Point Cloud Registration as a Classification Problem

Tejas Zodage

August 2021

CMU-RI-TR-21-47

Robotics Institute

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Abstract

Point Cloud Registration(PCR) is an important step in fields such as robotic manipulation, augmented and virtual reality, SLAM, etc. In the context of computer vision, registration in general refers to the process of aligning data obtained from different frames and as the name suggests PCR is the task of aligning point-clouds. The main contribution of this thesis is drawing parallels between PCR and classification which allows us to apply well studied concepts from classification in PCR. This thesis further shows two applications of drawing such parallels in the context of deep learning based PCR. We show the use of cross-entropy loss, and discrepancy loss from classification for partial to full PCR, and outlier filtering respectively. We finally show that many of the existing deep learning based PCR architectures can be easily modified to be trained using the loss functions from classification literature.

Acknowledgments

I am grateful to my co-advisers Drs. Howie Choset and Matthew Travers for providing me with the opportunity to work on these really interesting research topics. They have always inspired me to constantly improve my communication and and research skills. I really appreciate there help on my non-academic problems like job search and dealing with health issues. I would like to thank my family, Aai, Anna and Rajas for believing in me even when I did not. Their sacrifices for me are countless. This thesis wouldn't have been possible without my colleague and mentor Arun Srivatsan. He literally taught me how to do research, what is point cloud registration, and a lot about music, art, movies, and life in general. Rahul Chakwate can be definitely treated as the second author of this thesis. I am grateful to him to bear with my constant nagging about the debugging methods and my OCD about presentations. I would like to thank my lab mates and room mates Abhimanyu and Vinit for continuous emotional, technical, and logistical support to get me through my master's. I can never forget those all night pizza and brainstorming sessions with Abhimanyu and chai and code sessions with Vinit. Speaking of Chai, I can not thank Shruti, Sampada, Rohit Jena and Rohit Singh enough to make those chai time sessions more interesting than anything. Lastly, a big thanks to the administrative staff, Peggy, Rachel, and BJ for dealing with my complicated visa situations. I am very lucky to have these many well wishers in my life.

Table of Contents

List of Figures

Chapter 1
Introduction

Point cloud registration (PCR), the task of finding the alignment between pairs of point clouds, is often encountered in several computer vision [4, 5] and robotic applications [6–9]. The main contribution of this thesis is drawing parallels between PCR and multi-class classification problem Fig.1.1.

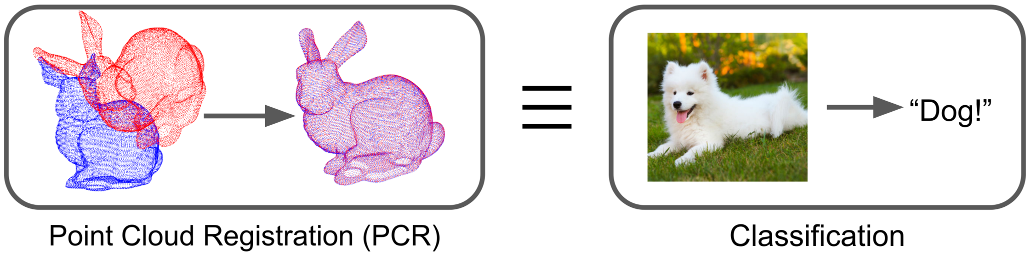


Figure 1.1: Main contribution of this thesis - finding parallels between Point Cloud Registration(PCR) and classification.

Finding such parallels can assist advancing research in PCR domain using already established research in the classification domain and vice-versa. This thesis further explores applications of finding these parallels in the context of deep-learning based PCR. We mainly consider two cases of PCR, partial point cloud to full point cloud registration and outlier filtering.

PCR community have been continuously working towards developing fast and accurate methods [9–16]. Recent developments in deep learning-based PCR approaches have resulted in faster and, under some circumstances, more accurate results [1, 5, 17, 18] as compared to the conventional or non-deep-learning based methods. Deep learning based methods approximate an oracle function using an artificial neural network, given a training data set consisting of inputs of the oracle function and expected outputs. The weights of a neural network are adjusted to minimize a metric comparing network's predicted output with expected output. This metric is typically known as loss function. The choice of the loss function is a critical aspect of any deep learning based algorithm and it is not obvious on what loss function will be most appropriate for a given application. In this thesis we show, how we can import well-studied and experimented loss functions from classification domain to PCR opposed to starting from scratch. We bring in the cross-entropy loss [19] developed by the classification community for partial to full point cloud registration and discrepancy loss [3] for outlier filtering.

The upcoming sections of this chapter will discuss the related work and mathematical formulation of PCR. Chapter 2 describes the similarities between multi-class classification and PCR. It further shows examples of modifying existing registration based methods to learn correspondence using cross-entropy loss. In Chapter 3 we describe the discrepancy loss in the context of image classification and its extension to PCR for outlier filtering. We show how can we modify the existing network DCP [1] in order to filter outliers. Both the chapters show experiments on the ModelNet40 [20] data set on various metrics such as chamfer distance between points of registered point clouds, root mean squared error of error in euler angles after registration, percentage of accurately predicted point clouds etc.

1.1 Related Work

In this section, we qualitatively review some prior and related work - first in conventional registration and then in its deep learning variant. Second, we assign the mathematical symbols and terminology for registration; this terminology and symbols will be used throughout this thesis

1.1.1 Conventional registration methods

Point cloud registration seeks to computer a rigid body displacement - translation and rotation - between two point clouds. Before computing the parameters of such a transformation, most approaches establish a correspondence between the points in their respective point clouds. Frankly, if the correspondance is know, then registation becomes a trivial optimization problem to solve. Iterative Closest Points (ICP) [14] 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 [21]. Over the years, several variants of ICP have been developed [22]. An important area of research in this space relates to efficient ways of finding correspondences, for example point-to-plane correspondences [23], probabilistic correspondences [9, 24–26], and feature-based correspondences [27, 28]. These methods are locally optimal and hence perform poorly in the case of large misalignment, *i.e.* when the transformation parameters between the two sets are large. The large misalignment can be expected for tasks like pose estimation, when the object in the scene and the model of the object can be in totally different orientations.

For large misalignment, stochastic optimization techniques have been developed such as genetic algorithms [29], particle swarm optimization [30], particle filtering [12, 31] etc. Another category of methods that deal with large misalignment include globally optimal techniques. A popular approach is the globally optimal ICP (Go-ICP) [15] 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 [13, 32]. These methods have theoretical guarantees to reach global optimal. The fact that they explicitly optimize over correspondences, motivates our work.

1.1.2 Deep learning-based registration methods

Deep learning PCR methods take as input two points cloud and out put a transformation. Some of the recent deep-learning based PCR methods train a network to directly predict the transformation between the input point clouds. PointNetLK [5] aligns the point clouds by minimizing the difference between the PointNet [33] feature descriptors of two input point clouds. PCRNet [17], first extracts feature descriptors of points using PointNet [33] and extracts a global feature descriptor for each input point cloud individually. These global feature descriptors are then concatenated and are passed through 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, approaches like Deep Closest Point(DCP) [1] learn to assign embedding (also known as feature descriptor) to the points in each point cloud based on its nearest neighbors 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. The loss function over here is the error between predicted transformation parameters and ground-truth transformation paramters. The DCP network architecture is iteratively used by PRNet [34] to align partial point clouds. This idea of using a correspondence predictor iteratively is also used by RPMnet [35], where the network structure uses FGR [36] feature descriptor unlike DGCNN [37] used by DCP and PRNet.

Some other methods such as Deep Global Registration (DGR) [2] and Multi-View Registraiton (MVR) [38], follow a two step process – (1) they find a set of plausible correspondence pairs between two sets of 3D points using Fully Convolution Geometric Features (FCGF) [39], and (2) these plausible correspondence pairs are passed through a network which filters outliers. After filtering outliers, the registration is trivial using weighted SVD. Note that DGR and MVR only find a subset of all plausible correspondence pairs, and 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. Treating PCR as a classification problem, provides us with ability to use the methods from classification literature to train our networks to learn the point-point correspondence.

