Correspondence-Free Fast and Robust Spherical Point Pattern Registration

Supplementary Material

7. Mathematical Preliminaries

This section contains additional mathematical background needed for the algorithms in our paper.

7.1. Rotations

The set of all rotations in three dimensions is denoted as SO(3), known as the "special orthogonal group." Rotations can be represented by 3×3 matrices that preserve both distance (i.e., $\|\mathbf{Rx}\| = \|\mathbf{x}\|$) and orientation (i.e., $\det(\mathbf{R}) = +1$). If we represent points on the sphere as 3D unit vectors $\mathbf{x}$, a rotation can be applied using the matrix-vector product $\mathbf{Rx}$ [20].

7.2. Rotation Between Two Vectors

To align one unit vector $\mathbf{v_2}$ with another unit vector $\mathbf{v_1}$, we can use the Rodrigues rotation matrix formula:

$$\mathbf{R} = \begin{cases} \mathbf{I}, & \text{if } \hat{\mathbf{v}}_\mathbf{1} = \hat{\mathbf{v}}_\mathbf{2} \\ -\mathbf{I} + 2\mathbf{v}\mathbf{v}^T, & \text{if } \hat{\mathbf{v}}_\mathbf{1} = -\hat{\mathbf{v}}_\mathbf{2} \\ \mathbf{I} + \mathbf{V}\sin\theta + \mathbf{V}^2(1-\cos\theta), & \text{otherwise} \end{cases} \quad (6)$$

where:

$$\hat{\mathbf{v}}_\mathbf{1} = \frac{\mathbf{v_1}}{\|\mathbf{v_1}\|}, \quad \hat{\mathbf{v}}_\mathbf{2} = \frac{\mathbf{v_2}}{\|\mathbf{v_2}\|}, \mathbf{v} = \frac{\hat{\mathbf{v}}_\mathbf{2} \times \hat{\mathbf{v}}_\mathbf{1}}{\|\hat{\mathbf{v}}_\mathbf{2} \times \hat{\mathbf{v}}_\mathbf{1}\|}.$$

$$\theta = \arccos(\hat{\mathbf{v}}_\mathbf{2} \cdot \hat{\mathbf{v}}_\mathbf{1}), \mathbf{V} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}.$$

7.3. Axis Direction Angles

The axis direction angle of an S2-point around an axis is defined as the angle between the point's projection onto the plane perpendicular to that axis and a reference axis. For example, in Fig. 10, the S2-point P is projected onto the perpendicular planes, and the axis direction angles are shown accordingly. Mathematically, these angles can be computed using the *atan2()* function, as shown in Eq. (7). We set the range of the angle to lie between $0°$ and $360°$.

$$\left.\begin{aligned} \theta_z &= \text{atan2}(y, x) \\ \theta_y &= \text{atan2}(x, z) \\ \theta_x &= \text{atan2}(z, y) \end{aligned}\right\} \quad (7)$$

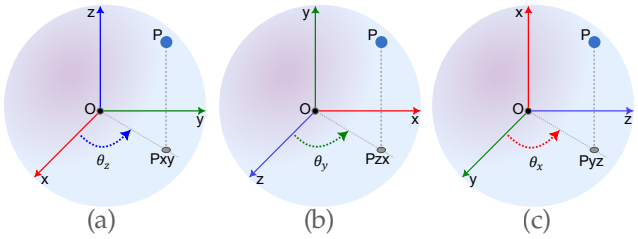

Figure 10. Spherical Point (P) to Axis Direction Angles. Projections of point P onto the xy, zx, and yz planes are denoted as P_{xy}, P_{zx}, and P_{yz}, respectively. (a) θ_z is the angle around the z-axis from the x-direction to the projection vector $\overrightarrow{OP_{xy}}$. (b) θ_y is the angle around the y-axis from the z-direction to the projection vector $\overrightarrow{OP_{zx}}$. (c) θ_x is the angle around the x-axis from the y-direction to the projection vector $\overrightarrow{OP_{yz}}$.

7.4. Rotation Matrix from Axis-Angles

Given rotation angles θ_x, θ_y, and θ_z around the X-, Y-, and Z-axes, respectively, we can compute the individual rotation matrices as follows:

$$R_z(\theta_z) = \begin{bmatrix} c_{\theta_z} & -s_{\theta_z} & 0 \\ s_{\theta_z} & c_{\theta_z} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_y(\theta_y) = \begin{bmatrix} c_{\theta_y} & 0 & s_{\theta_y} \\ 0 & 1 & 0 \\ -s_{\theta_y} & 0 & c_{\theta_y} \end{bmatrix},$$
$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{\theta_x} & -s_{\theta_x} \\ 0 & s_{\theta_x} & c_{\theta_x} \end{bmatrix}, \quad [\text{here } c\colon \cos(), s\colon \sin()] \quad (8)$$

The combined rotation matrix R is then obtained by sequentially applying the rotations around each axis in the ZYX order:

$$R = R_z(\theta_z) \cdot R_y(\theta_y) \cdot R_x(\theta_x) \quad (9)$$

8. Formal Justification of SPMC:

Centroid-based alignment is a classical strategy for estimating temporal shifts in noisy 1D signals, with theoretical performance bounds established in [36]. Our method is analogous to this idea in the spherical domain, where the azimuthal shift—after mean alignment and projection—corresponds to a 1D phase shift. In the absence of noise and outliers, aligning the mean directions of two spherical point sets ensures that the residual difference lies entirely in the azimuthal plane. A formal proof under this condition is below.

