CL	79.5	93.1	64.3	78.2	55.4	75.4	28.2	44.1
(d.1) w/ local SC	**92.2**	**96.9**	**72.3**	**84.4**	**82.6**	**86.8**	**41.0**	**58.3**
(d.2) w/o local SC	90.5	97.0	71.3	83.6	78.9	88.7	39.6	56.9

Table 3. Additional Ablation studies on 4DMatch and 4DLoMatch. Asterisk (*) indicates the default settings in our method. **FL**: focal loss. **CL**: consistency loss. **BCE**: binary cross-entropy loss. **Boldfaced** numbers highlight the best and the second best are underlined.

C.3. Additional Ablation Studies

Euclidean distance *vs*. geodesic distance. We first replace the distance metric in building deformation graph from Euclidean distance to geodesic distance. Each correspondence is assigned to its $k = 6$ nearest neighbors in the geodesic space. Note that we still use Euclidean distance during N-ICP for fair comparison. As shown in Tab. 3 (a), geodesic distance consistently degrades the performance. Compared to the Euclidean distance, the geodesic distance is less robust to occlusion as the points on the geodesic shortest path between two points can be missing. On the contrary, according to local rigidity, Euclidean distance is approximatedly preserved near each graph node, but is more robust and efficient.

Positional embedding. Next, we study the impact of the positional embedding used in the initial feature embedding in Tab. 3 (b). We first ablate the the fourier positional encoding and use only the point coordinates. This model achieves similar results on 4DMatch and slightly worse results on 4DLoMatch. We then ablate the point coordinates and use only the fourier positional encoding. This model achieves better recall but worse precision, especially in low-overlap scenarios. And the model with the both terms achieve the best results.

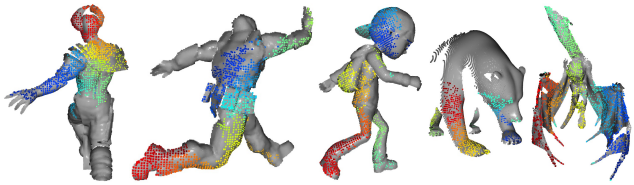


Figure 2. Feature distribution of inliers.

Loss functions. We further study the efficacy of the loss functions in Tab. 3 (c). We first ablate the feature consistency loss, which degrades the classification performance especially in low-overlap scenarios. Explicitly supervising the feature consistency between correspondences helps learn more discriminative features between inliers and outliers and thus contributes to better performance. Next we replace the binary focal loss with a binary cross-entropy loss, which significantly decreases the performance. As the putative correspondences are commonly extremely unbalanced, either predominated by inliers or outliers, cross-entropy loss hampers the convergency of the model.

Local spatial consistency. At last, we ablate the local spatial consistency in the self-attention. In Tab. 3 (d), removing the local spatial consistency considerably decreases the performance, especially in low-overlap scenarios. We also note that this model surpasses PointCN and PointDSC, indicating the efficacy of our deformation graph-based design.

C.4. Qualitative Results

We visualize the features of the detected true inliers by t-SNE. In Fig. 2, the inliers in different parts have different features, while the spatially-near ones also lie closely in the feature space. These results indicate that our method effectively learns the local motions in different parts.

We then provide more qualitative comparisons of the filtered correspondences on 4DMatch (Fig. 3), CAPE (Fig. 5) and DeepDeform (Fig. 4). Benefitting to the powerful local spatial consistency, GraphSCNet removes more outlier correspondences and achieves better inlier ratio (precision) than the baseline methods, especially under large deformations. Moreover, albeit achieving promising precision, PointDSC fails to preserve sufficient inliers. On the contrary, our method achieves both high precision and high recall, indicating it can effectively reject most outliers while better preserving inliers.

