7. Supplementary Material

7.1. Implementation Details of PointDSC

We provide additional information about the implementation and training details of our PointDSC. The source code will be made publicly available after the paper gets accepted.

Post-refinement. Alg. 1 shows the pseudo-code of our post-refinement step. Inspired by [7], we iteratively alternate between weighing the correspondences and computing the transformation, to improve the accuracy of the transformation matrices. The inlier threshold τ is set to 10cm and 60cm for 3DMatch and KITTI, respectively. We set the maximum iteration number to 20.

Algorithm 1: Post-Refinement Algorithm

Input: $\hat{\mathbf{R}}, \hat{\mathbf{t}}$: initial transformation; $\mathbf{X}, \mathbf{Y}$
Output: $\hat{\mathbf{R}}, \hat{\mathbf{t}}$: refined transformation.
Parameter: τ.
if $iter < max_{iter}$ **then**
  # Compute the residual and the inlier num.
  $res_i = ||\hat{\mathbf{R}}\boldsymbol{x_i} + \hat{\mathbf{t}} - \boldsymbol{y_i}||_2$
  $w_i = [\![res_i < \tau]\!]$
  $num = \sum w_i$
  # If inlier num does not change, then stop.
  if $\Delta num = 0$ **then**
    break
  else
    # Compute the weighting term.
    $\phi_i = (1 + (\frac{res_i}{\tau})^2)^{-1}$
    # Estimate transformation.
    $\hat{\mathbf{R}}, \hat{\mathbf{t}} =$
     $\arg\min_{\mathbf{R},\mathbf{t}} \sum_i^N \phi_i w_i \left\|\mathbf{R}\boldsymbol{x_i} + \mathbf{t} - \boldsymbol{y_i}\right\|^2$
  $iter = iter + 1$
else
  break

Calculation of M. In Sec. 3.4 of the main text, we calculate the compatibility between correspondences by multiplying the spatial consistency term and the feature similarity term mainly because of its simplicity and good performance. Other fusion schemes such as the weighted arithmetic average and weighted geometric average can also be used to define the compatibility metric. We have explored several alternatives but found only a marginal performance difference.

Power iteration algorithm. The power iteration algorithm can compute the leading eigenvector $\boldsymbol{e}$ of the matrix $\mathbf{M}$ in several iterations. For $\mathbf{M} \in \mathbb{R}^{k \times k}$, the power iteration operator is

$$\boldsymbol{e}^{\text{iter+1}} = \frac{\mathbf{M}\boldsymbol{e}^{\text{iter}}}{\|\mathbf{M}\boldsymbol{e}^{\text{iter}}\|}. \tag{12}$$

We initialize $\boldsymbol{e}^{\mathbf{0}} = \mathbf{1}$. By iterating Eq. 12 until convergence, we get the vector $\boldsymbol{e}$, whose elements can take real values in $[0, 1]$. In practice, we find that the power iteration algorithm usually converges in fewer than five iterations.

Data augmentation. During training, we apply data augmentation, including adding Gaussian noise with standard deviation of 0.005, random rotation angle $\in [0°, 360°)$ around an arbitrary axis, and random translation $\in [-0.5m, 0.5m]$ around each axis.

Hyper-parameters. The hyper-parameter σ_d controls the sensitivity to length difference, serving as a pairwise counterpart of the unary inlier threshold τ. The larger σ_d, the more length difference between two pairs of correspondences we can accommodate. It is set manually for a specific scene and kept fixed. Picking a scene-specific value of σ_d is easy due to its clear physical meaning. However, σ_f controlling the sensitivity to feature difference has no clear physical meaning. We thus leave σ_f to be learned by the network.