1.2 Mathematical Formulation

PCR is generally posed as an optimization problem. Consider two point clouds $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{N_x}] \in \mathbb{R}^{3\times N_x}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{N_y}] \in \mathbb{R}^{3\times N_y}$ containining N_x and N_y points respectively, where $\boldsymbol{x}_i \in \mathbb{R}^3$ and $\boldsymbol{y}_j \in \mathbb{R}^3$ are the points in the respective point clouds and generally, $N_x \neq N_y$. The ground truth transformation, $\boldsymbol{R}^* \in SO(3)$ and $\boldsymbol{t}^* \in \mathbb{R}^3$, that aligns the two point clouds can be represented as

$$\boldsymbol{R}^*, \boldsymbol{t}^* = \underset{\boldsymbol{R},\boldsymbol{t}}{\operatorname{argmin}} \left(\sum_{i=1}^{N_x} ||\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{y}_{\pi(\boldsymbol{x}_i)}||_2 \right), \tag{1.1}$$

where $||\dots||_2$ is the L2 norm and, π denotes a function $\pi : \mathbf{x}_i \to \mathbb{N}$, such that $\pi(\boldsymbol{x}_i)$ is the index of the point corresponding to $\boldsymbol{x}_i$ in $\boldsymbol{Y}$. This function can be represented by a binary matrix known as correspondence matrix $\boldsymbol{C} \in \mathbb{R}^{N_y \times N_x}$, where $C_{i,j} \in \{0, 1\}$ for $i \in [1, \dots, N_x]$ and $j \in [1, \dots, N_y]$. $\boldsymbol{C}_{j,i} = 1$ implies that $\boldsymbol{y}_j$ corresponds to $\boldsymbol{x}_i$. Or, $\boldsymbol{y}_j = \boldsymbol{y}_{\pi(\boldsymbol{x}_i)} = \boldsymbol{Y}\boldsymbol{C}_{:,i}$. Here $\boldsymbol{C}_{:,i}$ represents the i^{th} column of C. Since the correspondence is unknown, registration is restated as,

$$\boldsymbol{R}^*, \boldsymbol{t}^*, \boldsymbol{C}^* = \underset{\boldsymbol{R},\boldsymbol{t},\boldsymbol{C}}{\operatorname{argmin}} \left(\sum_{i=1}^{N_x} ||\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{Y}\boldsymbol{C}_{:,i}||_2 \right) \tag{1.2}$$

Where $\boldsymbol{C}^*$ is the ground truth correspondence matrix. Note that $\hat{\boldsymbol{Y}} = \boldsymbol{Y}\boldsymbol{C}$ denotes the rearranged $\boldsymbol{Y}$ such that i^{th} point of $\boldsymbol{X}$ corresponds to i^{th} point of $\hat{\boldsymbol{Y}}$.

In the case when the point-cloud $\boldsymbol{X}$ is expected to have outliers, an extra variable $\boldsymbol{O} \in \mathbb{R}^{N_x}$, and a constant $\lambda \in \mathbb{R}$, where $O_i \in \{0, 1\}$ for $i \in [1, \dots, N_x]$. Here O_i is a binary variable which takes value of 1 if i^{th} point is an outlier and λ is a threshold distance between transformed $\boldsymbol{x}_i$ and $\boldsymbol{y}_{\pi(\boldsymbol{x}_i)}$ beyond which $\boldsymbol{x}_i$ will be considered an outlier. In this case, the problem can be defined as

$$\boldsymbol{R}^*, \boldsymbol{t}^*, \boldsymbol{C}^*, \boldsymbol{O}^* = \underset{\boldsymbol{R},\boldsymbol{t},\boldsymbol{C},\boldsymbol{O}}{argmin} \left(\sum_{i=1}^{N_x} ||\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{Y}\boldsymbol{C}_{:,i}||_2 (1 - O_i) + \lambda O_i \right) \tag{1.3}$$

Chapter 2
Partial to full point cloud registration with cross-entropy loss

2.1 Introduction

A critical aspect of registration is determining a correspondance *i.e.* mapping between points of first cloud to the points of the second. Most approaches to registration (e.g., [14]) use simple rules-of-thumb or implement a separate procedure to establish correspondances. 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 [1, 5, 17, 18].

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 compared to perturbations in the axis-angle representation of the rotation (see section 2.1.1 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

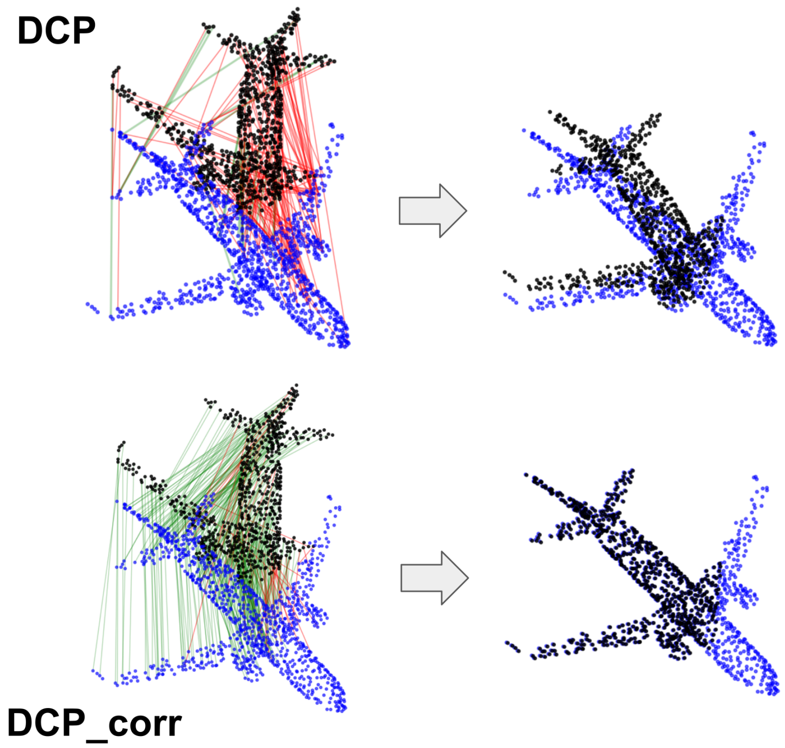


Figure 2.1: When learning to predict the pose parameters, deep-learning-based methods such as DCP [1] learn correspondence (mapping between the points of point clouds) implicitly. If the same networks are trained to explicitly learn correspondence (DCP_corr), the resulting registration is more accurate. Template point cloud is shown in blue, source point cloud in black. Green arrows show correct correspondence. Red arrows show incorrect correspondence.

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 [21]. We show that the performance of each network that we modified is substantially improved (see Fig. 2.8). 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 [1] and RPMNet [35] also

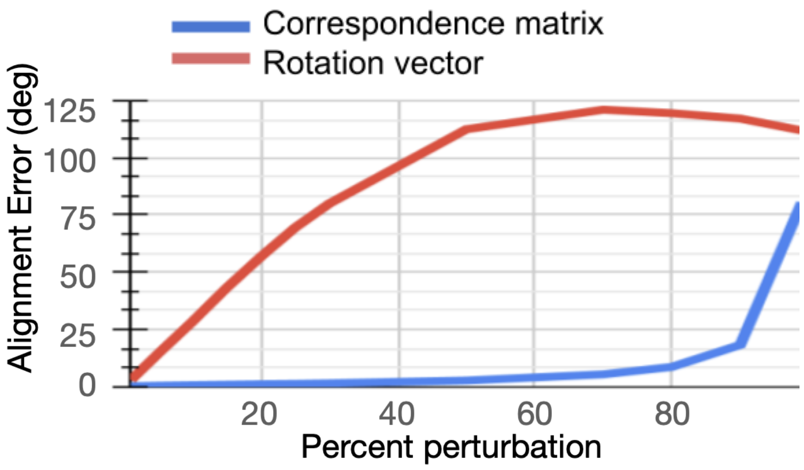


Figure 2.2: Graph showing percentage of perturbation to 'point-correspondence' and 'rotation vector' vs alignment error. The plot shows that the alignment error is low even with as high as 40% wrong correspondences. On the other hand the alignment error quickly increases with perturbation to the rotation parameters. Thus, we hypothesize that training the networks to learn correspondences would have better registration accuracy than learning pose parameters.

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.