Exact Recovery of SPMC (no noise or outliers)

Statement. Let $\mathcal{B} = R_{\text{true}}\mathcal{A}$, where $\mathcal{A}, \mathcal{B} \in \mathbb{S}^2$ are identical spherical point clouds and $R_{\text{true}} \in \text{SO}(3)$. Assume: (i) no noise or outliers, Then SPMC exactly recovers:

$$R_{\text{opt}} = R_{\text{true}}.$$

Proof. Let $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}} = R_{\text{true}}\bar{\mathbf{a}}$ be the mean directions of $\mathcal{A}$ and $\mathcal{B}$. Let $R_A, R_B \in \text{SO}(3)$ be rotations such that $R_A\bar{\mathbf{a}} = R_B\bar{\mathbf{b}} = [0,0,1]^T$. Then:

$$\mathcal{A}^{\text{NP}} = R_A\mathcal{A}, \quad \mathcal{B}^{\text{NP}} = R_B R_{\text{true}}\mathcal{A} = R_{\text{res}}\mathcal{A}^{\text{NP}},$$

where $R_{\text{res}} = R_B R_{\text{true}} R_A^{-1}$ fixes the north pole and is therefore a z-axis rotation: $R_{\text{res}} = R_z(\theta)$. Azimuthal histogram correlation over $\mathcal{A}^{\text{NP}}$ and $\mathcal{B}^{\text{NP}}$ recovers $s^* = \theta$. The final estimate is:

$$R_{\text{opt}} = R_A^{-1} R_z(s^*) R_B = R_{\text{true}}. \quad \blacksquare$$

9. Experiment 1: Robust Alignment

9.1. Dataset Description

In the first experiment, we evaluate the robustness, accuracy, and time complexity of our algorithm using the "Robust Vector Alignment Dataset." This dataset contains five template spherical patterns, labeled A_1, A_2, A_3, A_4, and A_5. The patterns are simulated to represent various distributions: Pattern 1 (22212 points) represents a localized simple trajectory, while Pattern 2 (9220 points) represents a more complex trajectory, resembling sharp features observed from objects like drones or other directional vector observations. Patterns 3 (28218 points), 4 (26527 points), and 5 (28557 points) represent spherical distributions covering the entire sphere, capturing characteristics such as random small islands (Pattern 3), large islands (Pattern 4), and non-uniform densities (Pattern 5).

To create the source sets, we add noise and outliers across seven stages (B_1-B_7). In Stage 1, there is no noise and are no outliers, meaning the source and template sets contain the same number of points. In Stage 2, Gaussian noise with a standard deviation of 0.01 is added, without any outliers. In Stage 3, we retain the 0.01 Gaussian noise and introduce random outliers by replacing 10% of the template points. In Stages 4, 5, 6, and 7, we progressively increase the proportion of template points replaced by random outliers to 25%, 50%, 75%, and 90%, respectively, while maintaining the same noise level as in Stages 1 and 2.

Unlike previous approaches that select a few specific rotations [47], the complete set of source configurations in our experiments is generated by sampling 100 random rotations (R100) over the $SO(3)$ space (shown in the bottom right of Fig. 4 in the main text) and applying each rotation to the source pattern. Thus, for each template pattern $(A_1, \ldots, A_5)$, we create a total of $7 \times 100 = 700$ source patterns. A sample configuration, such as $A1B7R95$, indicates that the template pattern is A_1, the source pattern is B_7 (which includes 90% outliers and Gaussian noise with a standard deviation of 0.01), and the rotation applied is labeled as rotation number 95. While Fig. 4 in the main text displays the quantitative results of the simulation experiments, Fig. 11 shows qualitative results of the SPMC+FRS algorithm on several example configurations.

As discussed in Sec. 4 in the main text, our problem is directly related to the "Robust Wahba Problem" [73]. In their work, Heng Yang et al. developed QUASAR, which achieved state-of-the-art results for the Wahba problem, solving it under extreme outlier conditions (up to 95% outliers) when putative correspondences are provided. In contrast, our problem assumes no correspondences are given. Therefore, in our first experiments, we compare the performance of QUASAR with our algorithms. Since QUASAR requires putative correspondences, we use FPFH [60] as a prior step before applying QUASAR.