D. Deformation Estimation

Given the source point cloud $\mathcal{P}$, the target point cloud $\mathcal{Q}$, and the correspondences $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{y}_i) \mid \mathbf{x}_i \in \mathcal{P}, \mathbf{y}_i \in \mathcal{Q}\}$ between them, we adopt embedded deformation [11] to formulate the warping function. It parameterizes the deformation on the deformation graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The graph nodes $\mathcal{V}$ are sampled from the *source* point cloud with uniform furthest point sampling and the node coverage is $\sigma_g = 0.08$m. Each point $\mathbf{p}_i$ in the source point cloud is assigned to its $k_g = 6$ nearest nodes $\mathcal{K}_i$. Two nodes are connected by an undirected edge if they share a common point. Each node $\mathbf{v}_j$ is associated with a local rigid transformation $\{\mathbf{R}_j, \mathbf{t}_j\}$. And the warping function $\mathcal{W}$ is then approximated as:

$$\mathcal{W}(\mathbf{p}_i) = \sum_{\mathbf{v}_j \in \mathcal{V}} \alpha_{i,j} \big(\mathbf{R}_j(\mathbf{p}_i - \mathbf{v}_j) + \mathbf{t}_j + \mathbf{v}_j\big),$$

where $\alpha_{i,j}$ is the skinning factor as defined in [8]:

$$\alpha_{i,j} = [\![\mathbf{v}_j \in \mathcal{K}_i]\!] \cdot \frac{\exp(-\|\mathbf{p}_i - \mathbf{v}_j\|^2/(2\sigma_n^2))}{\sum_{\mathbf{v}_k \in \mathcal{K}_i} \exp(-\|\mathbf{p}_i - \mathbf{v}_k\|^2/(2\sigma_n^2))},$$

where $[\![\cdot]\!]$ is the Iverson bracket. We then solve for $\mathcal{W}$ by minimizing the following objective function:

$$E = \lambda_c E_{\text{corr}} + \lambda_r E_{\text{reg}},$$

where E_{corr} is the mean squared residual between the correspondences and E_{reg} is an as-rigid-as-possible [3] regularization term to constrain the smoothness of deformations:

$$E_{\text{corr}} = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}} \|\mathcal{W}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2,$$

$$E_{\text{reg}} = \sum_{(\mathbf{v}_u, \mathbf{v}_v) \in \mathcal{E}} \|\mathbf{R}_u(\mathbf{v}_v - \mathbf{v}_u) + \mathbf{v}_u + \mathbf{t}_u - (\mathbf{v}_v + \mathbf{t}_v)\|_2^2.$$

The weights to balance the two terms are set to $\lambda_c = 25$ and $\lambda_r = 1$, respectively.

This problem can be efficiently solved by Non-rigid ICP (N-ICP) algorithm [5, 11]. Following [2, 6], we update the associated rigid transformations incrementally:

$$\mathbf{R}_j^{(t)} = \Delta\mathbf{R}_j^{(t)} \cdot \mathbf{R}_j^{(t-1)},$$
$$\mathbf{t}_j^{(t)} = \mathbf{t}_j^{(t-1)} + \Delta\mathbf{t}_j^{(t)},$$

where $\mathbf{R}_j^{(0)} = \mathbf{I}$ and $\mathbf{t}_j^{(0)} = \mathbf{0}$. For simplicity, we omit the superscript (t) in the following text. The residual rotations are formulated in the axis-angle representation $\Delta\mathbf{R}_j = \exp(\boldsymbol{\omega}_j^\wedge)$, where $\exp(\cdot)$ is the exponential map function and $(\cdot)^\wedge$ computes the skew-symmetric matrix of a 3-d vector. We then solve for $\{\boldsymbol{\omega}_j, \Delta\mathbf{t}_j\}$ with Gauss-Newton algorithm. The residual terms are computed as:

$$\mathbf{r}_{\text{corr}}^i = \sqrt{\lambda_c}\big(\mathcal{W}(\mathbf{x}_i) - \mathbf{y}_i\big),$$
$$\mathbf{r}_{\text{reg}}^i = \sqrt{\lambda_r}\big(\mathbf{R}_u(\mathbf{v}_v - \mathbf{v}_u) + \mathbf{v}_u + \mathbf{t}_u - (\mathbf{v}_v + \mathbf{t}_v)\big).$$

where $c_i = (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}$ and $e_i = (\mathbf{v}_u, \mathbf{v}_v) \in \mathcal{E}$.

