Correspondence Matrices are Underrated

Tejas Zodage *1, Rahul Chakwate2, Vinit Sarode1, Rangaprasad Arun Srivatsan1, and Howie Choset1

1Carnegie Mellon University
2Indian Institute of Technology, Madras
*tzodage@andrew.cmu.edu

Abstract

Point-cloud registration (PCR) is an important task in various applications such as robotic manipulation, augmented and virtual reality, SLAM, etc. PCR is an optimization problem involving minimization over two different types of interdependent variables: transformation parameters and point-to-point correspondences. Recent developments in deep-learning have produced computationally fast approaches for PCR. The loss functions that are optimized in these networks are based on the error in the transformation parameters. We hypothesize that these methods would perform significantly better if they calculated their loss function using correspondence error instead of only using error in transformation parameters. We define correspondence error as a metric based on incorrectly matched point pairs. We provide a fundamental explanation for why this is the case and test our hypothesis by modifying existing methods to use correspondence-based loss instead of transformation-based loss. These experiments show that the modified networks converge faster and register more accurately even at larger misalignment when compared to the original networks.

1. Introduction

Point-cloud registration (PCR), the task of finding the alignment between pairs of point clouds is often encountered in several computer vision [15, 2] and robotic applications [1, 33, 19, 22]. A critical aspect of registration is determining a correspondence i.e. mapping between points of first cloud to the points of the second. Most approaches to registration (e.g., [4]) use simple rules-of-thumb or implement a separate procedure to establish correspondences. While these approaches have been widely used, they do suffer from computational complexity impacting their performance to determine pose parameters in real-time. Recent developments in deep learning-based registration approaches have resulted in faster and, under some circumstances, more accurate results [2, 21, 29, 9].

Unlike conventional approaches, most deep learning approaches directly estimate the pose and often do not explicitly estimate point correspondences. Instead, they implicitly learn the correspondences while being trained. While exploring the relation between correspondence and registration, we observed that perturbing the correspondence produced only small changes in the final pose estimation when...
We show that their networks can register more accurately compared to perturbations in the axis-angle representation of the rotation (see section 3.1 for more details).

Based on this observation, we hypothesize that higher registration accuracy can be achieved by training the networks to explicitly predict point correspondences instead of implicitly learning them. In order to test this hypothesis, we modify the loss function of existing registration approaches and compare the results. To develop a suitable loss function, we come up with a novel way of posing registration as a multi-class classification problem. Wherein, each point in one point cloud is classified as corresponding to a point in the other point cloud. These modified networks predict correspondences, from which pose parameters are then calculated using Horn’s method [11]. We show that the performance of each network that we modified is substantially improved (see Fig. 7). Notably, these networks give more accurate registration results when faced with large initial misalignment and are more robust to partial point cloud data as compared to the original networks.

Even though recent learning-based methods such as DCP [29] and RPMNet [36] also calculate correspondence as an intermediate step in order to calculate pose parameters, the networks are not explicitly trained to learn them. We show that their networks can register more accurately when explicitly trained to learn the correspondence.

The key contributions of our work are listed as follows—

• We provide a fundamental reasoning of why explicitly predicting correspondence provides better accuracy and results in faster convergence and verify it through extensive experimentation.

• We introduce a new way of formulating point cloud registration as a multi-class classification problem and develop a suitable loss using point correspondence.

2. Related Work

2.1. Conventional registration methods

Iterative Closest Points (ICP) [4] is one of the most popular methods for point cloud registration. ICP iteratively computes nearest neighbor correspondences and updates transformation parameters by minimizing the least-squares error between the correspondences [11]. Over the years, several variants of ICP have been developed [18]. An important area of research in this space relates to efficient ways of finding correspondences, for example point-to-plane correspondences [6], probabilistic correspondences [22, 5, 13, 14], and feature-based correspondences [17, 27]. These methods are locally optimal and hence perform poorly in the case of large misalignment.