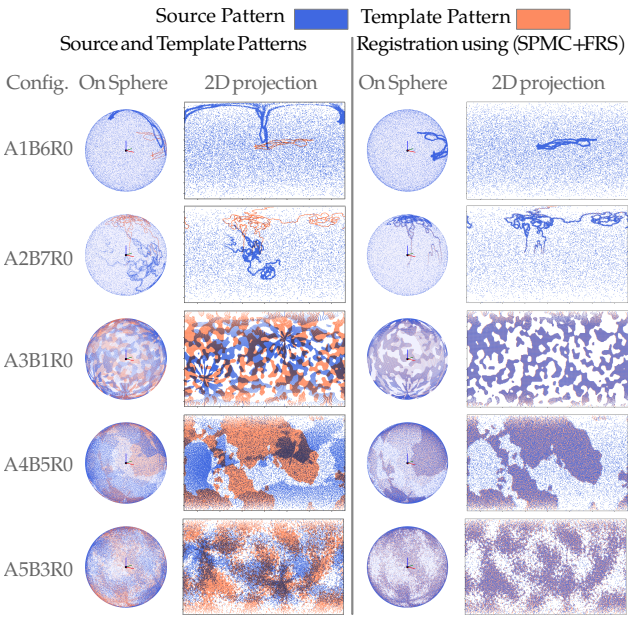

Figure 11. Qualitative results of the SPMC+FRS algorithm on "Robust Vector Alignment Dataset." On the left, the 3D spherical point cloud and 2D projection of the template pattern for each of the five datasets are shown, along with one configuration of the source pattern. On the right, the pattern is displayed after registration. The results demonstrate that the source pattern has successfully aligned with the template pattern. For configuration number, A_i denotes the template pattern number, B_i denotes the source set configuration, and R_i represents the rotation number.

9.2. Iteration Analysis:

In Fig. 12, we present the iteration analysis for the FRS algorithm using the "Robust Vector Alignment Dataset." Across all datasets, including the subsets detailed in Sec. 5.1 in the main text, the algorithm converges in slightly more than 10 iterations on average.

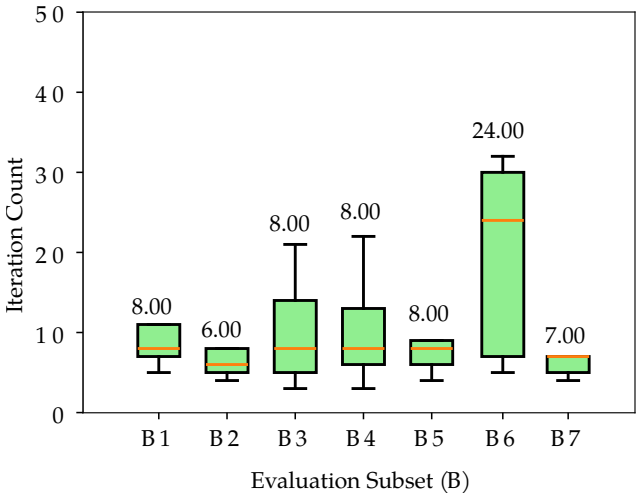

Figure 12. Iteration analysis for the FRS algorithm

10. Experiment 2: Point Cloud Registration

10.1. Dataset Preparation

From the ModelNet40 dataset [72], we selected 166 objects from 35 object classes. We chose to exclude classes containing highly symmetric objects, such as bowls, cones, vases, bottles, and glass boxes. These objects exhibit symmetry across multiple directions, leading to rotational ambiguity. Even if the algorithm registers these objects accurately, rotational errors may still occur due to their indistinguishable orientations.

Complete-to-Complete (Comp2Comp) Dataset Using the 3D CAD models, we first scaled each model to fit within a unit cube of length 1, then sampled 5000 points per object. We divided the source and target shapes based on two primary criteria: **a. No Correspondence** (No corr.): We selected 2500 points for the source and 2500 points for the target, leaving no one-to-one correspondence. **b. 10% Correspondence** (10% corr.): In this case, we split the dataset so that the source and target share 10% of the points with absolute correspondence. We also created two additional cases by adding Gaussian noise with a standard deviation of 0.01 to each configuration. After creating the basic dataset, we applied 10 random rotations and a fixed translation of $[0.1, 0.2, 0.3]$ to simulate the source.

Partial-to-Complete (Par2Comp) Dataset For the Par2Comp dataset, we used the same source dataset but selected 8 equidistant viewpoints across the $SO(3)$ space to cover the entire sphere. We discarded partial views with ambiguous or highly symmetric content (e.g., one view showing only a portion of an airplane wing where both wings are symmetric). Out of 166 objects, we generated a total of 881 partial views.