7.2. Implementation Detail of Baseline Methods

The baseline methods RANSAC [24] and FGR [82] have been implemented in Open3D [83]. For GC-RANSAC [5] and TEASER [71], we use the official implementations. Note that we use TEASER with reciprocal check; otherwise, it takes an extremely long time for testing when the number of input correspondences becomes large. For DGR [16], we use its official implementation and the released pre-trained model. Due to the unsatisfactory results of publicly released code, we re-implement SM [38] and 3DRegNet [51], with the implementation details as follows.

Spectral matching. Traditional spectral matching [38] uses a greedy algorithm based on a one-to-one mapping constraint to discretize the leading eigenvector into the inlier/outlier labels. However, the greedy algorithm often does not show satisfactory performance in real cases. For example, if the input correspondences are pre-filtered by reciprocal check, the greedy algorithm could not reject any correspondences since all of them already satisfy the one-to-one mapping constraint. The Hungarian algorithm [49] can also be used for discretization but provides results similar to the greedy algorithm. In our work, we simply select 10% of the input correspondences with the highest confidence values as the inlier set. This approach empirically shows to be effective throughout our experiments. Then the transformation between two point clouds can be estimated using the selected correspondences.

3DRegNet. We keep the network architecture proposed in 3DRegNet [51] and train it on 3DMatch using the same settings as PointDSC. However, as observed in [16], 3DRegNet does not converge during training and the registration block cannot produce reasonable results. We speculate that directly regressing the pose results in the poor performance

due to the non-linearity of the rotation space [53, 30]. Thus we regard the output of the classification block as the inlier confidence and use the confidence as the weight for weighted least-squares fitting. We then train the network using the classification loss only, since we find the registration loss does not improve the performance. The modified 3DRegNet becomes a 3D variant of PointCN [48] and achieves reasonable results.

7.3. Time Complexity Analysis

We report the average runtime of each component in the proposed pipeline on the 3DMatch test set (roughly 5k putative correspondence per fragment) in Table 6. The reported times are measured using an Intel Xeon 8-core 2.1GHz CPU (E5-2620) and an NVIDIA GTX1080Ti GPU.

SCNonlocal	Seed Selection	NSM	Post Refine	Overall
62.0	2.0	14.4	11.1	89.5

Table 6: Runtime of each component in **milli-seconds**, averaged over 1,623 test pairs of 3DMatch. The time of hypothesis selection is included in the NSM module.

7.4. Additional Statistics

We report the area under cumulative error curve (AUC) of the rotation and translation errors defined in Eq. 11 at different thresholds, as shown in Table 7. PointDSC consistently outperforms the state-of-the-arts on both the AUC of the *Rotation Error* (RE) and *Translation Error* (TE).

	RE AUC			TE AUC					
	5°	10°	15°	5cm	10cm	15cm	20cm	25cm	30cm
SM	50.14	67.24	74.37	16.29	35.98	48.61	56.90	62.57	66.67
DGR	50.22	69.98	77.78	14.13	35.28	49.32	58.50	64.74	69.19
RANSAC	49.99	70.43	78.31	12.16	33.15	47.99	57.81	64.33	68.95
GC-RANSAC	52.81	71.56	78.90	15.33	36.77	50.94	59.95	65.94	70.19
PointDSC	**57.32**	**74.85**	**81.50**	**17.85**	**40.63**	**54.56**	**63.32**	**69.02**	**73.00**

Table 7: Registration results on 3DMatch. We calculate the exact AUC following [62]: the higher, the better. We run 100k iterations for both RANSAC and GC-RANSAC.

We also report the scene-wise registration results of our method on 3DMatch in Table 8.

	RR(%)	RE(°)	TE(cm)	IP(%)	IR(%)	F1(%)
Kitchen	98.81	1.67	5.12	80.57	88.83	84.26
Home1	97.44	1.87	6.45	83.34	88.91	85.88
Home2	82.21	3.36	7.46	71.39	80.20	74.78
Hotel1	98.67	1.88	6.04	83.96	91.48	87.38
Hotel2	92.31	1.98	5.74	81.07	86.97	83.82
Hotel3	92.59	2.00	5.87	82.65	88.57	85.03
Study	89.04	2.29	9.20	77.00	83.72	79.97
Lab	80.52	1.91	8.41	70.31	77.88	73.46

Table 8: Scene-wise statistics for PointDSC on 3DMatch.