2.1.1 Robustness of correspondences Vs robustness of transformations

To understand the effect of wrong correspondences on alignment, we perform the following experiment. We initially sample n points randomly from a unit 3D cube and denote it as point cloud $\boldsymbol{X}$. We then transform $\boldsymbol{X}$ with a random but known rotation $\boldsymbol{R}^*$ to create point cloud $\boldsymbol{X'} = \boldsymbol{R}^*\boldsymbol{X}$. For convenience, we convert $\boldsymbol{R}^*$ to a rotation vector form $\boldsymbol{v}^* \in \mathbb{R}^3$ and add $p\%$ corruption to generate $\boldsymbol{v}^{pert} = \boldsymbol{v}^* + \boldsymbol{v}^{corrupt}$. We then calculate the error between $\boldsymbol{R}^{pert}$ and $\boldsymbol{R}^*$. This is noted as alignment error between corrupted rotation and ground truth rotation. We gradually increase the percentage corruption and calculate the corresponding alignment error (Figure 2.2). To observe the robustness of the correspondences, in another independent experiment, we randomly corrupt $p\%$ of the ground truth correspondence and calculate the resulting rotation matrix based on perturbed correspondences using Horn's method [1] [21]. The error between the ground truth rotation and perturbed rotation is then calculated. We gradually increase the percentage corruption and observe it's effect on the rotation error. We observe that even if 40% of the correspondences are wrong, the alignment error is $\approx 5°$.

Based on this observation we hypothesize that if a network is trained explicitly to predict correspondence, the network will align point clouds more accurately than a network with similar architecture but trained to predict pose.

2.2 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 correspondence but these loss functions do not provide any strong physical intuition about the correspondence.

We introduce a novel way of treating the task of correspondence assignment as a multi-class classification problem. We treat each point in $\boldsymbol{Y}$ to be a different class and each point in $\boldsymbol{X}$ 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, $\forall \boldsymbol{x}_i, \exists \boldsymbol{y}_j$ but converse need not be true. Note that this is fundamentally different from MVR [38], where each correspondence pair is classified as a binary: inlier or outlier and the correspondence

[1]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 [10].

matrix constraints are not respected.

We consider a general framework that first generates per point features $\boldsymbol{F_X} = [\boldsymbol{f}_{x_1}, \boldsymbol{f}_{x_2}, ..., \boldsymbol{f}_{x_{N_x}}] \in \mathbb{R}^{N_e \times N_x}$ and $\boldsymbol{F_Y} = [\boldsymbol{f}_{y_1}, \boldsymbol{f}_{y_2}, ..., \boldsymbol{f}_{y_{N_y}}] \in \mathbb{R}^{N_e \times N_y}$ for input point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$ where $\boldsymbol{f}_j \in \mathbb{R}^{N_e \times 1}$ and N_e is the embedding space dimension. We generate a soft correspondence matrix based on a differentiable distance metric in the feature space [2]. The metric can be distance-based as introduced by MVR [38] or projection-based as suggested in DCP [1]. Without any loss of generality, we choose DCP's approach to generate a soft correspondence matrix *i.e.* a matrix where each element denotes probability of a correspondence between point pairs as $\boldsymbol{C} = softmax(\boldsymbol{F_Y^T F_X})$. We compare $\boldsymbol{C}$ with a ground truth correspondence matrix $\boldsymbol{C}^*$. We define the ground-truth correspondence as nearest neighbor of a point $\boldsymbol{x}_i$ in $\boldsymbol{Y}$ when $\boldsymbol{X}$ and $\boldsymbol{Y}$ are aligned. It is worth noting that this is not a reversible mapping *i.e.* if $\boldsymbol{y}_1 \in \boldsymbol{Y}$ is the nearest neighbor of $\boldsymbol{x}_1$, it is possible that the nearest neighbor of $\boldsymbol{y}_1$ in $\boldsymbol{X}$ is $\boldsymbol{x}_j$, where $j \neq 1$. This adds a constraint on the correspondence matrix that each the sum of the elements of each column should add up to one.

The multi-class classification framework allows us to use a cross-entropy loss, L_{CE}. This loss function implicitly applies the constraint that sum of the elements of a column should be one unlike binary cross entropy (BCE). For the sake of convenience of notation, we define $\boldsymbol{C'} = \boldsymbol{F_Y^T F_X}$

$$L_{CE}(\boldsymbol{C'}, \boldsymbol{C^*}) = -\sum_{i=1}^{N_x} log \left(\frac{exp\left(\sum_{j=1}^{N_y} \boldsymbol{C'}_{j,i} \boldsymbol{C^*}_{j,i}\right)}{\sum_{j=1}^{N_y} exp(\boldsymbol{C'}_{j,i})} \right)$$

While beyond the scope of this paper, it is worth noting that our framework can be further modified to add an additional class to classify an $\boldsymbol{x}_i$ as an outlier. If we incorporate an extra class for outliers in the correspondence matrix $\boldsymbol{C} \in \mathbb{R}^{N_y+1 \times N_x}$, this becomes a single-step generalised version of the two step process used by DGR (see Fig. 2.3).

[2]The soft correspondence is similar to the matrix used in conventional registration approaches [13, 24, 26, 32], where every element of the correspondence matrix denotes the probability of matching.

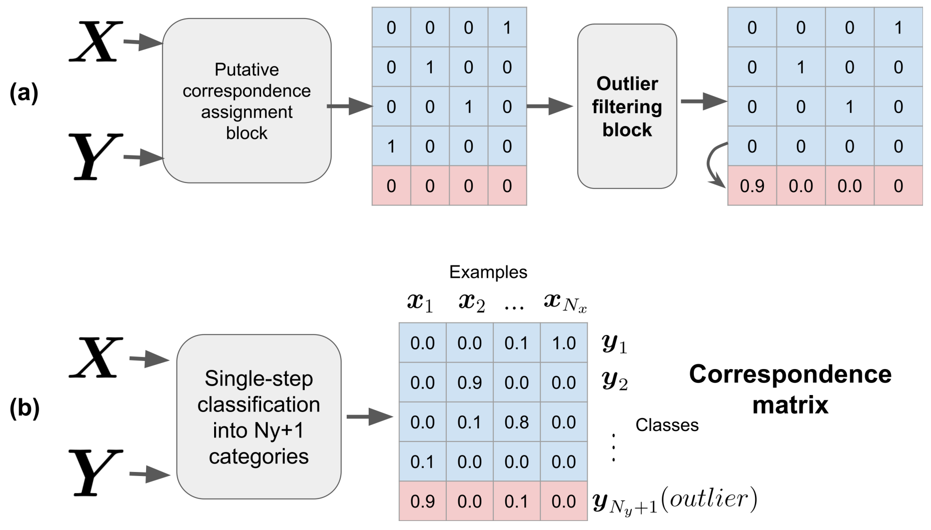

(a)
X
Y
Putative correspondence assignment block
Outlier filtering block
(b)
Single-step classification into Ny+1 categories
Examples
x_1 x_2 ... x_{N_x}
y_1
y_2
Classes
$y_{N_y+1}(outlier)$
Correspondence matrix

Figure 2.3: a) Deep Global Registration [2], b) Possible extension of our approach where two stage process of DGR can be merged into a single step thus resulting in a faster registration. Note that in our current work, networks output $\boldsymbol{C} \in \mathbb{R}^{N_y \times N_x}$ we assume that the data can be partial but there are no outliers

2.3 Experimental setup

We consider RPMNet [35], DCP [1], and PCRNet [17] to study the effects of training the network to learn correspondence vs training the network to learn pose parameters. Note that these methods were originally developed to register point clouds with small ($\pm 45°$) 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 *method*s and *method_corr*s on ModelNet40 [40] dataset.

DCP [1] and RPMNet [35] generate an implicit correspondence based on the similarity between per-point features of the input point clouds (Fig. 2.4). This intermediate correspondence is used to find the transformation parameters between input point clouds using Horn's method [21] and weighted SVD method [35] 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. 2.2.