10.2. Centroid Aware Spherical Embedding (CASE)

Different spherical projection techniques are discussed in the literature, such as ray tracing from a 3D object from a viewpoint [20] and spherical remeshing [58]. Centroid-Aware Spherical Embedding (CASE) represents the spherical projection of a point cloud relative to its centroid. This embedding is rotation- and scale-invariant and ensures that all points reside on the unit sphere, making it effective for tasks such as point cloud registration.

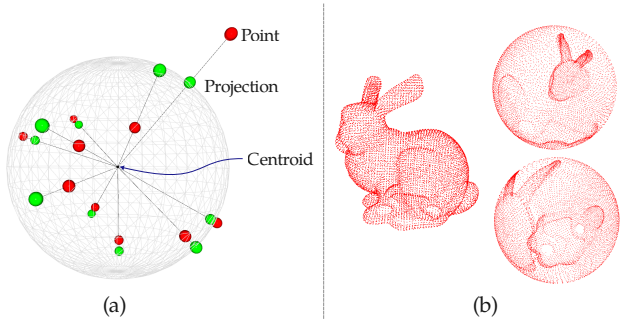

Figure 13. CASE Representation of Point Cloud. (a) Point cloud (red) projected onto the unit sphere (green) from the centroid. (b) CASE representation of the bunny point cloud from two different viewpoints.

Point Cloud to CASE: Given a point cloud, let (x_c, y_c, z_c) denote its centroid, and consider a 3D point (x_1, y_1, z_1) in the cloud. The projection of this point onto the unit sphere is computed as follows:

1. Compute the vector from the centroid to the point:

$$\vec{v} = \begin{bmatrix} x_1 - x_c \\ y_1 - y_c \\ z_1 - z_c \end{bmatrix}$$

2. Normalize this vector to have a unit norm:

$$\vec{v}_{\text{unit}} = \frac{\vec{v}}{\|\vec{v}\|} = \frac{\begin{bmatrix} x_1 - x_c \\ y_1 - y_c \\ z_1 - z_c \end{bmatrix}}{\sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2 + (z_1 - z_c)^2}}$$

3. The coordinates of the point projected onto the unit sphere are:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \vec{v}_{\text{unit}}$$

Fig. 13(a) illustrates a sample point cloud (red) and its projection (green) onto the unit sphere from the centroid. Fig. 13(b) depicts the bunny point cloud, highlighting its original structure alongside its CASE representation from two different viewpoints. Additionally, Fig. 14 demonstrates the robustness of CASE, along with a qualitative evaluation showcasing its invariance to rotation and translation, assuming the centroid is known.

10.3. Translation Estimation Algorithm

The translation estimation proceeds as follows:

- **Voxel Assignment and Weighting:** Each point in the source and target clouds is assigned to a voxel grid, with voxel size set to 0.2 units. Points are mapped to voxel indices (i, j, k) by applying $\mathit{floor}(\mathbf{p}/\text{voxel\_size})$ for each coordinate $\mathbf{p}$. A weight is assigned to each voxel, proportional to the number of points it contains, yielding voxel sets $\mathcal{V}_{\text{source}}$ and $\mathcal{V}_{\text{target}}$ with respective weights.
- **Translation Vector Voting:** For each voxel pair $(\mathbf{v}_i \in \mathcal{V}_{\text{source}}, \mathbf{u}_j \in \mathcal{V}_{\text{target}})$, a candidate translation vector $\mathbf{t}_{ij} = (\mathbf{u}_j - \mathbf{v}_i) \times \text{voxel\_size}$ is computed. The vote for each candidate translation $\mathbf{t}_{ij}$ is weighted by the product of the source and target voxel weights. The translation with the highest accumulated weight across all candidate translations is selected as the coarse translation estimate.
- **Fine Alignment with ICP:** After coarse alignment, the partial source and target clouds will be roughly aligned. For N points in the partial source cloud, we select N nearest neighbors from the target cloud and perform translation-only ICP. This refinement iteratively adjusts $\mathbf{t}_{\text{final}}$ by minimizing residual distances between corresponding points in the aligned source and target clouds.

10.4. Qualitative Results of Point Cloud Registration

Fig. 15 shows qualitative results of our point cloud registration algorithms compared to other works.

10.5. Point Cloud Registration on Real-World Datasets

In Fig. 16, we evaluate our algorithm on the 3DMatch and KITTI datasets. For 3DMatch, we select scans with an overlap of at least 65%. Our method is robust when the point clouds have more than 65% overlap, as is the case for nearly all subsequent scans in KITTI and for many scan pairs in 3DMatch. We exclude cases with less than 65% overlap, as the method becomes unreliable in such scenarios. A failure case is detailed in Sec. 10.6.