7.5. Additional Experiments

Registration results on KITTI. Due to the space limitation and the saturated performance under the FCGF setting, we only report the registration results on KITTI under the FPFH setting in the main text. Here we report the performance of all the methods combined with FCGF in Table 9. For the learning-based models DGR and PointSM, we report the performance of the models trained from scratch (labelled "re-trained") and pre-trained on the indoor dataset 3DMatch (no extra label) with the FCGF descriptor.

	RR(↑)	RE(↓)	TE(↓)	F1(↑)	Time
SM	96.76	0.50	**19.73**	22.84	0.10
RANSAC-1k	97.12	0.48	23.37	84.26	0.22
RANSAC-10k	98.02	0.41	22.94	85.05	1.43
RANSAC-100k	**98.38**	0.38	22.60	**85.42**	13.4
DGR	95.14	0.43	23.28	73.60	0.86
PointDSC	97.84	0.33	20.99	85.29	0.31
DGR re-trained	96.90	0.33	21.29	73.56	0.86
PointDSC re-trained	98.20	**0.33**	20.94	85.37	0.31

Table 9: Registration results on KITTI under the FCGF setting. The reported time numbers do not include the construction of initial correspondences.

Under low-overlapping cases. Recently, Huang et. al [32] have constructed a low-overlapping dataset 3DLoMatch from the 3DMatch benchmark to evaluate the point cloud registration algorithms under low-overlapping scenarios. To demonstrate the robustness of our PointDSC, we further evaluate our method on the 3DLoMatch dataset and report the results[1] in Table 10. Note that we directly use the model trained on 3DMatch without fine-tuning and keep 5cm voxel for the FCGF descriptor. All the other settings are the same as [32] for a fair comparison.

	5000	2500	1000	500	250	Δ
FCGF[18] + RANSAC	35.7	34.9	33.4	31.3	24.4	-
FCGF[18] + PointDSC	52.0	51.0	45.2	37.7	27.5	+10.74
Predator[32] + RANSAC	54.2	55.8	56.7	**56.1**	**50.7**	-
Predator[32] + PointDSC	**61.5**	**60.2**	**58.5**	55.4	50.4	+2.50

Table 10: Registration recall on the 3DLoMatch dataset using different numbers of points to construct the input correspondence set. The last column is the average increase brought by PointDSC.

As shown in Table 10, our method consistently outperforms RANSAC when combined with different descriptors. Moreover, our method can further boost the performance of Predator [32], a recently proposed learning-based descriptors especially designed for low-overlapping registration, showing the effectiveness and robustness of our method under high outlier ratios. PointDSC increases the registration recall by **16.3%** and **7.3%** under 5000 points setting for FCGF and Predator, respectively. Note that PointDSC does

[1]The computation of registration recall is slightly different with ours, we refer readers to [32] for more details.

not bring much performance gain when only a small number of points (e.g. less than 500) are used to construct the input correspondences mainly because some of the point cloud pairs have too few (e.g. less than 3) correspondences to identify a unique registration.

Prioritized RANSAC. Despite the common usage of the inlier probability predicted by networks in weighted least-squares fitting [16, 27], little attention has been drawn to leverage the predicted probability in a RANSAC framework. In this experiment, we derive a strong RANSAC variant (denoted as *Prioritized*) by using the inlier probability for selecting seeds to bias the sampling distribution. For a fair comparison, we implement *Prioritized* using the same codebase (`Open3D`) as RANSAC. As shown in Table 11, *Prioritized* outperforms classic RANSAC by more than 30% in terms of registration recall, indicating that the inlier probability predicted by our method is meaningful and accurate, and thus could help RANSAC to sample all-inlier subsets earlier and to achieve better performance in fewer iterations. This RANSAC variant can also be used for each correspondence subset to replace the weighted LS in Eq. 5, denoted as *Local Prioritized* in Table 11. Still, PointDSC outperforms the strong baselines with better accuracy and faster speed.