We sample n number of points from a point cloud chosen from training data and denote this as point cloud $\boldsymbol{X}$. We generate a copy of $\boldsymbol{X}$ and shuffle the order of points to generate $\boldsymbol{Y'}$. To sample a rotation, we randomly choose a unit vector in $\mathbb{R}^3$ and an angle $\theta \in \mathcal{U}(-\theta_0, \theta_0)$, this axis and angle is used to generate a rotation vector which is further transformed into a ground-truth rotation matrix $\boldsymbol{R}^*$. Here, θ_0 depends upon the specific experiment and $\mathcal{U}(a, b)$ denotes a uniform distribution in the range $[a, b]$. Further we generate a ground-truth translation vector $\boldsymbol{t^*} \in [\mathcal{U}(-0.5, 0.5), \mathcal{U}(-0.5, 0.5), \mathcal{U}(-0.5, 0.5)]$. Now $\boldsymbol{Y'}$ is transformed with $\boldsymbol{R^*}$ and $\boldsymbol{t^*}$ to generate $\boldsymbol{Y}$. To generate the ground-truth correspondence matrix $\boldsymbol{C^*}$, we find the nearest neighbour of each $\boldsymbol{x}_i \in \boldsymbol{X}$ in $\boldsymbol{Y'}$. If $\boldsymbol{y'}_j \in \boldsymbol{Y}$ is the nearest neighbour of $\boldsymbol{x}_i$ then $\boldsymbol{C}^*(j, i)$ 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 ($\boldsymbol{X}$). We then randomly choose either up or down directtion of the plane and remove a predetermined number of points from the source farthest from the plane.

2.3.1 DCP Vs DCP_corr

DCP uses DGCNN [37] 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 $\boldsymbol{C}$. They further calculate an intermediate representation of target point cloud $\boldsymbol{Y}$ as $\hat{\boldsymbol{Y}} = \boldsymbol{C}\boldsymbol{Y}$. DCP uses Horn's method to estimate the rotation matrix $\boldsymbol{R}$ and translation $\boldsymbol{t}$ which minimizes the distance between corresponding points of $\hat{\boldsymbol{Y}}$ and $\boldsymbol{X}$. The loss for DCP is defined as

$$L_{DCP} = ||\boldsymbol{R}^T \boldsymbol{R}^* - \boldsymbol{I}||_2^2 + ||\boldsymbol{t} - \boldsymbol{t}^*||_2^2 \tag{2.1}$$

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

$$L_{DCP\_corr} = cross\ entropy(\boldsymbol{C}, \boldsymbol{C^*}) \tag{2.2}$$

2.3.2 RPMNet vs RPMNet_corr

RPMNet follows an iterative procedure. In each iteration, the point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$ 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 [41] normalization layer in an unsupervised manner. In order to estimate the transformation parameters, the target points $\boldsymbol{Y}$ are weighted with the correspondence matrix weights $\boldsymbol{C}$ to obtain putative source correspondences $\hat{\boldsymbol{Y}} = \boldsymbol{YC}$. RPMNet_corr uses this correspondence matrix to define the cross entropy loss (see Fig. 2.4). RPMNet evaluates transformation parameters $\boldsymbol{R}, \boldsymbol{t}$ based on $\boldsymbol{X}, \hat{\boldsymbol{Y}}$ and $\boldsymbol{C}$. These transformation parameters are then used to define the primary loss function L_{reg},

$$L_{reg} = \frac{1}{N_x} \sum_{i=1}^{N_x} |(\boldsymbol{R}^* \boldsymbol{x}_i + \boldsymbol{t}^*) - (\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t})|_1 \tag{2.3}$$

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 [35]), except for the learning rate. RPMNet_corr is trained with an initial learning rate of 0.01 which decays upto 0.0001 during training. We tried a higher learning for both the methods but training of RPMNet is unstable for higher learning rates.

2.3.3 PCRNet Vs PCRNet_Corr

PCRNet is a correspondence-free network that estimates registration parameters given a pair of input point clouds ($\boldsymbol{X}$ and $\boldsymbol{Y}$). As shown in Fig. 2.4, PCRNet uses PointNet [33] 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 ($\in \mathbb{R}^{1x1024}$). 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,

$$\begin{aligned} CD(\boldsymbol{X}, \boldsymbol{Y}) = & \frac{1}{N_x} \sum_{\boldsymbol{x}_i \in \boldsymbol{X}} \min_{\boldsymbol{y}_j \in \boldsymbol{Y}} \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2 + \\ & \frac{1}{N_y} \sum_{\boldsymbol{y}_j \in \boldsymbol{Y}} \min_{\boldsymbol{x}_i \in \boldsymbol{X}} \|\boldsymbol{y}_j - \boldsymbol{x}_i\|_2 \end{aligned} \tag{2.4}$$

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

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

2.4 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 2.1: Effect of initial misalignment on registration accuracy

Rotation	Rotation MAE (deg)		Correspondence (%)	
range (deg)	DCP	DCP_corr	DCP	DCP_corr
0-30	0.99	0.005	8.90	99.99
30-60	1.55	0.008	6.12	99.97
60-90	1.69	0.010	5.78	99.96
90-120	1.56	0.010	5.69	99.96
120-150	1.62	0.010	5.66	99.95
150-180	1.64	0.010	5.60	99.96

2.4.1 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. 2.5 shows the comparison between DCP and DCP_corr. The first column shows that every training procedure converged. Second show 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. 2.5 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. 2.8. 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. 2.3 . 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 $[-45°, +45°]$. We observe that both the networks converge, and the rotation accuracy of DCP and DCP_corr are $1.036°$ and $0.034°$ 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 2.1). 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 remarkably high percentage of correct correspondence.

2.4.2 RPMNet Vs RPMNet_corr

We present the comparisons between RPMNet and RPMNet_corr in Fig. 2.6. 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 [35]. Likewise, we present the Chamfer distance (CD) between registered point clouds, in the fourth column of Fig. 2.6, as suggested by the authors of RPMNet [35]. 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 $0.059°$ with RPMNet_corr as compared to an error of $0.56°$ 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 $0.34°$ compared to $3.79°$ of RPMNet.

Experiment 2.3 RPMNet is specifically designed for $[-45°, +45°]$ 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 $[-180°, +180°]$ 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 2.2 that the MAE for rotation is always lower for RPMNet_corr when compared to RPMNet.

Table 2.2: Effect of initial misalignment on registration accuracy

Rotation	Rotation MAE (deg)		Correspondence (%)		Chamfer distance MSE× 1E-5	
range (deg)	RPMNet	RPMNet_corr	RPMNet	RPMNet_corr	RPMNet	RPMNet_corr
0-30	0.52	0.011	26.96	98.28	8.51	0.56
30-60	0.58	0.013	26.97	98.28	8.48	0.56
60-90	0.69	0.015	26.96	98.28	8.51	0.55
90-120	0.89	0.26	26.96	98.18	8.49	0.68
120-150	1.01	0.47	26.97	98.15	9.36	0.12
150-180	0.96	0.66	26.97	98.07	8.68	0.79

2.4.3 PCRNet Vs PCRNet_corr

The results showing the comparison between PCRNet and PCRNet_corr are shown in 2.7. 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 [17] 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 $\approx 5°$ 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 $[-45°, 45°]$. We observe that PCRNet converges to a rotation accuracy of 9.97° compared to 1.8° 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° at the end of 250 epochs.

2.5 Extending cross entropy to filter outliers

2.5.1 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 $\boldsymbol{F_X} = [\boldsymbol{f}_{x_1}, \boldsymbol{f}_{x_2}, ..., \boldsymbol{f}_{x_{N_x}}] \in \mathbb{R}^{N_e \times N_x}$ and $\boldsymbol{F_Y} = [\boldsymbol{f}_{y_1}, \boldsymbol{f}_{y_2}, ..., \boldsymbol{f}_{y_{N_y}}] \in \mathbb{R}^{N_e \times N_y}$ for input point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$. Here $\boldsymbol{f}_j \in \mathbb{R}^{N_e \times 1}$ and N_e 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

$$\boldsymbol{C} = softmax(\boldsymbol{F}_{\boldsymbol{Y}}^T \boldsymbol{F_X}) \in \mathbb{R}^{N_y \times N_x}$$

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 $\boldsymbol{f_{Y_O}} \in \mathbb{R}^{N_e \times 1}$ that can suitably represent an outlier. With such embedding for outliers, we can predict an outlier variant of correspondence matrix $\boldsymbol{C}^O \in \mathbb{R}^{N_y+1 \times N_x}$ as

$$\boldsymbol{C}^O = softmax([\boldsymbol{f}_{y_1}, \boldsymbol{f}_{y_2}, ..., \boldsymbol{f}_{y_{N_y}}, \boldsymbol{f_{Y_O}}]^T \boldsymbol{F_X})$$