We summarize our results to those of other algorithms (EGST [78] and GeDi [57]) in Table 1. EGST [78] appears to be the current state-of-the-art algorithm for point cloud registration, using learned geometric structure descriptors. GeDi is a recent machine learning method that also uses learned 3D descriptors and was recently state-of-the-art. The results from these other algorithms were taken from their respective papers.

Even though our algorithms were not originally intended for point cloud registration, our CASE method with Complete-to-Complete point clouds in Modelnet40 gives a better result than EGST for the "No correlation and No Noise" and "10% correlation and No Noise" cases. Our EGI method gives comparable results to EGST on Modelnet40 when there is Gaussian noise and for partial point clouds. However, their algorithm is much more accurate on the KITTI and 3DMatch datasets. The GeDi algorithm is comparable to ours on KITTI and 3DMatch.

The EGST algorithm paper quotes an execution time of 0.012 s on the partial overlap data, as compared to 0.05 s for ours (although they used a faster computer). The GeDi algorithm is much slower than ours, since their algorithm is slightly faster than FPFH, and ours is roughly 20x faster than FPFH.

10.6. Failure Case

Fig. 17 shows an example of how our algorithm can fail with point cloud registration if there is insufficient overlap between the two point clouds. In the example, there is only ˜20% overlap; the algorithm works reliably if there is >65% overlap.

11. Experiment 3: Rotation Estimation From Spherical Images

While converting spherical images to spherical points (see Sec. 5.3 in the main text), the choice of threshold value is crucial. A lower threshold captures more features but includes more clutter, while a higher threshold captures fewer features with reduced clutter. Fig. 18 illustrates the effects of varying threshold values.

11.1. Qualitative Results

Fig. 19 shows all source spherical images and their 2D projections alongside the template spherical image and its 2D projection for a sample rotation. The results after alignment using our algorithm are also displayed, including a pixel-wise difference map and a binary thresholded map (clutter map) for each configuration.