	RR(%)	RE(°)	TE(cm)	F1(%)	Time(s)
RANSAC-1k	40.05	5.16	13.65	39.23	0.08
Prioritized-1k	74.31	2.83	8.26	67.58	0.13
Local Prioritized	78.00	2.08	**6.42**	69.44	0.24
PointDSC	**78.50**	**2.07**	6.57	**69.85**	0.09

Table 11: Results on 3DMatch test set using FPFH.

Ablation on loss function. The L_{sm} is proposed to provide additional supervision, i.e., the pairwise relations between correspondences, serving as a complement to the node-wise supervision. The edge-wise supervision encourages the inliers to be concentrated in the embedding space, and this is the key assumption of our NSM module. To demonstrate its effectiveness, we compare the model trained with Eq. 10 and the model trained without the proposed spectral matching loss L_{sm} (Eq. 9) in Table 5. As shown in Table 12, L_{sm} improves the registration recall by 0.67% over the strong baseline.

	RR(↑)	RE(↓)	TE(↓)	F1(↑)	Time
PointDSC	**93.28**	**2.06**	**6.55**	**82.35**	0.09
w/o L_{sm}	92.61	2.07	6.75	81.58	0.09

Table 12: Ablation experiments of NSM module.

Effect of neighborhood size k. The size of correspondence subset, k, (Sec. 3.4) is a key parameter of our proposed method, and controls the size of each correspondence subset for neural spectral matching. We test the performance of our method with k being 10, 20, 30, 40, 50, 60, 100, and 200, respectively. As shown in Table 13, the results show that our method is robust to the choice of k. We ascribe the robustness to the neural spectral matching module, which effectively prunes the potential outliers in the retrieved subsets, thus producing a correct model even when starting from a not-all-inlier sample. We finally choose $k = 40$ for its best *Registration Recall* and modest computation cost.

	RR(↑)	RE(↓)	TE(↓)	IP(↑)	IR(↑)	F1(↑)
10	92.73	2.04	**6.44**	79.01	85.51	81.87
20	92.79	2.04	6.50	78.88	85.86	81.96
30	93.10	2.04	6.50	79.07	86.35	82.25
40	**93.28**	2.06	6.55	79.10	86.54	82.35
50	93.10	2.05	6.54	79.10	86.47	82.34
60	92.91	2.04	6.51	**79.14**	**86.61**	**82.42**
100	92.91	2.04	6.53	78.87	86.25	82.12
200	92.79	2.04	6.51	78.96	86.37	82.22

Table 13: Performance of our PointDSC when varying the size of correspondence subsets in the NSM module.

Joint training with descriptor and detector. In this part, we explore the potential of jointly optimizing the local feature learning and outlier rejection stages. A recently proposed method D3Feat [4], which efficiently performs dense feature detection and description by a single network, best suits our need. By back-propagating gradients to the input descriptors, the detector network can also be updated. Thus we build an end-to-end registration pipeline by taking the output of D3Feat as the input to our outlier rejection algorithm. We establish the correspondences using soft nearest neighbor search proposed in [27] to make the whole pipeline differentiable. We first train the feature network and the outlier rejection network separately, and then finetune them together using the losses in [4] and Eq. 10.

However, we did not observe performance improvement for the feature network in this preliminary joint training experiment. We suspect that the current losses are unable to provide meaningful gradients to the feature network. We believe that it is an interesting future direction to design proper loss formulations for end-to-end learning of both feature and outlier rejection networks.