For large misalignment, stochastic optimization techniques have been developed such as genetic algorithms [23], particle swarm optimization [28], particle filtering [20, 25] etc. Another category of methods that deal with large misalignment include globally optimal techniques. A popular approach is the globally optimal ICP (Go-ICP) [35] that uses a branch and bound algorithm to find the pose. Recently, mixed integer programming has been used to optimize a cost function over transformation parameters and correspondences simultaneously [26, 12]. These methods have theoretical guarantees to reach global optima. The fact that they explicitly optimize over correspondences motivates our work.

2.2. Deep learning-based registration methods

Some of the recent deep-learning based PCR methods train a network to directly predict the transformation between the input point clouds. PointNetLK [2] aligns the point clouds by minimizing the difference between the PointNet [16] feature descriptors of two input point clouds. PCRNet [21], passes the concatenated global feature vectors through a set of fully connected layers to predict the pose parameters. These methods operate on global point cloud features and fail to capture the local geometrical intrinsics of the points.

In order to capture local geometry, Deep Closest Point [29] learns to assign embedding to the points in each point cloud based on its nearest neighbours and attention mechanism. Further based on the similarity between the features, a correspondence matrix is generated which calculates transformation parameters that are used to define the loss function. DCP network architecture is iteratively used by PRNet [30] to align partial point clouds. This idea of using a correspondence predictor iteratively is also used by RPMnet [36], where the network structure uses FGR [38] feature descriptor unlike DGCNN [31] used by DCP and PRNet.

Some other methods such as deep global registration (DGR) [7] and multi view registration (MVR) [10], follow a two step process – (1) they find a set of correspondence pairs between two sets of 3D points using fully convolution geometric features (FCGF) [8], and (2) these correspondence pairs are passed through a network which filters outliers. Note that DGR and MVR only find a subset of all possible correspondence pairs, and yet register more accurately than methods that directly predict pose parameters. This observation motivates us to study the effect of explicitly training a network to predict all point-point correspondences. A key difference between the approach taken by MVR and DGR from our approach is that, they take point-pairs and classify them as inlier/outliers, while our approach classifies each point in one set (source) as belonging to one of multiple available classes (points in the other set).

3. Mathematical Formulation

PCR is generally posed as an optimization problem. Consider two point clouds $X = [x_1, x_2, ..., x_N] \in \mathbb{R}^{3 \times N}$
Training the networks to learn correspondences would have better registration accuracy than learning to represent correspondences. On the other hand, the alignment error quickly increases with perturbation to the rotation parameters. Thus, we hypothesize that training the networks to learn correspondences will have better registration accuracy than learning to represent correspondences.

Figure 2. Graph showing percentage of perturbation to ‘point-correspondence’ and ‘rotation vector’ vs alignment error. The plot shows that the alignment error is low even with as high as 40% wrong correspondences. On the other hand, the alignment error quickly increases with perturbation to the rotation parameters. Thus, we hypothesize that training the networks to learn correspondences would have better registration accuracy than learning to represent correspondences.

4. PCR as multi-class classification

To test our hypothesis, we first develop a suitable loss function that can explicitly learn the correspondences. An obvious choice could be a mean square error or absolute error between predicted and ground truth correspondences, but these loss functions do not provide any strong physical intuition about the correspondences.

We introduce a novel way of treating the task of correspondence assignment as a multi-class classification problem. We treat each point in \( Y \) to be a different class and each point in \( X \) belongs to one of the classes i.e., \( N_x \) examples and \( N_y \) classes. Note that each example needs to belong to at least one class but there can be classes with no corresponding example. This framework is particularly suitable to register partial point clouds where, \( \forall x_i \neq y_j \) but conversely need not be true. Note that this is fundamentally different from MVR [10], where each correspondence pair is classified as a binary: inlier or outlier and the correspondence matrix constraints are not respected.