Note that for i^{th} source point, $\boldsymbol{C}^O_{1,i}, \boldsymbol{C}^O_{2,i}, ..., \boldsymbol{C}^O_{N_y,i}$ denotes the probability of the point belonging to $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{N_y}$ respectively and $\boldsymbol{C}^O_{N_y+1,i}$ 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 $\boldsymbol{f_{Y_O}}$ as,

$$\boldsymbol{f_{Y_O}} = \underset{\boldsymbol{f}}{argmin} \left(||\boldsymbol{F_Y}^T \boldsymbol{f} - b\mathbf{1}_{N_y}||_2 \right)$$

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

2.6 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 $[-45°, +45°]$ for both DCP and DCP_corr. Then position of 10% points from $\boldsymbol{X}$ 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 $\boldsymbol{C^{O*}}$, we find the nearest neighbour of each $\boldsymbol{x}_i \in \boldsymbol{X}$ in $\boldsymbol{Y'}$ and it's distance to the nearest neighbor. If d_i is less than a predefined threshold then $\boldsymbol{y'}_j \in \boldsymbol{Y}$ is denoted as nearest neighbour of $\boldsymbol{x}_i$ by setting $\boldsymbol{C}^{O*}(j,i)$ to 1 and other elements of i^{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 $\boldsymbol{C}^{O*}(N_y+1, i)$ to 1 and other elements of i^{th} column to be 0.

Further DCP_corr is trained to minimize the loss function *cross entropy*$(\boldsymbol{C^O}, \boldsymbol{C^{O*}})$ as mentioned in Sec. 4. The last row of $\boldsymbol{C^O}$ 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 1.485° 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 loss on transformation parameters. We observe a rotation error of 6.704° and translation error of 0.519 units. The effect of outliers on DCP vs DCP_corr can be observed in the Fig. 9.

2.7 Conclusion and future work

In this chapter 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 chapter 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 [42] and Sun3d [43]. In addition, future work will involve extended the multi-class classification approach to deal with outliers in the point clouds.

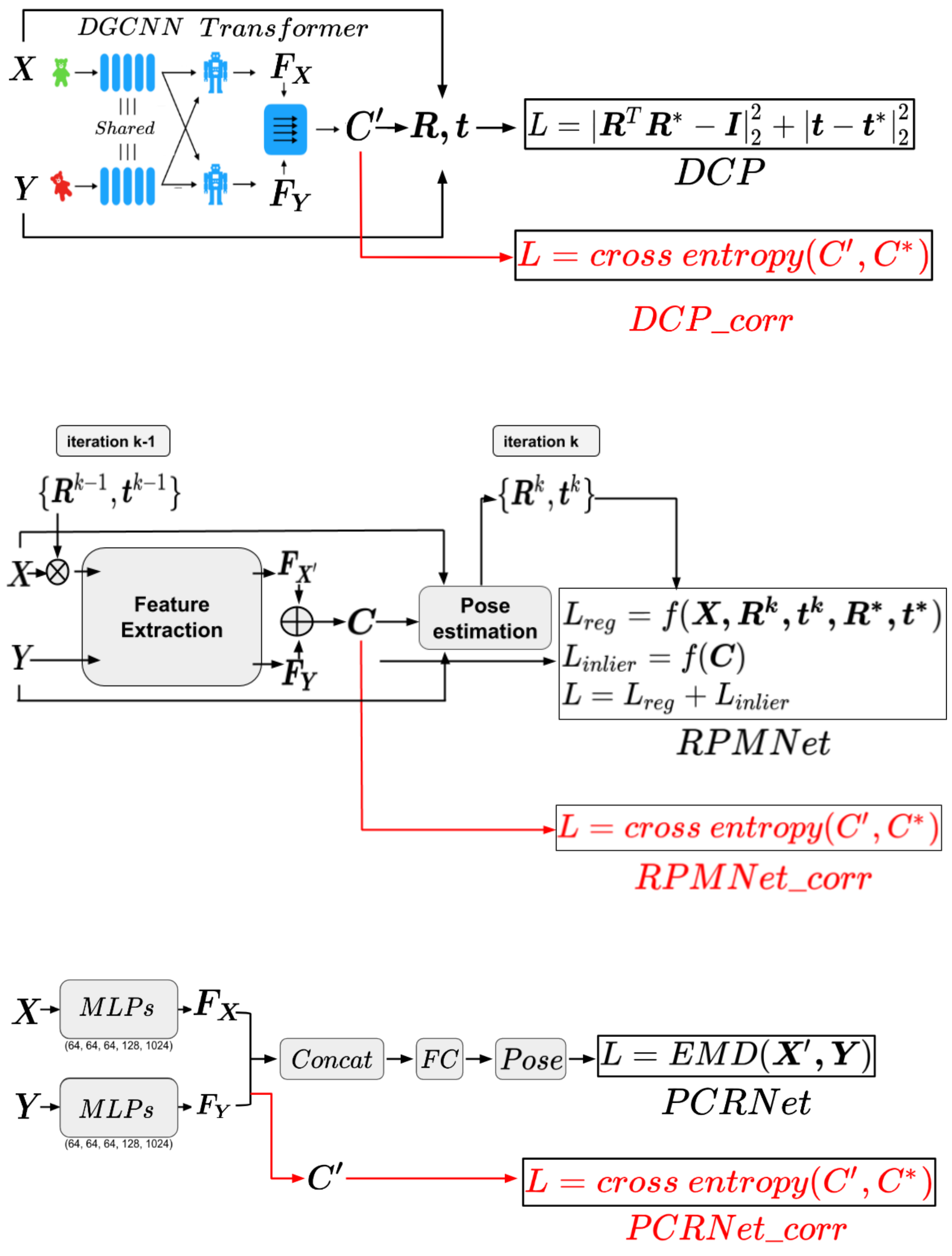


Figure 2.4: DCP and RPMNet architctures internally calculate the correspondence matrix $\boldsymbol{C}$. This correspondence matrix is further used along with $\boldsymbol{X}, \boldsymbol{Y}$ to calculate $\boldsymbol{R}, \boldsymbol{t}$. In order to make these networks explicitly learn correspondence, we use $\boldsymbol{C}$ along with ground truth $\boldsymbol{C^*}$ to calculate cross entropy loss. Since PCRNet does not explicitly calculate $\boldsymbol{C}$, we modify the network architecture and compare the PointNet's per-point features to generate the correspondence matrix.

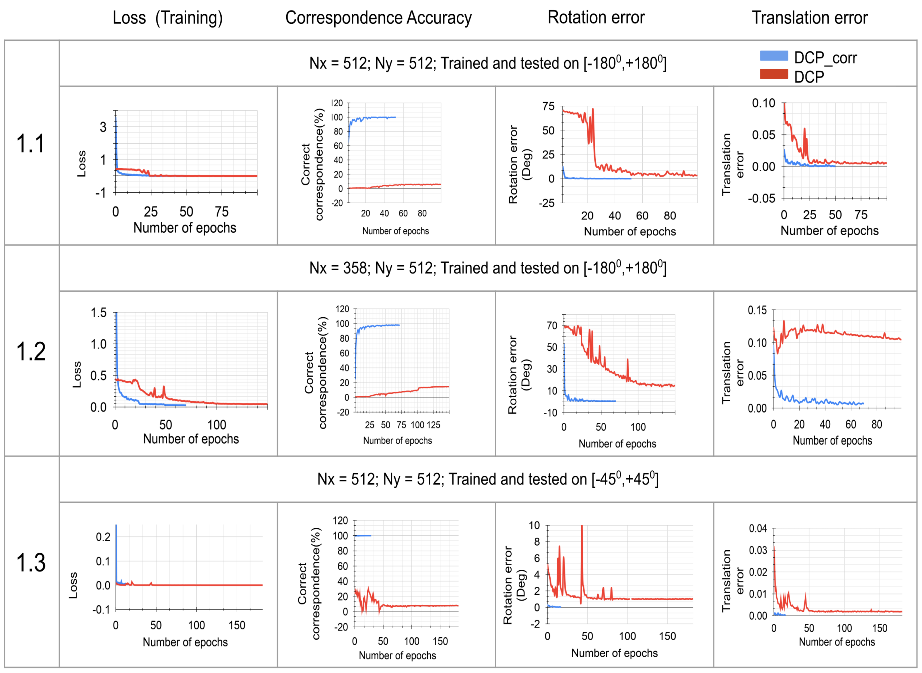


Figure 2.5: Results of experiments on DCP vs DCP_corr.