Nevertheless, it is noteworthy that within a reasonable range, D3Feat + PointDSC achieves improved results when using fewer but more confident keypoints to build the input putative correspondences for outlier rejection. We ascribe the performance improvement to the elimination of keypoints in non-salient regions like smooth surface regions, reducing the failure registration caused by large symmetric objects in the scene. (See the visualization of failure cases Fig. 11 for more detail.) The results of D3Feat + PointDSC under different numbers of keypoints (labelled by **Joint(#num)**) are provided in Table 14 for comparisons.

7.6. Derivation of Eq. 5

For completeness, we summarize the closed-form solution of the weighted least-squares pairwise registration

	RR(↑)	RE(↓)	TE(↓)	IP(↑)	IR(↑)	F1(↑)
PointDSC	93.28	2.06	6.55	79.10	**86.54**	82.35
Joint (5000)	92.42	**1.83**	**5.87**	79.02	85.14	81.72
Joint (4000)	92.67	1.86	5.88	79.67	85.54	82.26
Joint (3000)	93.35	1.85	5.92	80.78	86.26	83.19
Joint (2500)	**93.59**	1.86	6.00	81.05	86.40	**83.38**
Joint (2000)	93.53	1.85	6.02	**81.14**	86.11	83.30
Joint (1000)	90.82	1.96	6.38	78.75	83.41	80.64

Table 14: Registration results of joint training with descriptor and detector on 3DMatch.

problem [8],

$$\hat{\mathbf{R}}, \hat{\mathbf{t}} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i}^{N} e_i \left\| \mathbf{R}\boldsymbol{x_i} + \mathbf{t} - \boldsymbol{y_i} \right\|^2, \tag{13}$$

where $(\boldsymbol{x_i}, \boldsymbol{y_i})$ is a pair of corresponding points, with $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ being from point clouds $\mathbf{X} \in \mathbb{R}^{N\times 3}$ and $\mathbf{Y} \in \mathbb{R}^{N\times 3}$, respectively. Let $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ denote the weighted centroids of $\mathbf{X}$ and $\mathbf{Y}$, respectively:

$$\bar{\boldsymbol{x}} = \frac{\sum_i^N e_i \boldsymbol{x_i}}{\sum_i^N e_i}, \quad \bar{\boldsymbol{y}} = \frac{\sum_i^N e_i \boldsymbol{y_i}}{\sum_i^N e_i}. \tag{14}$$

We first convert the original coordinates to the centered coordinates by subtracting the corresponding centroids,

$$\tilde{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{x_i} - \bar{\boldsymbol{x}}, \;\; \tilde{\boldsymbol{y}}_{\boldsymbol{i}} = \boldsymbol{y_i} - \bar{\boldsymbol{y}}, \quad i = 1, ..., N. \tag{15}$$

The next step involves the calculation of the weighted covariance matrix $\mathbf{H}$,

$$\mathbf{H} = \tilde{\mathbf{X}}^{\mathbf{T}} \mathbf{E} \tilde{\mathbf{Y}}, \tag{16}$$

where $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are the matrix forms of the centered coordinates and $\mathbf{E} = \text{diag}(e_1, e_2, ..., e_N)$. Then the rotation matrix from $\mathbf{X}$ to $\mathbf{Y}$ can be found by singular value decomposition (SVD),

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{SVD}(\mathbf{H}), \tag{17}$$

$$\hat{\mathbf{R}} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{V}\mathbf{U}^{\mathbf{T}}) \end{bmatrix} \mathbf{U}^{\mathbf{T}}, \tag{18}$$

where $\det(\cdot)$ denotes the determinant, which is used to avoid the reflection cases. Finally, the translation between the two point clouds is computed as,

$$\hat{\mathbf{t}} = \bar{\boldsymbol{y}} - \hat{\mathbf{R}}\bar{\boldsymbol{x}}. \tag{19}$$

7.7. Qualitative Results

We show the outlier rejection results on 3DMatch and KITTI in Fig. 9 and Fig. 10, respectively. For the KITTI dataset, we use the FPFH descriptor to better demonstrate the superiority of our method. RANSAC suffers from significant performance degradation because the FPFH descriptor results in large outlier ratios, where it is harder to sample an outlier-free set. In contrast, our PointDSC still gives satisfactory results.