We consider a general framework that first generates per point features \( F_x = [f_{x_1}, f_{x_2}, ..., f_{x_{N_x}}] \in \mathbb{R}^{N_x \times N_y} \) and \( F_y = [f_{y_1}, f_{y_2}, ..., f_{y_{N_y}}] \in \mathbb{R}^{N_y \times N_x} \) for input point clouds \( X \) and \( Y \) where \( f_{j \in \mathbb{R}^{N_x \times 1}} \) and \( N_y \) is the embedding space dimension. We generate a soft correspondence matrix based on a differentiable distance metric in the feature space. The metric can be distance-based as introduced.

Note that Horn’s method is just one of many closed form approaches to obtain transformation given corresponding pairs of point clouds. The results will be identical if Horn’s method is replaced by weighted SVD, or Arun’s method [3].

The soft correspondence is similar to the matrix used in conventional registration approaches [14, 12, 26, 5], where every element of the correspondence matrix denotes the probability of matching.
Note that these methods were originally developed to register point clouds with small (±45°) initial misalignment. From here on we follow the notation that method is the network trained with loss function suggested in the original paper while method_corr is trained using our loss function (cross-entropy on correspondence matrix). We train and test all these methods and method_corr on ModelNet40 [32] dataset.

DCP [29] and RPMNet [36] generate an implicit correspondence based on the similarity between per-point features of the input point clouds (Fig. 4). This intermediate correspondence is used to find the transformation parameters between input point clouds using Horn’s method [11] and weighted SVD method [36] respectively. For a network to implicitly learn correspondence, we define the loss as a function of the output transformation as suggested by the respective method. While to explicitly learn the correspondence, we define the loss as a function of intermediate correspondence as defined in Sec. 4.

We sample n number of points from a point cloud chosen from training data and denote this as point cloud X. We generate a copy of X and shuffle the order of points to generate Y'. To sample a rotation, we randomly choose a unit vector in ℝ3 and an angle \( \theta \in U(-\theta_0, \theta_0) \), this axis and angle is used to generate a rotation vector which is further transformed into a ground-truth rotation matrix \( R^* \). Here, \( \theta_0 \) depends upon the specific experiment and \( U(a, b) \) denotes a uniform distribution in the range \([a, b]\). Further we generate a ground-truth translation vector \( t^* \in [U(-0.5, 0.5), U(-0.5, 0.5), U(-0.5, 0.5)] \). Now \( Y' \) is transformed with \( R^* \) and \( t^* \) to generate \( Y \). To generate the ground-truth correspondence matrix \( C^* \), we find the nearest neighbour of each \( x_i \) in \( X \) in \( Y' \). If \( y_j' \in Y' \) is the nearest neighbour of \( x_i \) then \( C^*(j, i) \) is set to 1 and other elements of \( i \)th column are set to 0.

To generate the partial point clouds, we randomly choose a plane passing through the centroid of the original point cloud of source (X). We then randomly choose either up or down direction of the plane and remove a predetermined number of points from the source farthest from the plane.

5.1. DCP Vs DCP_corr

DCP uses DGCNN [31] features along with transformer network-based attention and co-attention mechanism to generate interrelated per point features of a point cloud. These features are used to generate probability distribution of source points on the target points matrix \( C \). They further calculate an intermediate representation of target point cloud \( Y' = CY \). DCP uses Horn’s method to estimate the rotation matrix \( R \) and translation \( t \) which minimizes the distance between corresponding points of \( Y \) and \( X \). The loss for DCP is defined as

\[
L_{DCP} = ||R^T R^* - I||_2^2 + ||t - t^*||_2^2
\]  

5. Experimental setup

We consider RPMNet [36], DCP [29], and PCRNet [21] to study the effects of training the network to learn correspondence vs training the network to learn pose parameters.
Figure 4. DCP and RPMNet architectures internally calculate the correspondence matrix $C$. This correspondence matrix is further used along with $X, Y$ to calculate $R, t$. In order to make these networks explicitly learn correspondence, we use $C$ along with ground truth $C^\ast$ to calculate cross entropy loss. Since PCRNet does not explicitly calculate $C$, we modify the network architecture and compare the PointNet’s per-point features to generate the correspondence matrix.