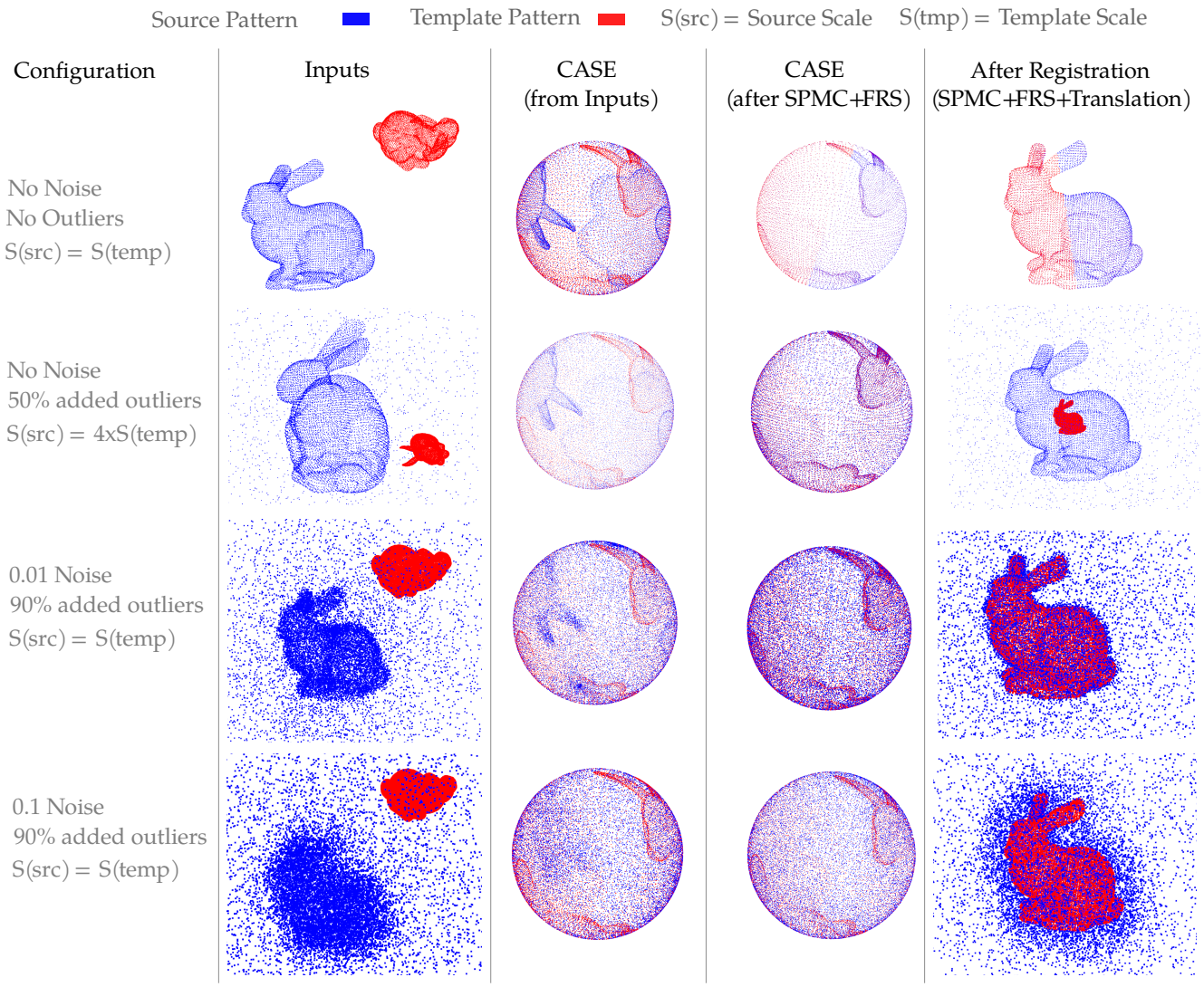

Figure 14. Robustness and Invariance of CASE to Rotation, Translation, and Scale. The first column depicts the input point clouds, where the source set is first scaled to fit into a unit cube. Isotropic Gaussian noise is added to the source set with a mean of 0 and a standard deviation of 0.01 (third row) or 0.1 (fourth row). The second column visualizes the CASE directly derived from the input. The third column shows the CASE after applying SPMC+FRS, and the final column presents the registration results in $\mathbb{R}^3$ space. The top row illustrates a simple one-to-one case. The second row demonstrates a source set with 50% outliers and a scale four times that of the template. The third row depicts a source set with the same scale as the template but includes 0.01 Gaussian noise and 90% added outliers. The bottom row shows a source set with the same scale as the template, 0.1 Gaussian noise, and 90% added outliers.

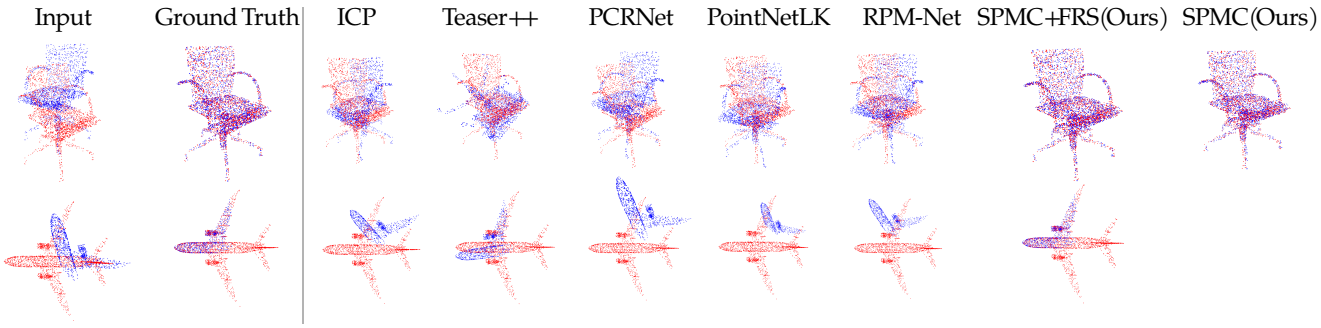

Figure 15. Qualitative results of point cloud registration for complete-to-complete (first row) and partial-to-complete (second row) cases with different methods. These are: ICP [59], Teaser++ [75], PCRNet [62], PointNetLK [2], and RPM-Net [77]. For complete-to-complete registration, we used two spherical embeddings: CASE in the SPMC method, and EGI in the SPMC+FRS method. For partial-to-complete registration, only the EGI embedding was used in the SPMC+FRS method.

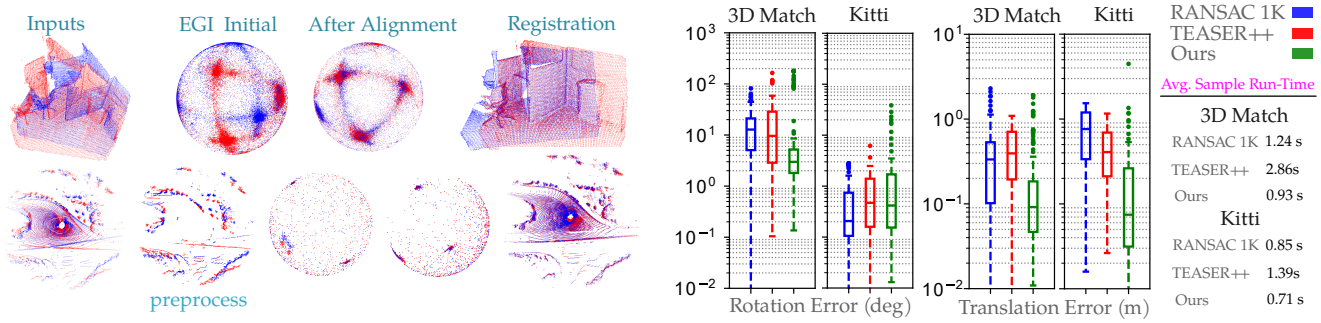

Figure 16. Qualitative (left) and quantitative (right) results on the 3DMatch and KITTI datasets. Point clouds were voxel-downsampled with a voxel size of 0.03. For KITTI, an additional preprocessing step was performed to remove ground reflection points by simply removing the points with very low elevation.