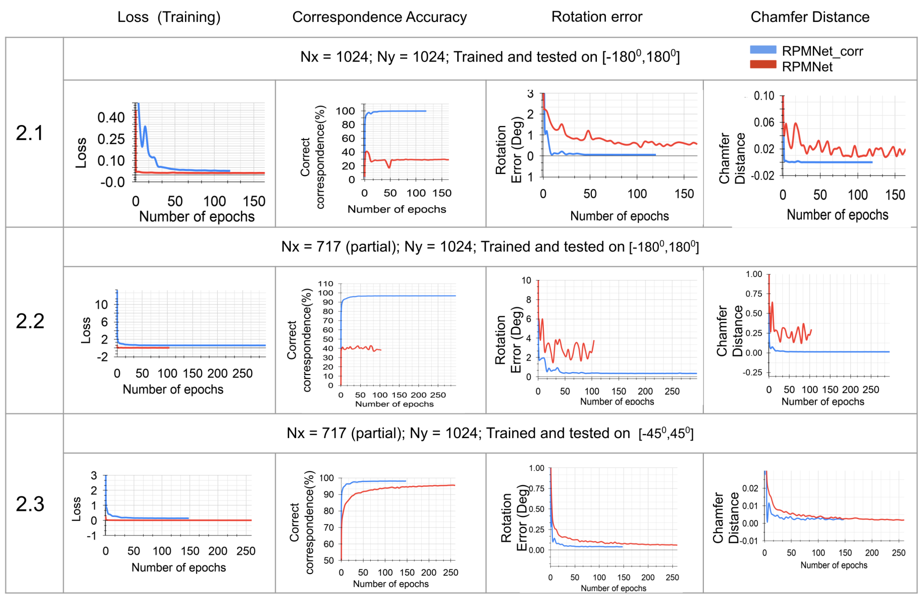


Figure 2.6: Results of experiments on RPMNet vs RPMNet_corr

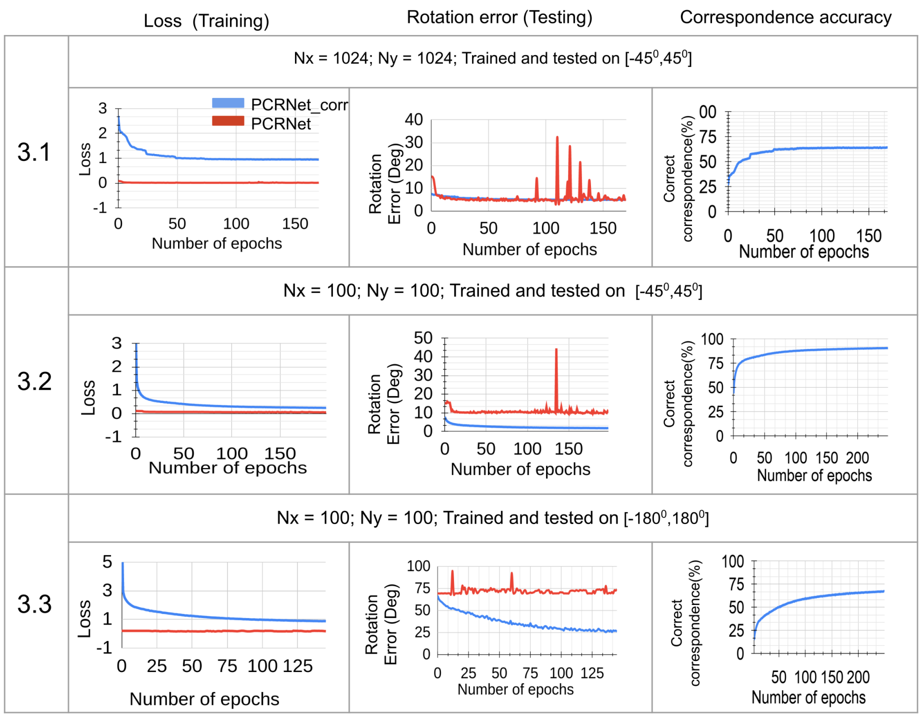


Figure 2.7: Results of experiments on PCRNet Vs PCRNet_corr

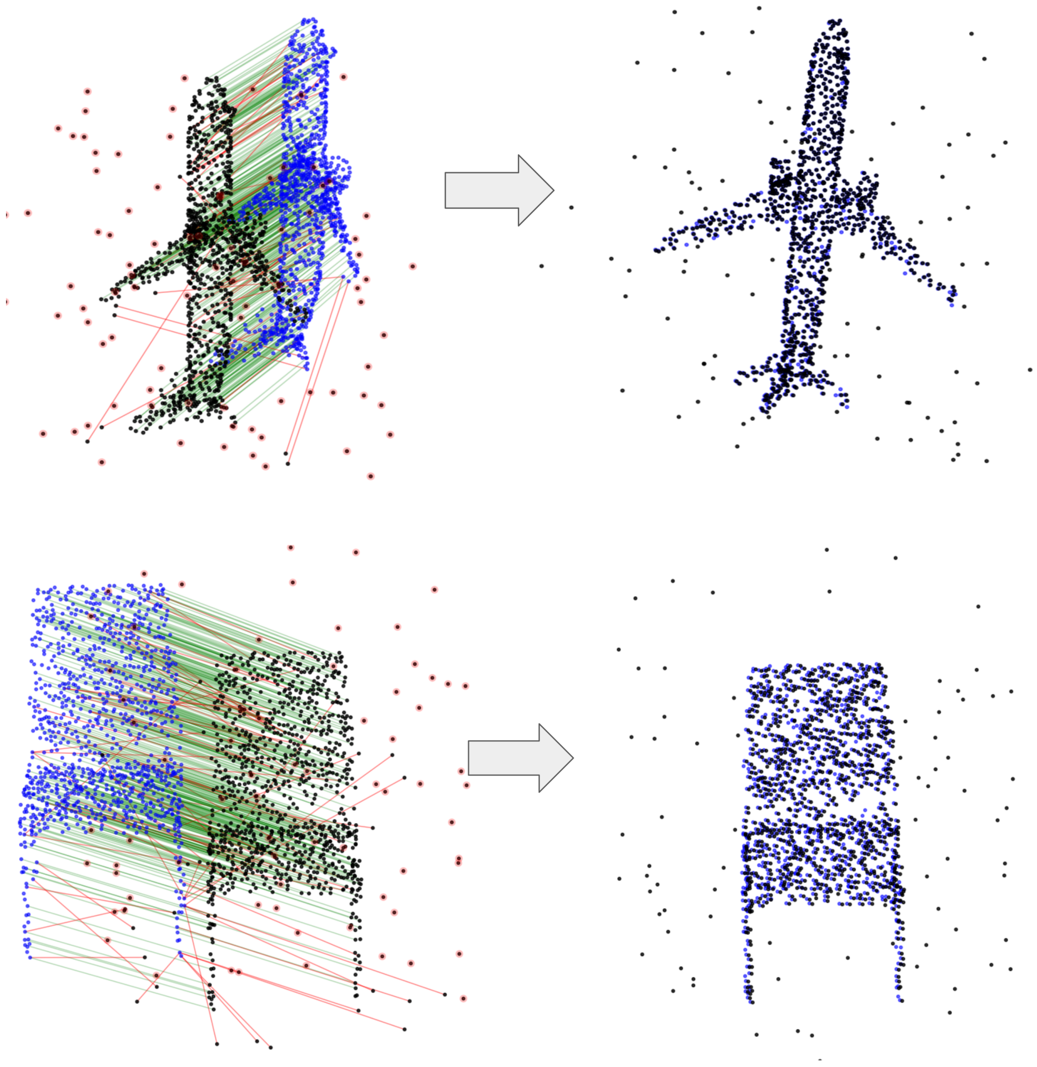


Figure 2.8: Left: Initially misaligned point clouds. Black points are source, blue points are target. Red lines denote correspondence wrongly predicted by DCP_corr while green lines denote correct predictions. Black points circled by red spheres denote points marked as outliers by the network. Right: Registered point clouds using predicted correspondence matrix.