Next, we compute the partial derivatives of $\{\boldsymbol{\omega}_j, \Delta\mathbf{t}_j\}$. As $\boldsymbol{\omega}_j$ is a residual rotation, it is expected to near $\mathbf{0}$ and thus we approximate its partial derivatives with those at $\mathbf{0}$:

$$\frac{\partial \mathcal{W}(\mathbf{p}_i)}{\partial \boldsymbol{\omega}_j} \approx \frac{\partial \mathcal{W}(\mathbf{p}_i)}{\partial \boldsymbol{\omega}_j}\bigg|_{\mathbf{0}} = -\alpha_{i,j}\big(\mathbf{R}_j^{(t-1)}(\mathbf{x}_i - \mathbf{v}_j)\big)^\wedge,$$
$$\frac{\partial \mathcal{W}(\mathbf{p}_i)}{\partial \Delta\mathbf{t}_j} = \alpha_{i,j}\mathbf{I}.$$

To this end, the partial derivatives are computed as:

$$\frac{\partial \mathbf{r}_{\text{corr}}^i}{\partial \boldsymbol{\omega}_j} = -\sqrt{\lambda_c}\alpha_{i,j}\big(\mathbf{R}_j^{(t-1)}(\mathbf{x}_i - \mathbf{v}_j)\big)^\wedge,$$
$$\frac{\partial \mathbf{r}_{\text{corr}}^i}{\partial \Delta\mathbf{t}_j} = \sqrt{\lambda_c}\alpha_{i,j}\mathbf{I},$$
$$\frac{\partial \mathbf{r}_{\text{reg}}^i}{\partial \boldsymbol{\omega}_u} = -\sqrt{\lambda_r}\big(\mathbf{R}_u^{(t-1)}(\mathbf{v}_v - \mathbf{v}_u)\big)^\wedge,$$
$$\frac{\partial \mathbf{r}_{\text{reg}}^i}{\partial \Delta\mathbf{t}_u} = \sqrt{\lambda_r}\mathbf{I},$$
$$\frac{\partial \mathbf{r}_{\text{reg}}^i}{\partial \Delta\mathbf{t}_v} = -\sqrt{\lambda_r}\mathbf{I}.$$

We denote the collection of the residual terms as:

$$\mathbf{r} = [(\mathbf{r}_{\text{corr}}^1)^T, ..., (\mathbf{r}_{\text{corr}}^{|\mathcal{C}|})^T, (\mathbf{r}_{\text{reg}}^1)^T, ..., (\mathbf{r}_{\text{reg}}^{|\mathcal{E}|})^T]^T \in \mathbb{R}^{3|\mathcal{C}|+3|\mathcal{E}|},$$

the collections of variables $\{\boldsymbol{\omega}_j, \Delta\mathbf{t}_j\}$ as:

$$\Delta\mathbf{T} = [\boldsymbol{\omega}_1^T, ..., \boldsymbol{\omega}_{|\mathcal{V}|}^T, \Delta\mathbf{t}_1^T, ..., \Delta\mathbf{t}_{|\mathcal{V}|}^T]^T \in \mathbb{R}^{6|\mathcal{V}|},$$

and the Jaccobian matrix between $\mathbf{r}$ and $\Delta\mathbf{T}$ is denoted as $\mathbf{J} \in \mathbb{R}^{(3|\mathcal{C}|+3|\mathcal{E}|)\times(6|\mathcal{V}|)}$ following the computation of derivatives above. $\Delta\mathbf{T}$ can then be computed by solving the linear system:

$$(\mathbf{J}^T\mathbf{J} + \lambda_m\mathbf{I})\Delta\mathbf{T} = \mathbf{J}^T\mathbf{r}.$$

where $\lambda_m = 0.01$ is the Marquardt factor.