For all the comparisons between DCP and DCP$_{corr}$, we use learning rate $= 0.001$ as recommended by DCP.

DCP$_{corr}$ uses the correspondence matrix obtained in the intermediate step and compares it with ground truth correspondence using cross entropy

$$L_{DCP\_corr} = \text{cross entropy}(C, C^\ast)$$ (4)

5.2. RPMNet vs RPMNet$_{corr}$

RPMNet follows an iterative procedure. In each iteration, the point clouds $X$ and $Y$ and transformation from previous iterations are passed into the feature extraction network which computes point-wise features. The extracted features are then used to estimate the correspondence matrix which is further refined using Sinkhorn [24] normalization layer in an unsupervised manner. In order to estimate the transformation parameters, the target points $Y$ are weighted with the correspondence matrix weights $C$ to obtain putative source correspondences $\hat{Y} = YC$. RPMNet$_{corr}$ uses this correspondence matrix to define the cross entropy loss (see Fig. 4). RPMNet evaluates transformation parameters $R, t$ based on $X, \hat{Y}$ and $C$. These transformation parameters are then used to define the primary loss function $L_{reg}$,

$$L_{reg} = \frac{1}{N_x} \sum_{i=1}^{N_x} \left| \left( R^\ast x_i + t^\ast \right) - \left( Rx_i + t \right) \right|_1$$ (5)

RPMNet uses an additional unsupervised loss function $L_{inlier}$ which forces the network to predict majority of the correspondences as inliers. These two loss functions together form $L_{RPMNet} = L_{reg} + L_{inlier}$.

Both RPMNet and RPMNet$_{corr}$ are trained with the same hyper-parameters (as recommended in [36]), except for the learning rate. RPMNet$_{corr}$ is trained with an initial learning rate of 0.01 which decays up to 0.0001 during training. We tried a higher learning for both the methods but training of RPMNet is unstable for higher learning rates.

5.3. PCRNet Vs PCRNet$_{corr}$

PCRNet is a correspondence-free network that estimates registration parameters given a pair of input point clouds $(X$ and $Y)$. As shown in Fig. 4, PCRNet uses PointNet [16] as a backbone to compute the point-wise features of each input point cloud arranged in a siamese architecture. In order to avoid input permutations, a symmetry function (max-pool) is operated on point-wise features to obtain a global feature vector $(\in \mathbb{R^{12x1024}})$. PCRNet concatenates the global feature vectors of both the inputs and uses a set of fully connected layers to regress the registration parameters. Rather than defining the loss function on the ground truth transformation, PCRNet uses chamfer distance (CD) as the loss function,

$$CD(X, Y) = \frac{1}{N_x} \sum_{x_i \in X} \min_{y_j \in Y} \|x_i - y_j\|_2 + \frac{1}{N_y} \sum_{y_j \in Y} \min_{x_i \in X} \|y_j - x_i\|_2$$ (6)

CD calculates the average closest distance between the template $X$ and the point cloud obtained by applying predicted transformation on $Y$.

Even though PCRNet uses an unsupervised loss function, CD is a function of $X, Y, R$ and $t$. In other words, the training of PCRNet again depends on the accuracy of $R, t$ when compared to the ground truth.

6. Results

In this section, we present results of different existing approaches, referred to as method, and provide comparisons to versions of those approaches modified by training using our correspondence based loss – referred to as
method corr. We specifically highlight the improvement shown by method corr compared to method to large initial misalignment errors as well as ability to register partial point-clouds.