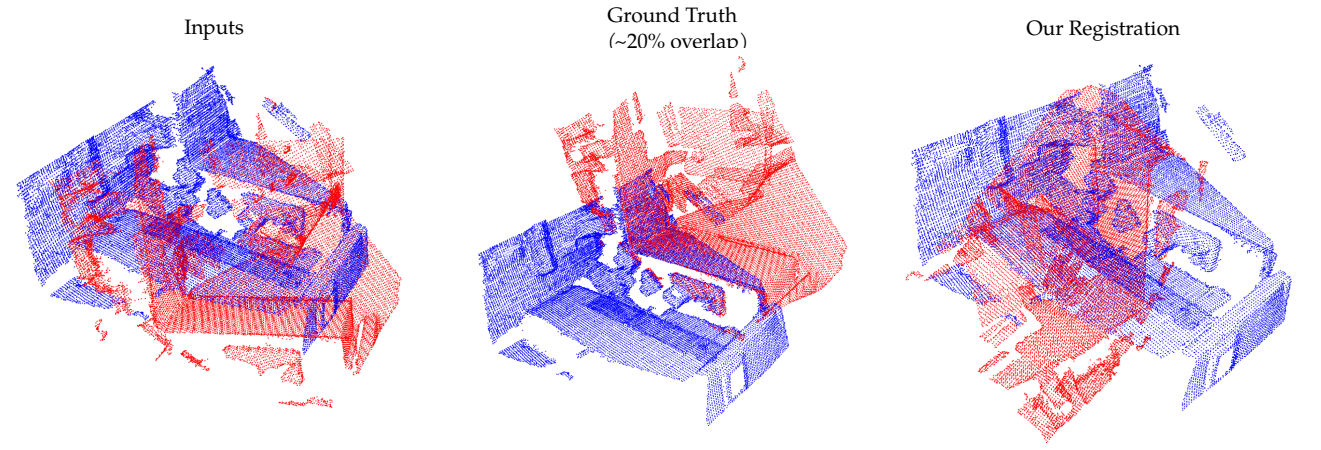

Figure 17. Example of point cloud registration where our algorithm fails. In this example, there is only around 20% overlap between the two input point clouds, leading the algorithm to not work correctly.

Table 1. Rotation ($E(R)$, [°]) and Translation ($E(t)$, [cm]) Errors across different datasets and algorithms.

Dataset/Scenario	**Method**	$E(R)$ [°]	$E(t)$ [cm]
Modelnet40: Complete-Complete			
Unseen Objects	EGST HA [78]	0.1803°	0.15
Unseen Categories	EGST HA [78]	0.4554°	0.32
Gaussian Noise	EGST HA [78]	1.1687°	0.96
Comp2Comp: No corr., No Noise	Ours - CASE	0.029°	0.13
Comp2Comp: No corr., 0.01 Noise	Ours - CASE	1.1°	0.8
Comp2Comp: 10% corr., No Noise	Ours - CASE	0.001°	0.1
Comp2Comp: 10% corr., 0.01 Noise	Ours - CASE	1.7°	0.8
Comp2Comp: No corr., No Noise	Ours - EGI	1.3°	0.8
Comp2Comp: No corr., 0.01 Noise	Ours - EGI	1.7°	0.8
Comp2Comp: 10% corr., No Noise	Ours - EGI	1.6°	0.8
Comp2Comp: 10% corr., 0.01 Noise	Ours - EGI	1.7°	0.8
Modelnet40: Partial-Partial			
Partial-Partial	EGST HA [78]	3.3040°	4.92
Part2Comp: No corr., No Noise	Ours - EGI	3.2°	3.9
Part2Comp: No corr., 0.01 Noise	Ours - EGI	3.0°	4.2
Part2Comp: 10% corr., No Noise	Ours - EGI	2.3°	7.8
Part2Comp: 10% corr., 0.01 Noise	Ours - EGI	3.0°	5.8
KITTI			
KITTI	EGST [78]	0.0168°	0.18
3DMatch (training) → KITTI (testing)	GeDi [57]	0.40°	8.21–10.34
KITTI (training) → KITTI (testing)	GeDi [57]	0.32–0.33°	7.22–7.55
KITTI	Ours	0.42°	7.5
3DMatch			
3DMatch	EGST [78]	0.2086°	0.87
3DMatch	Ours	3.0°	9.2

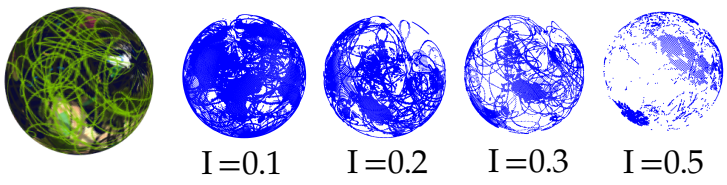

Figure 18. Effect of SphImg2SphPoints at different threshold intensities.