We also provide the visualization of failure cases of our method on 3DMatch in Fig. 11. One common failure case happens when there are large symmetry objects (e.g., the wall, floor) in a scene, resulting in rotation errors around 90°or 180°. In this case, the clusters formed by outlier correspondences could become dominant, leading to incorrect transformation hypotheses. Then an incorrect transformation is probably selected as the final solution since a large number of outlier correspondences would satisfy this transformation. To highlight this issue, we draw the distribution of rotation errors of unsuccessful registration pairs on the 3DMatch test set in Fig. 8, from which we can find that a large portion of pairs has around 90°and 180°.

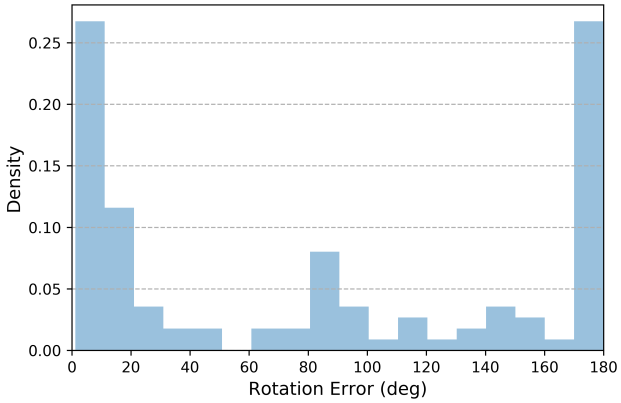

Figure 8: Rotation errors of unsuccessful registration pairs of PointDSC on the 3DMatch test set.