### Table 1. Effect of initial misalignment on registration accuracy

| Rotation range (deg) | Rotation MAE (deg) | Correspondence (%) |
|----------------------|--------------------|--------------------|
|                      | DCP                | DCP corr           |
|                      |                   |                   |
| 0-30                 | 0.99              | 0.005             |
| 30-60                | 1.55              | 0.008             |
| 60-90                | 1.69              | 0.010             |
| 90-120               | 1.56              | 0.010             |
| 120-150              | 1.62              | 0.010             |
| 150-180              | 1.64              | 0.010             |

### 6.1. DCP Vs DCP corr

The authors of DCP, consider 1024 points in all of their experiments. Due to limited GPU space, we re-ran all the DCP experiments using 512 points with the same hyper-parameters including learning rate for both. We sample rotations from SO(3) with rotation vectors instead of Euler angles. This helped us to train DCP even for large misalignment. For different experimental settings Fig. 5 shows the comparison between DCP and DCP corr. The first column shows that every training procedure converged. Second shows the accuracy of correspondence estimation of both the methods. Third column shows rotation error as an RMSE over Euler angle error and fourth column denotes translation error.

### Experiment 1.1

We have $N_x = 512$ points in the source and $N_y = 512$ points in the target. The initial misalignment between them is uniformly sampled from $SO(3)$ while the translation is bound in cube of unit size centered on origin. As observed in Fig. 5 we can see that DCP corr converges faster than DCP and is more accurate.

### Experiment 1.2

The results of this section are visualized in Fig. 7. We have $N_x = 358$ points in the source and $N_y = 512$ points in the target. The source point cloud is made partial as described in Sec. 5. We observe that even though DCP’s loss function converges, the RMSE rotation error is 14.7° while the rotation error of DCP corr is 0.51°. This can be considered as an empirical evidence that multi-class classification approach can deal with partial data without any major modification to the network architecture.

### Experiment 1.3

In this experiment, we compare DCP to DCP corr for the specific task DCP was developed for, i.e. full-to-full point cloud registration for initial misalignment in the range of $[-45^\circ,$ $+45^\circ]$. We observe that both the networks converge, and the rotation accuracy of DCP and DCP corr are $1.036^\circ$ and $0.034^\circ$ respectively.

### Experiment 1.4

In this experiment we observe the effect of initial misalignment on registration accuracy of DCP and DCP corr trained for arbitrary initial misalignment (Table 1). For this experiment, we set the translation to zero and only allow a rotational misalignment between the input point clouds. We observe that DCP corr always registers more accurately than DCP, which is attributed to the
6.2. RPMNet Vs RPMNet_{corr}

We present the comparisons between RPMNet and RPMNet_{corr} in Fig. 6. Unlike the previous experiment with DCP, the rotation error metric used to evaluate these experiments is the mean absolute anisotropic rotation error also known as axis angle error. We chose this metric to be compliant with the choice of the authors of RPMNet [36]. Likewise, we present the Chamfer distance (CD) between registered point clouds, in the fourth column of Fig. 6, as suggested by the authors of RPMNet [36]. The initial misalignment in translation is sampled uniformly between \([-0.5, 0.5]\).

**Experiment 2.1** In this experiment, both the point clouds have \(N_x = N_y = 1024\) points. The misalignment between these clouds is uniformly sampled from \(SO(3)\). It can be observed that the rotation error converges faster and to a lower value of \(0.059^\circ\) with RPMNet_{corr} as compared to an error of \(0.56^\circ\) for RPMNet.

**Experiment 2.2** To test the ability of multi-class classification approach to handle partial point clouds, in this experiment we generate the partial source point cloud by retaining \(70\%\) of the points above a random plane such that \(N_x = 717\) and \(N_y = 1024\). We carry out this experiment with uniform sampling from \(SO(3)\). Note that, even though one of the key features of RPMNet is the ability to deal with partial point clouds, RPMNet_{corr} has higher registration accuracy of \(0.34^\circ\) compared to \(3.79^\circ\) of RPMNet.

**Experiment 2.3** RPMNet is specifically designed for \([-45^\circ, 45^\circ]\) initial misalignment. Even in this range, we observe that RPMNet_{corr} converges faster and registers more accurately. We also observe that eventually RPMNet reaches \(96\%\) correspondence accuracy. We believe that the RPMNet’s Sinkhorn algorithm along with the unsupervised loss on correspondence \(L_{inlier}\), pushes the intermediate correspondence matrix towards the ground truth correspondence matrix in an unsupervised manner.