E. Limitations

Our method could have the following two potential limitations. First, our method serves as a post outlier rejection step after the correspondence extractor. To this end, our method is able to make given correspondences as clean as possible, but cannot infer new correspondences and improve the coverage of the correspondences on point clouds. Second, our method is based on deformation graph and local rigidity of deformations, so it could have difficulty in modeling sudden changes of geometric structures. We would leave these for future work.

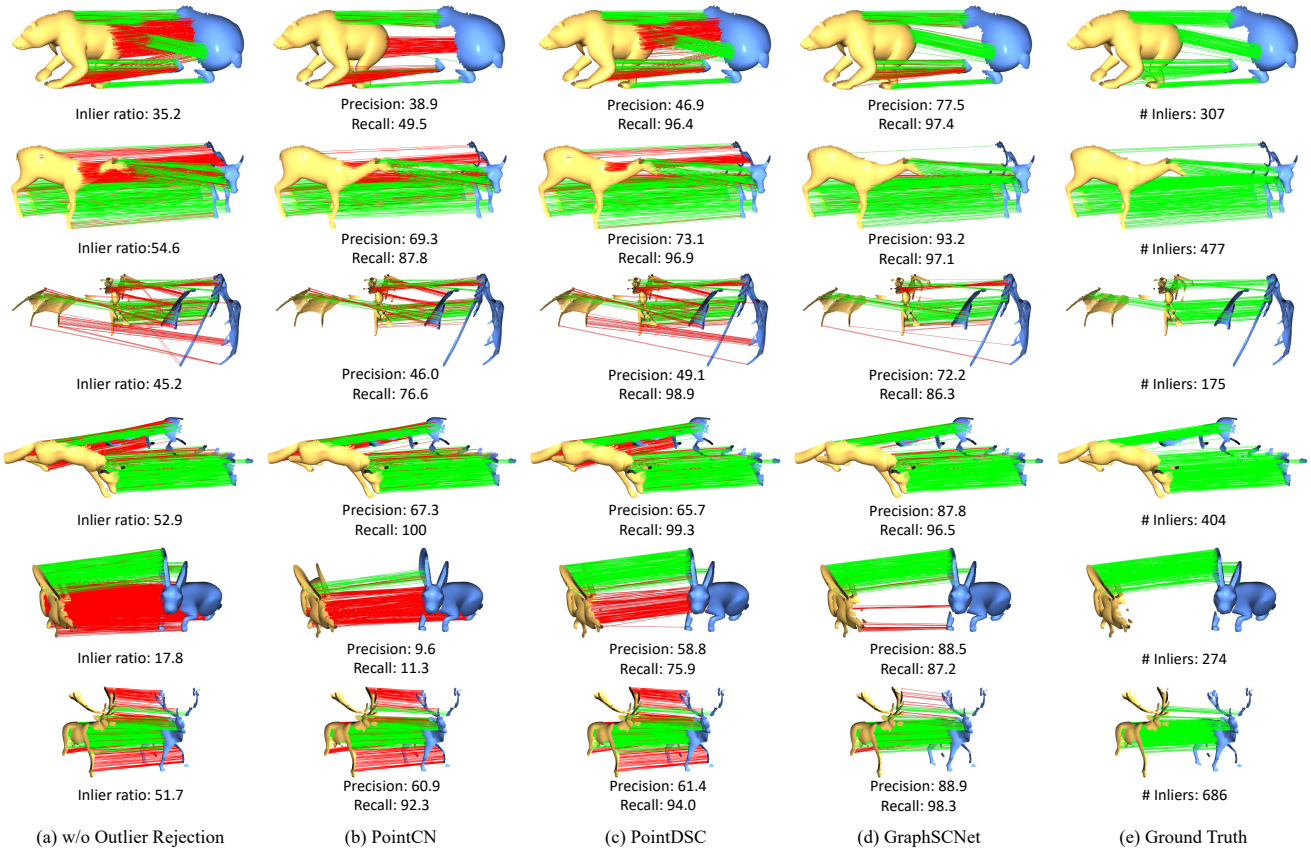


Figure 3. Comparison of different methods on 4DMatch and 4DLoMatch.