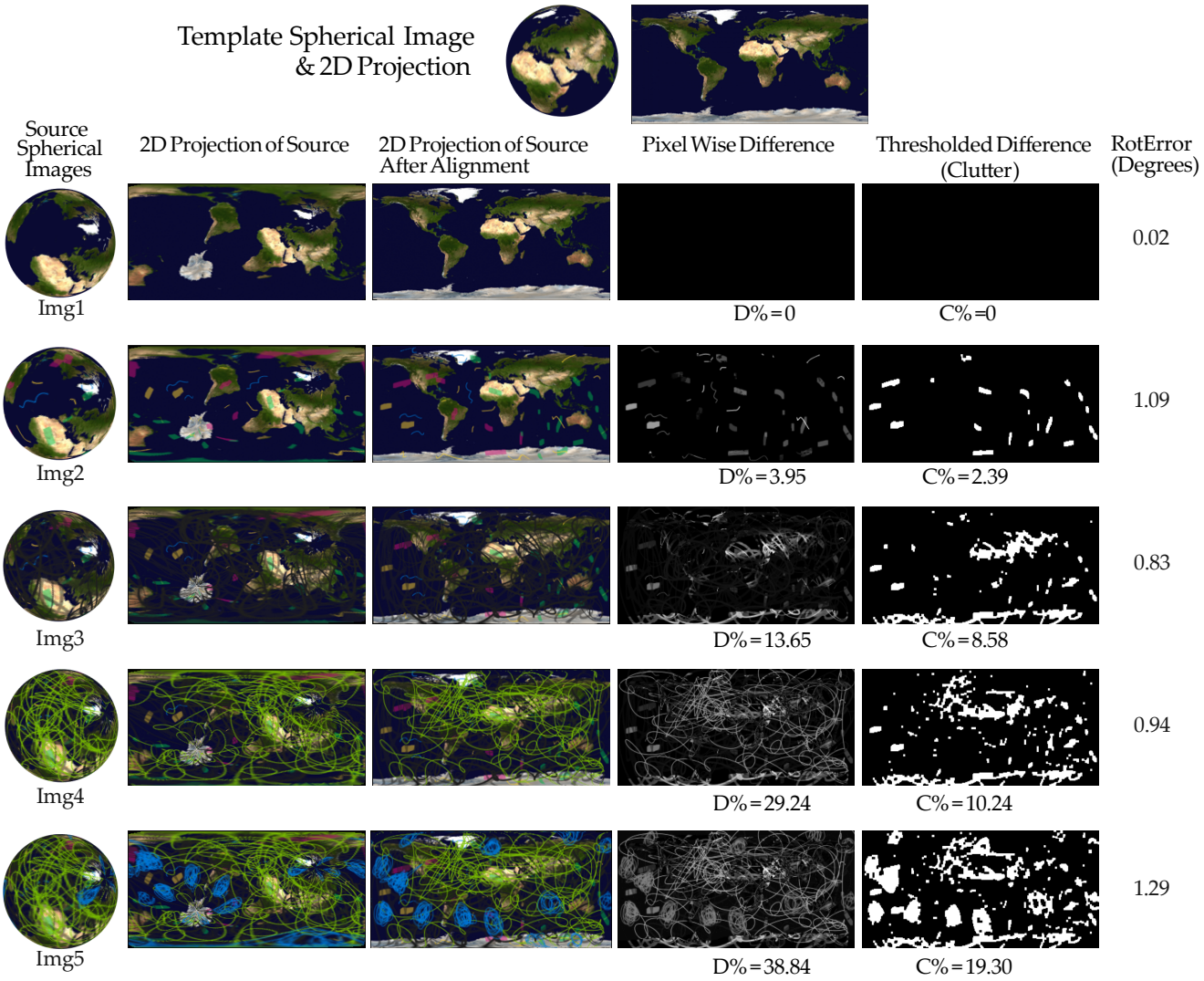

Figure 19. Results of rotation estimation from spherical images. The top row displays the template spherical image and its 2D projection. In the next six consecutive rows, source spherical images undergo varying levels of pixelwise difference (D%) or thresholded difference/clutter (C%). For this case, the input rotations were XYZ Euler angles with $\alpha = 55.27°$, $\beta = 12.11°$, and $\gamma = 11.02°$. The estimated rotation error using our algorithm is shown in the right columns.