**Experiment 2.4** In this experiment we study the effect of initial misalignment on the registration accuracy of RPMNet and RPMNet_{corr}. Both the networks are trained with arbitrary initial misalignment in the range of \([-180^\circ, 180^\circ]\) between the input point clouds. In this experiment, we set the translation to zero and only allow a rotational misalignment between the input point clouds. We calculate Mean Absolute Error (MAE) between predicted and ground truth rotation in Euler angles. We observe from Table 2 that the MAE for rotation is always lower for RPMNet_{corr} when compared to RPMNet.

6.3. PCRNet Vs PCRNet_{corr}

The results showing the comparison between PCRNet and PCRNet_{corr} are shown in 7. We only provide a rotational misalignment between the input point clouds.

**Experiment 3.1** We consider \(N_x = N_y = 1024\) points for both the input point clouds. We train PCRNet with the hyper-parameters recommended in [21] and compare it with PCRNet_{corr}. Note that PCRNet_{corr} has fewer tunable parameters than PCRNet due to the removal of MLPs. We observe that both the approaches converge to \(\approx 5^\circ\) rotation accuracy. Based on the results of this experiment, we believe that PCRNet lacks depth or number of parameters to achieve higher accuracy. Another reason to believe this is, even after doing a thorough hyper-parameter search, we could not achieve correspondence accuracy of \(\geq 70\%\).

**Experiment 3.2** In this experiment, we have two point clouds with 100 points each. The initial misalignment between them is in the range of \([-45^\circ, 45^\circ]\). We observe that PCRNet converges to a rotation accuracy of \(9.97^\circ\) compared to \(1.8^\circ\) of PCRNet_{corr}.

**Experiment 3.3** We repeat the previous experiment but use an initial misalignment that is uniformly sampled from \(SO(3)\). We observe that PCRNet_{corr} outperforms PCRNet and is able to learn correspondences and the rotation accuracy reaches \(21^\circ\) at the end of 250 epochs.

7. Conclusion and Future Work

In this paper we demonstrate that higher registration accuracy can be achieved if a network is trained to explicitly learn correspondences instead of learning them implicitly by training on registration parameters. This paper adds to the ever-increasing body of work demonstrating how carefully selecting the desired output of a data-driven approach can lead to drastic improvements in performance. We observe faster convergence, higher registration accuracy and ability to register partial point clouds when networks are explicitly trained to learn correspondence instead of pose parameters. We also developed a new way to approach registration as a multi-class classification task.

While in this work we have limited ourselves to results on ModelNet40, we plan to extend it to real world datasets such as 3DMatch [37] and Sun3d [34]. In addition, future work will involve extended the multi-class classification approach to deal with outliers in the point clouds.
Figure 6. Results of experiments on RPMNet vs RPMNet_corr

| Loss (Training) | Correspondence Accuracy | Rotation error | Chamfer Distance |
|-----------------|-------------------------|----------------|-----------------|
| Nx = 1024; Ny = 1024; Trained and tested on [-180°,180°] |

2.1

Figure 7. Results of experiments on PCRNet Vs PCRNet_corr

| Loss (Training) | Rotation error (Testing) | Correspondence accuracy |
|-----------------|--------------------------|-------------------------|
| Nx = 717 (partial); Ny = 1024; Trained and tested on [-180°,180°] |

2.2

| Loss (Training) | Rotation error (Testing) | Correspondence accuracy |
|-----------------|--------------------------|-------------------------|
| Nx = 717 (partial); Ny = 1024; Trained and tested on [-45°,45°] |

2.3

| Loss (Training) | Rotation error (Testing) | Correspondence accuracy |
|-----------------|--------------------------|-------------------------|
| Nx = 1024; Ny = 1024; Trained and tested on [-45°,45°] |