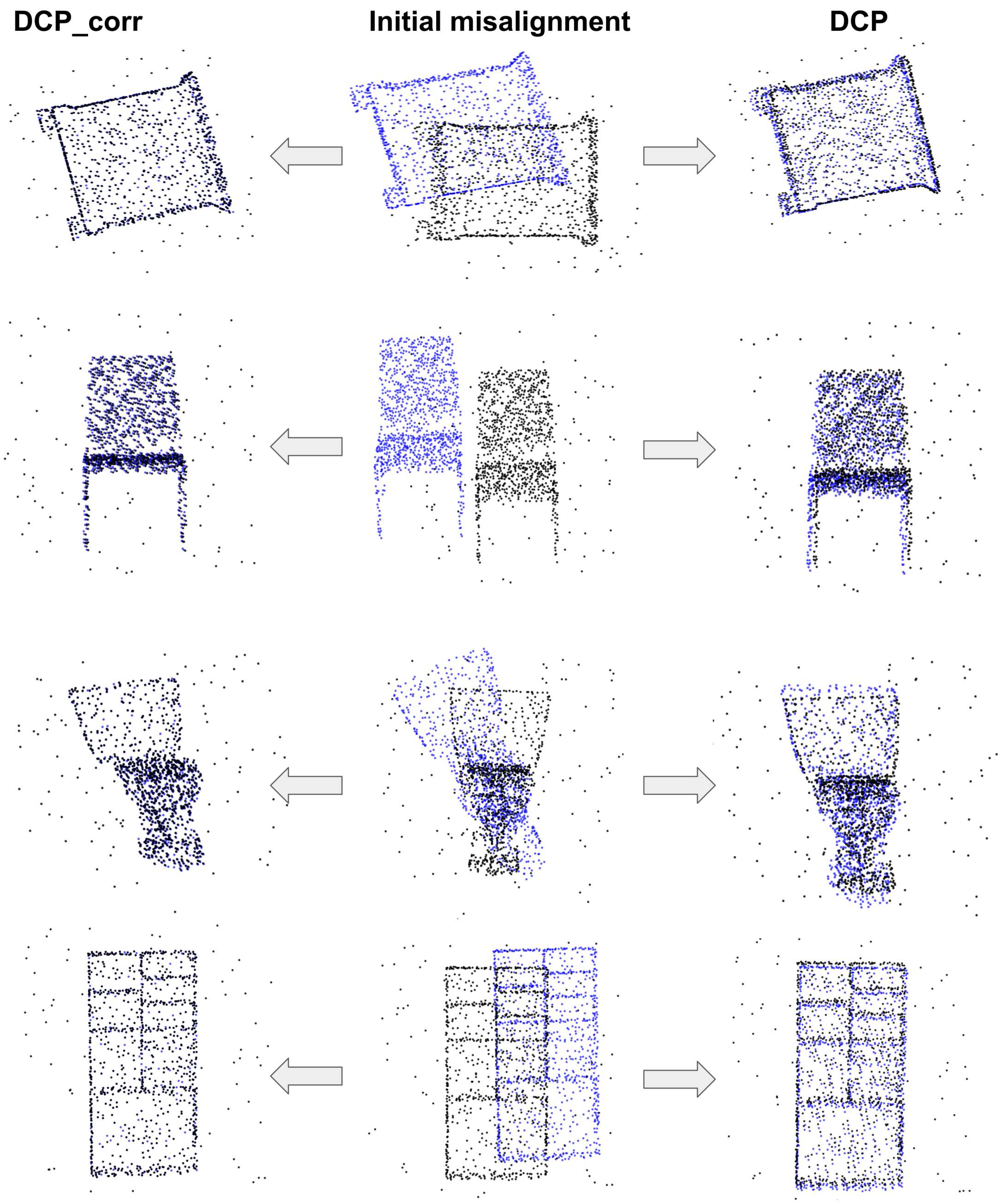


Figure 2.9: Figure 9. Visualization of DCP_corr and DCP registration in presence of outliers

Chapter 3
Outlier filtering with maximum discrepancy loss

3.1 Introduction

When registering a pair of point clouds we often encounter points in once point cloud that do not have a corresponding point in the other. For example when we are trying to register a real world point cloud of a chair with CAD model of the chair the real world point clouds will have points belonging to the floor and other objects in the scene. These points that do not correspond to points on the CAD model are known as outliers. Incorrect matching of outliers leads to erroneous registration, thus outliers need to be filtered out before registration. To deal with this outlier filtering problem, we start with an observation that the classification community deals with a similar problem.

Typically, a classification neural network is trained to classify the input data into a fixed preset categories. The prediction of the network (or classifier) is unpredictable if the test data does not belong to the any of the preset categories. We thus want to predict if the test data does not belong to the training data distribution. This problem is typically known as out of distribution detection. For example, consider classifier trained to classify dogs vs cat images. If such classifier is tested on a bird image, by itself it will predict it as a cat or dog. One approach developed to filter out these out of distribution images is discrepancy loss. This chapter shows how the idea of discrepancy loss from image classification can be applied in the context of point cloud registration.

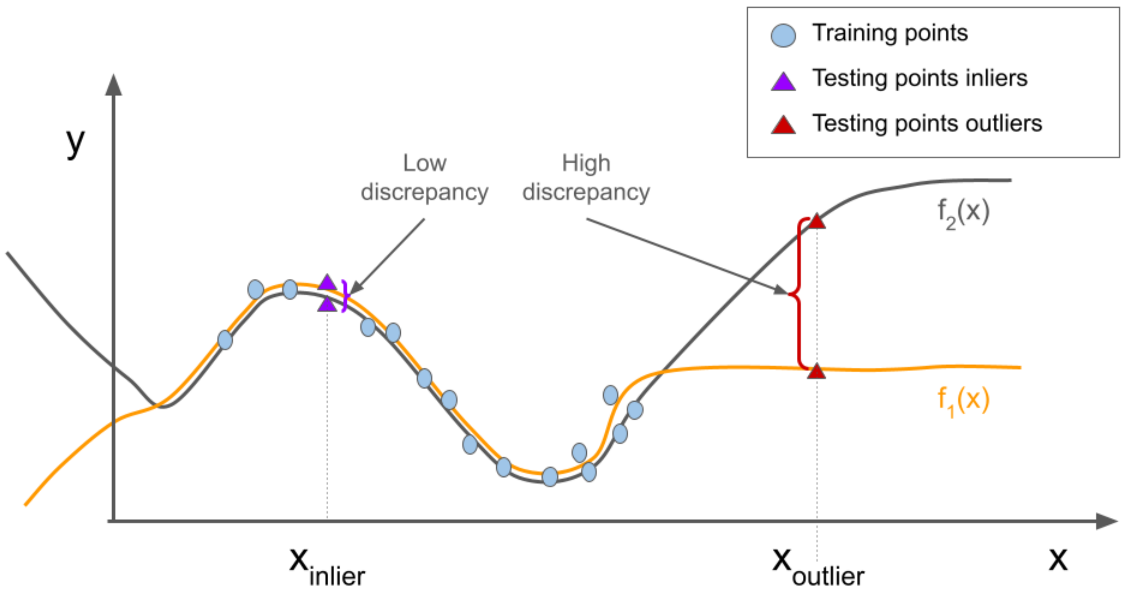
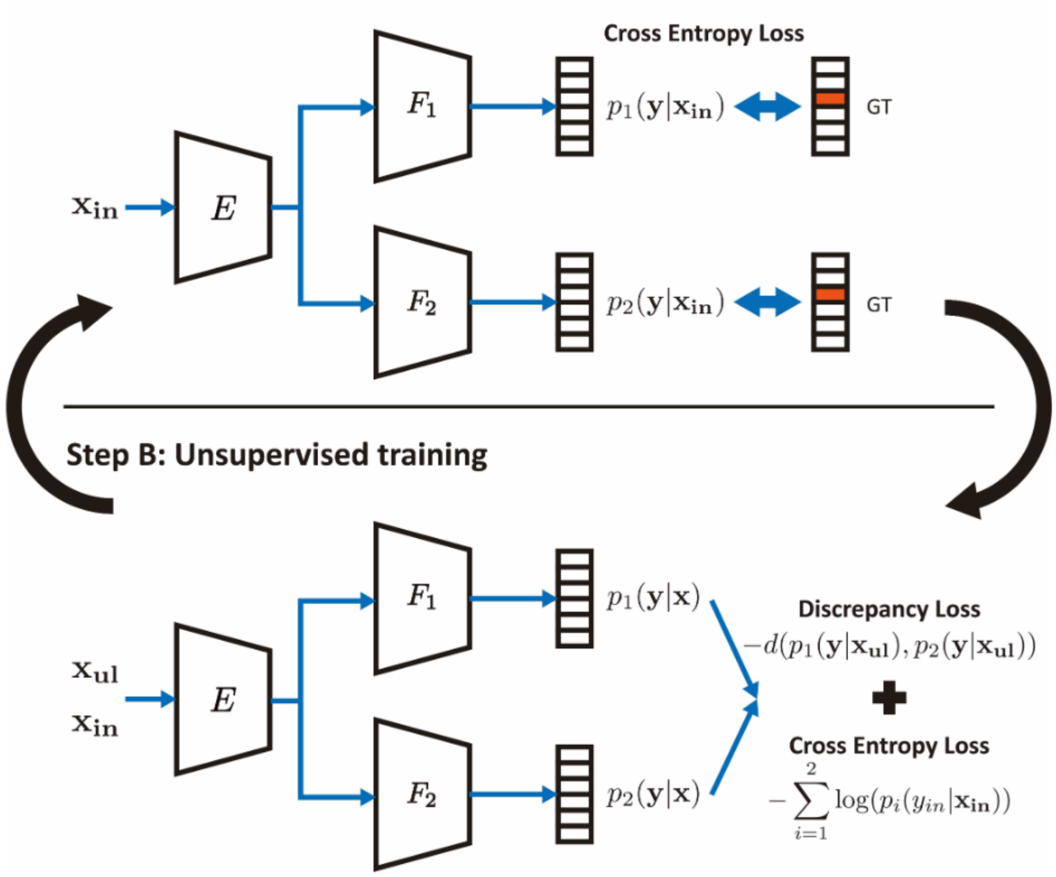