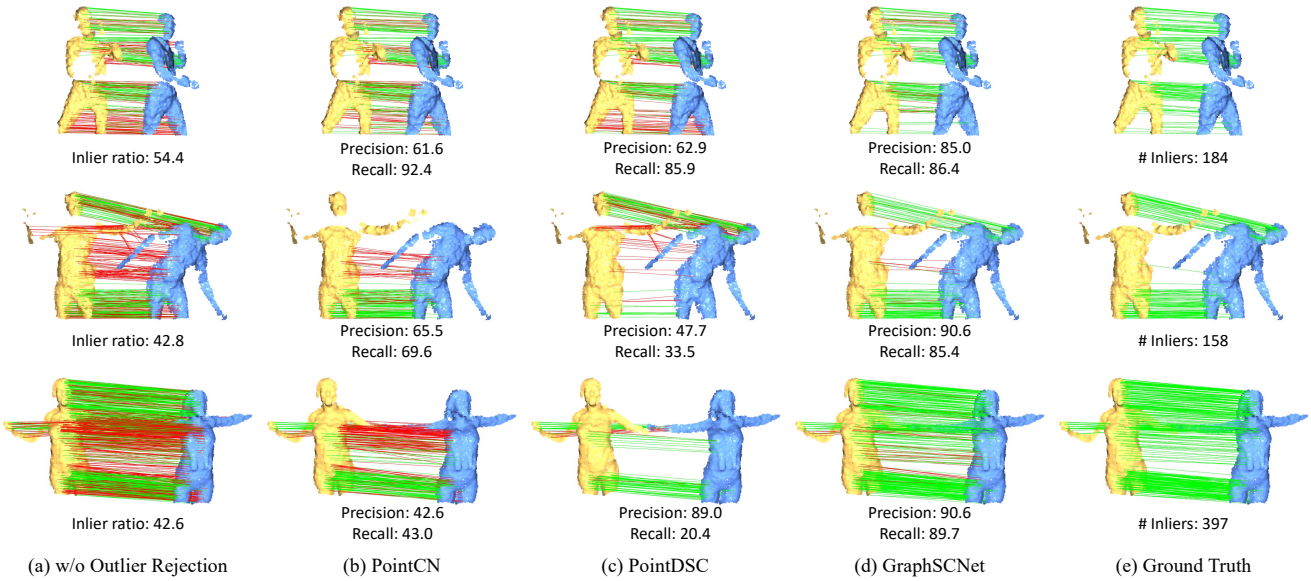


Figure 4. Comparison of different methods on DeepDeform.

References

[1] Xuyang Bai, Zixin Luo, Lei Zhou, Hongkai Chen, Lei Li, Zeyu Hu, Hongbo Fu, and Chiew-Lan Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. In *CVPR*, pages 15859–15869, 2021. 1