3.1

| Loss (Training) | Rotation error (Testing) | Correspondence accuracy |
|-----------------|--------------------------|-------------------------|
| Nx = 100; Ny = 100; Trained and tested on [-45°,45°] |

3.2

| Loss (Training) | Rotation error (Testing) | Correspondence accuracy |
|-----------------|--------------------------|-------------------------|
| Nx = 100; Ny = 100; Trained and tested on [-180°,180°] |

3.3
Supplementary

A. Registration in the presence of outliers

To extend multi-class classification approach to filter outliers, we consider a general framework that first generates per point features \( F_X = [f_{x_1}, f_{x_2}, \ldots, f_{x_{N_x}}] \in \mathbb{R}^{N_x \times N_y} \) and \( F_Y = [f_{y_1}, f_{y_2}, \ldots, f_{y_{N_y}}] \in \mathbb{R}^{N_y \times N_y} \) for input point clouds \( X \) and \( Y \). Here \( f_{y_i} \in \mathbb{R}^{N_y \times 1} \) and \( N_y \) is the embedding space dimension. In the absence of outliers, we predicted the correspondence matrix (probability of each source point to belong to one of the target points) as

\[
C = \text{softmax}(F_Y^T F_X) \in \mathbb{R}^{N_y \times N_x}
\]

In presence of outliers, we want to classify a source point to belong to one of the target points or as an outlier. i.e. we need \( N_y + 1 \) number of classes. To get an extra outlier class for classification we use a feature vector embedding \( f_{Y_O} \in \mathbb{R}^{N_y \times 1} \) that can suitably represent an outlier. With such embedding for outliers, we can predict an outlier variant of correspondence matrix \( C^O = \mathbb{R}^{N_y+1 \times N_x} \) as

\[
C^O = \text{softmax}([f_{y_1}, f_{y_2}, \ldots, f_{y_{N_y}}, f_{Y_O}]^T F_X)
\]

Note that for \( i \)th source point, \( C_{1:i}^O, C_{2:i}^O, \ldots, C_{N_y:i}^O \) denotes the probability of the point belonging to \( y_1, y_2, \ldots, y_{N_y} \) respectively and \( C_{N_y+1:i}^O \) denotes the probability of it being an outlier.

In case that a point is predicted as an outlier, we want the probability of it being classified as one of the source points, as less as possible. In order to do so, we want the outlier embedding to have large-negative projections on all the target point embeddings so that after the softmax operation probabilities of outlier to be classified as an inlier will be close to zero. There can be various ways to obtain such embedding. For preliminary experiments, we generate such embedding \( f_{Y_O} \) as,

\[
f_{Y_O} = \arg \min_f \left( \| F_Y^T f - b1_{N_y} \|_2 \right)
\]

Here \( 1_{N_y} \) denotes a column vector of ones of length \( N_y \) and \( b \) is a scalar. We empirically choose \( b \) to be \(-1\).

B. Experiment - DCP Vs DCP_corr in the presence of outliers

We take \( N_x = 512 \) and \( N_y = 512 \) with initial misalignment in the range of \([-45^\circ, 45^\circ]\) for both DCP and DCP_corr. Then position of 10% points from \( X \) is randomly corrupted to be a random point in a unit cube centered at the origin. To generate the outlier variant of ground-truth correspondence matrix \( C^{O^*} \), we find the nearest neighbour of each \( x_i \in X \) in \( Y' \) and it’s distance to the nearest neighbor. If \( d_i \) is less than a predefined threshold then \( y'_{y_j} \in Y' \) is denoted as nearest neighbour of \( x_i \) by setting \( C^{O^*}(j,i) \) to 1 and other elements of \( j \)th column to be 0. If \( d_i \) is greater than the predefined threshold, then \( i \)th source point is marked as an outlier by setting \( C^{O^*}(N_y + 1, i) \) to 1 and other elements of \( i \)th column to be 0.