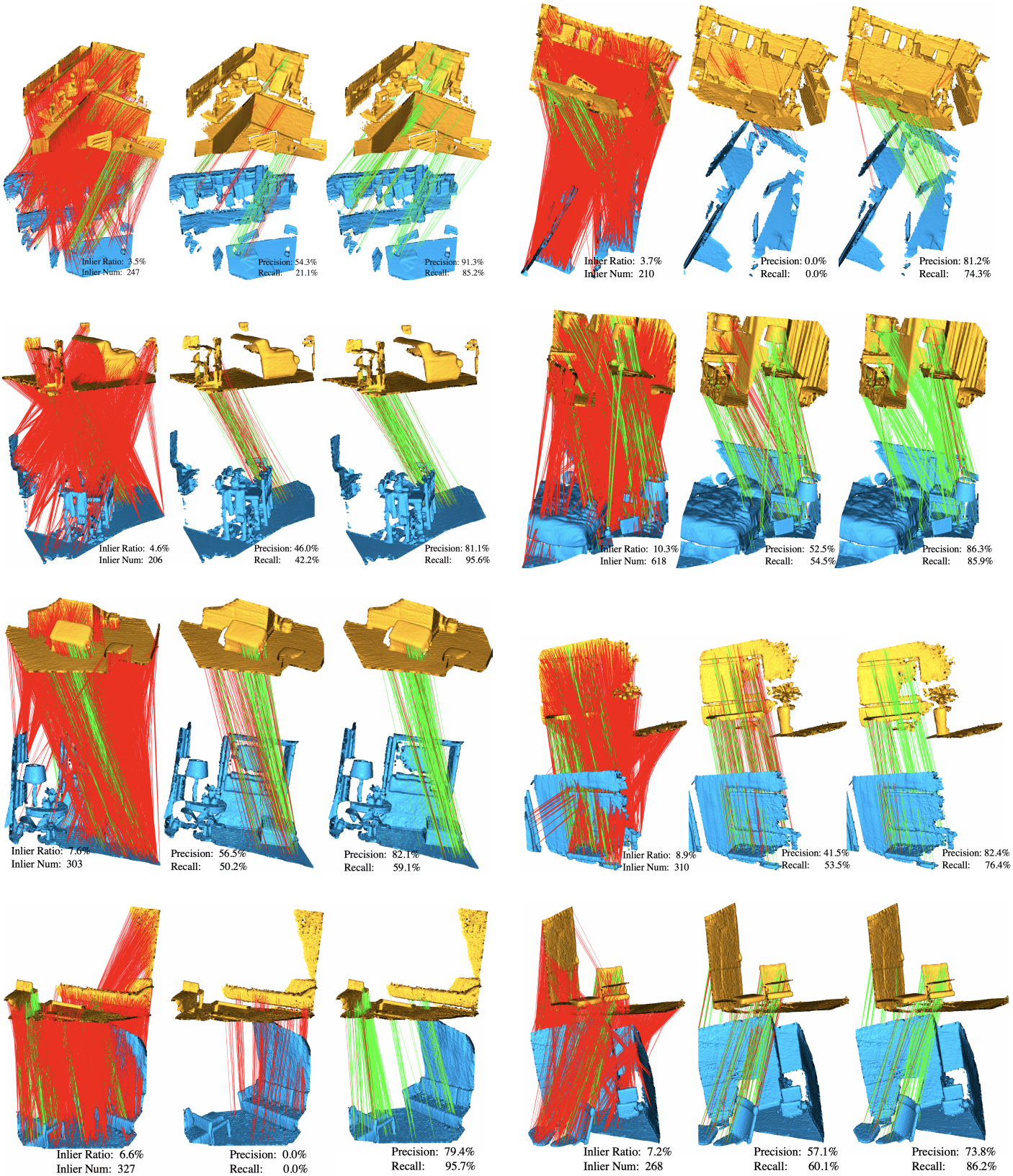

Figure 9: Visualization of outlier rejection results on the 3DMatch dataset. From left to right: input correspondences constructed by FCGF, results of RANSAC-100k, and results of PointDSC. Best viewed with color and zoom-in.