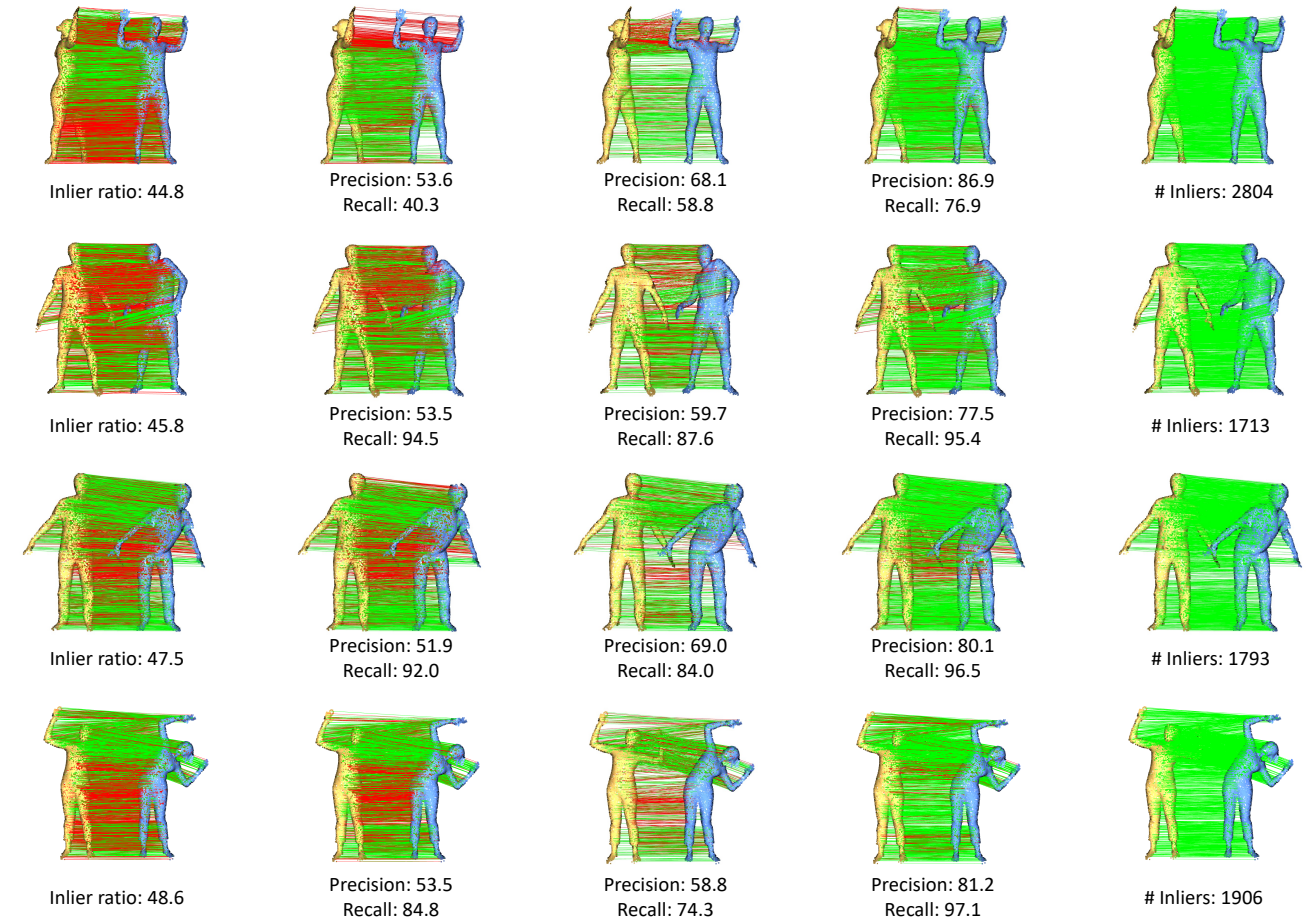


Figure 5. Comparison of different methods on DeepDeform.

[2] Aljaz Bozic, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking. *NeurIPS*, 33:18727–18737, 2020. 4

[3] Takeo Igarashi, Tomer Moscovich, and John F Hughes. As-rigid-as-possible shape manipulation. *ACM TOG*, 24(3):1134–1141, 2005. 4

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[5] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, volume 27, pages 1421–1430. Wiley Online Library, 2008. 4

[6] Yang Li and Tatsuya Harada. Lepard: Learning partial point cloud matching in rigid and deformable scenes. In *CVPR*, pages 5554–5564, 2022. 4

[7] Yang Li and Tatsuya Harada. Non-rigid point cloud registration with neural deformation pyramid. In *NeurIPS*, 2022. 1, 2

[8] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015. 4

[9] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *CVPR*, pages 7193–7203, 2020. 1

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32:8026–8037, 2019. 1

[11] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *SIGGRAPH*, pages 80–es. 2007. 4

[12] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, pages 3–19, 2018. 1

[13] Zi Jian Yew and Gim Hee Lee. Regtr: End-to-end point cloud correspondences with transformers. In *CVPR*, pages 6677–6686, 2022. 1