Further DCP_corr is trained to minimize the loss function cross entropy \( (C^O, C^{O^*}) \) as mentioned in Sec. 4. The last row of \( C^O \) is used as outlier weights and a weighted SVD operation is performed to obtain transformation parameters (refer [6] for details). For DCP_corr, we observe an rotation error (RMSE) of 1.485º and an translation error (RMSE) of 0.000138 after 15 epochs. The outlier filtering by DCP_corr can be visualized in (Fig. 8). On the other hand, DCP is trained with the mean squared error on transformation parameters. We observe a rotation error of 6.704º and translation error of 0.519 units. The effect of outliers on DCP vs DCP_corr can be observed in the Fig. 9.

References

[1] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze. Our-cvfh-oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In Joint DAGM (German Association for Pat-
Figure 9. Visualization of DCP_corr and DCP registration in presence of outliers

[1] T. Kanade, K. Kim, A. Srivatsan, and S. Lucey. Multiview stereo and registration of nonrigid shapes for dynamic scenes. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3776–3783. IEEE, 2011.

[2] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using point-
G. Izatt, H. Dai, and R. Tedrake. Globally optimal object pose estimation in point clouds with mixed-integer programming. In Proceedings third international conference on 3-D digital imaging and modeling, pages 145–152. IEEE, 2001.

R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski. Detecting and segmenting objects for mobile manipulation. In 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pages 47–54. IEEE. 2009.

R. Sandhui, S. Dambreville, and A. Tannenbaum. Point set registration via particle filtering and stochastic dynamics. IEEE transactions on pattern analysis and machine intelligence, 32(8):1459–1473, 2009.

V. Sarode, X. Li, H. Goforth, Y. Aoki, R. A. Srivatsan, S. Lucey, and H. Choset. PCRNet: Point Cloud Registration Network using PointNet Encoding. arXiv e-prints, page arXiv:1908.07906, Aug. 2019.

A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In Robotics: Science and Systems, volume 2, 2009.

L. Silva, O. R. P. Bellon, and K. L. Boyer. Precision range image registration using a robust surface interpolation measure and enhanced genetic algorithms. IEEE transactions on pattern analysis and machine intelligence, 27(5):762–776, 2005.

R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. The annals of mathematical statistics, 35(2):876–879, 1964.

R. A. Srivatsan and H. Choset. Multiple start branch and prune filtering algorithm for nonconvex optimization. In The 12th International Workshop on The Algorithmic Foundations of Robotics, Springer, 2016.

R. A. Srivatsan, T. Zodage, and H. Choset. Globally optimal registration of noisy point clouds. arXiv preprint arXiv:1908.08162, 2019.

I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., volume 2, pages II–II. IEEE, 2003.

M. P. Wachowiak, R. Smoliková, Y. Zheng, J. M. Zurada, and A. S. Elmaghraby. An approach to multimodal biomedical image registration utilizing particle swarm optimization. IEEE Transactions on evolutionary computation, 8(3):289–301, 2004.

Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. CoRR, abs/1905.03304, 2019.

Y. Wang and J. M. Solomon. Pnet: Self-supervised learning for partial-to-partial registration. In Advances in Neural Information Processing Systems, pages 8814–8826, 2019.

Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog), 38(5):1–12, 2019.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric scenes. In Proceedings of the IEEE conference on
computer vision and pattern recognition, pages 1912–1920, 2015. 4324

[33] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In Proceedings of Robotics: Science and Systems, Pittsburgh, Pennsylvania, June 2018. 4321

[34] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In Proceedings of the IEEE international conference on computer vision, pages 1625–1632, 2013. 4327

[35] J. Yang, H. Li, and Y. Jia. Go-ICP: Solving 3D Registration Efficiently and Globally Optimally. In 2013 IEEE International Conference on Computer Vision (ICCV), pages 1457–1464, Dec 2013. 4322

[36] Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 4322, 4324, 4325, 4327

[37] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1802–1811, 2017. 4327

[38] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In European Conference on Computer Vision, pages 766–782. Springer, 2016. 4322