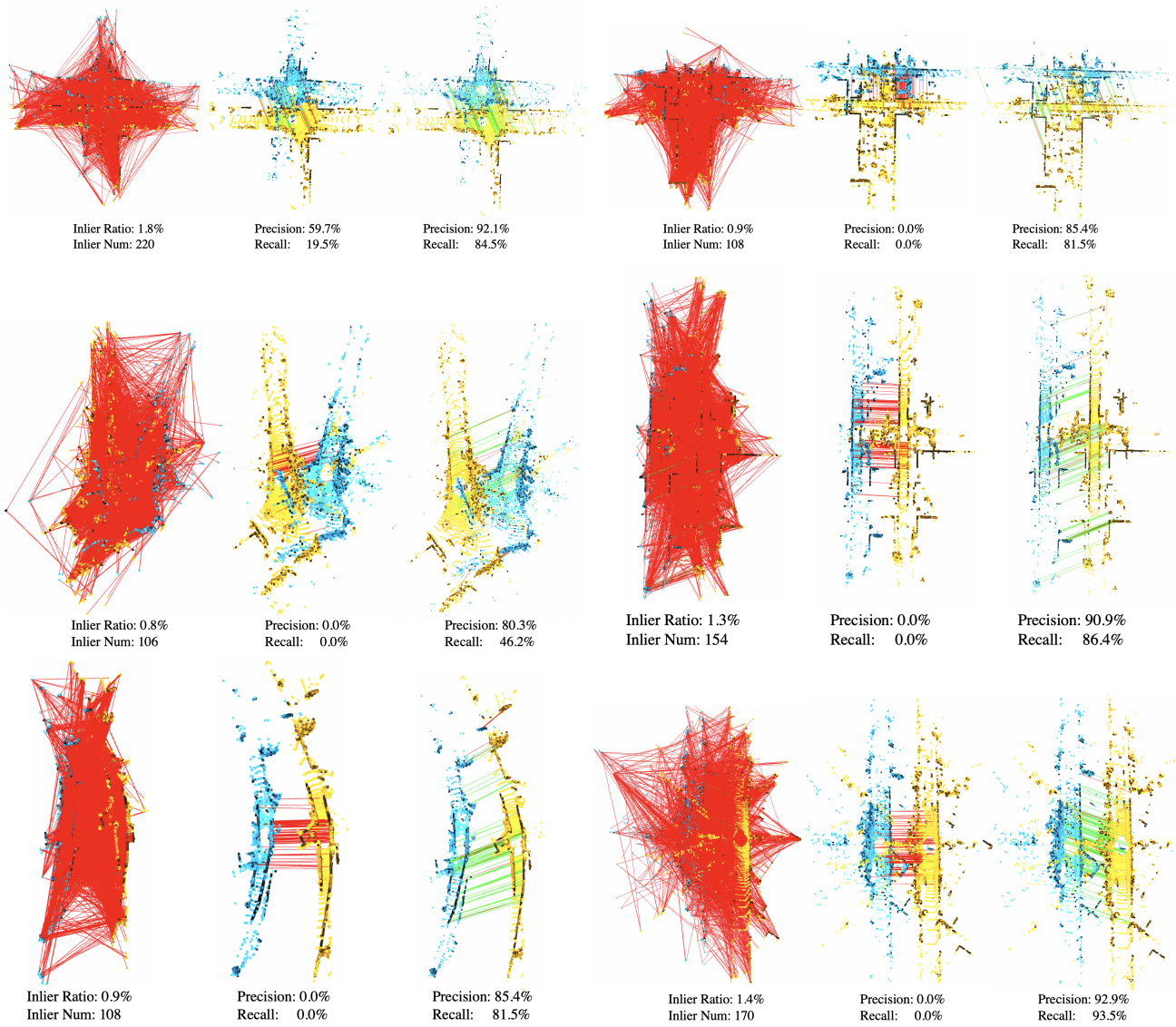

Figure 10: Visualization of outlier rejection results on the KITTI dataset. From left to right: input correspondences constructed by FPFH (we choose FPFH to better demonstrate the robustness of our method to high outlier ratios), results of RANSAC-100k, and results of PointDSC. Best viewed with color and zoom-in.

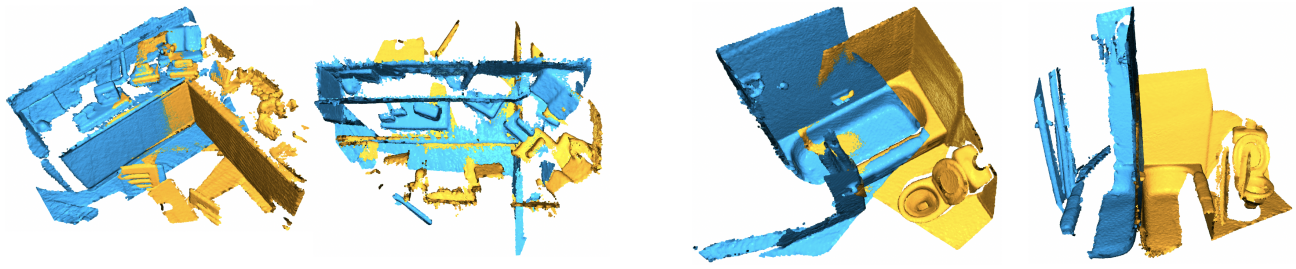

Figure 11: Two representative failure examples of our method on 3DMatch. In each example, ground-truth registration (Left) and estimated registration (Right). We observe that our method fails mainly due to the symmetries in the scene.