Figure 3.1: Discrepancy

3.2 What is discrepancy?

The idea of discrepancy can be explained with a toy problem of polynomial fitting. Polynomial fitting is generally treated as a local optimization problem minimizing a cost function based on a series of data points. Consider a data points or training points as shown in Fig. 3.1. If we try to regress two functions $f_1(x)$ and $f_2(x)$ to this data, starting from different initial seeds, there shapes will be similar near the inilier region. Since the rest of the regions do not contribute to the cost function, the shape of the fitted functions can be totally different from each other. Thus we can use discrepancy or the difference $\Delta f(x) = f_1(x) - f_2(x)$ to differentiate between inlier and outlier region. We need to choose initial conditions or local optimization approaches such that $|\Delta f(x_{\text{inlier}})| << |\Delta f(x_{\text{outlier}})|$ for easy outlier filtering.



Cross Entropy Loss
$\mathbf{x_{in}}$
E
F_1
F_2
$p_1(\mathbf{y}|\mathbf{x_{in}})$
$p_2(\mathbf{y}|\mathbf{x_{in}})$
GT
GT
Step B: Unsupervised training
$\mathbf{x_{ul}}$
$\mathbf{x_{in}}$
E
F_1
F_2
$p_1(\mathbf{y}|\mathbf{x})$
$p_2(\mathbf{y}|\mathbf{x})$
Discrepancy Loss
$-d(p_1(\mathbf{y}|\mathbf{x_{ul}}), p_2(\mathbf{y}|\mathbf{x_{ul}}))$
Cross Entropy Loss
$-\sum_{i=1}^{2} \log(p_i(y_{in}|\mathbf{x_{in}}))$

Figure 3.2: Discrepancy training procedure. Image credits Qing and Kiyoharu [3]

3.3 Maximum discrepancy loss

Qing and Kiyoharu [3] describe maximum discrepancy loss and a training procedure for this loss in the context of out of distribution image detection. As shown in Fig. 3.2, they have a feature extractor E. The extracted features are modified by the network blocks $F1$ and $F2$. The blocks $F1$ and $F2$ are trained so that class prediction has high discrepancy for out of distribution data. Here the total loss is defined as

$$L = L_{sup} + L_{unsup} \tag{3.1}$$

$$L_{sup} = -\frac{1}{|X_{in}|} \sum_{\mathbf{x}_{in} \in X_{in}} \sum_{i=1}^{2} log(p_i(y_{in}|\mathbf{x}_{in})) \tag{3.2}$$

$$L_{unsup} = \max(0, m - \frac{\sum_{\mathbf{x}_{ul} \in X_{in}} d(p_1(\mathbf{y}|\mathbf{x_{ul}}), p_2(\mathbf{y}|\mathbf{x_{ul}}))}{|X_{ul}|}) \tag{3.3}$$

The supervised loss L_{sup} is just cross-entropy that brings functions F_1 and F_2 closer for in distribution data. While L_{unsup} pushes functions apart if they are close to each other than a threshold m. Here the discrepancy loss is defined as,

$$d(p_1(\mathbf{y}|\mathbf{x_{ul}}), p_2(\mathbf{y}|\mathbf{x_{ul}})) = H(p_1(\mathbf{y}|\mathbf{x_{ul}}) - p_2(\mathbf{y}|\mathbf{x_{ul}})) \tag{3.4}$$

Where $H(.)$ defines entropy over the softmax distribution. Reader is suggested to refer to [3] for more details on using maximum discrepancy loss for out of distribution detection for image classification.

3.4 Network Architecture for outlier filtering in point clouds

Our approach is outlined in Figure 3.3. We first pass the source and target point clouds through a feature extractor such as DCP, DGCNN or RPMNet. We then pass the extracted source features through two separate classifiers $\boldsymbol{FC_1}$ and $\boldsymbol{FC_2}$. However, the target features are passed only through a single classifier $\boldsymbol{FC_{target}}$. This gives us two different source representations and one target representation. We matrix multiply the source and target representations (as in DCP) to generate two separate correspondence matrices $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$. The discrepancy between them gives us the information whether the corresponding source point is an inlier or not. Finally, weighing the target point clouds according to any one of the correspondence matrix gives us the source-target pairs of inlier point clouds, which can then be used for registration using Horn's method.

3.5 Discrepancy loss for PCR

During training, we supervise the network using two kinds of loss functions: correspondence loss and discrepancy loss. The correspondence loss is the same as that defined in Chapter

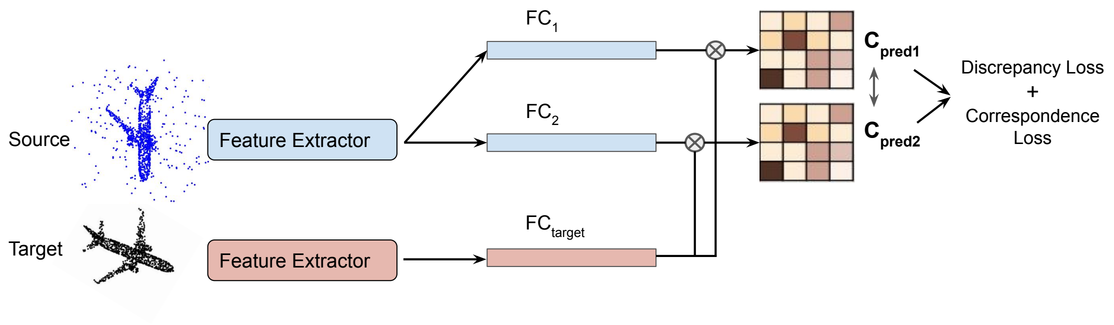


Figure 3.3: Architecture of our outlier filtering approach. The source and target point clouds are fed into a feature extractor (e.g. DCP). The extracted source features are passed into two separate fully connected layers $\boldsymbol{FC_1}$ and $\boldsymbol{FC_2}$ with different initializations. The target features are passed through a single fully connected layer $\boldsymbol{FC_{target}}$. The matrix multiplication of the source and target features gives the correspondence matrices $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$. These are then supervised using the discrepancy and correspondence-based loss functions.

2 Section 2.2. It trains the network to predict the correspondence matrix accurately by comparing it with the ground-truth correspondence matrix. Only inlier source points are passed into this loss function as only they form a valid pair with the target. We use the cross entropy loss for this purpose, whose mathematical formulation is given in Equation (3.5).

$$L_{CE}(\boldsymbol{C_{pred}}, \boldsymbol{C_{gt}}) = -\frac{1}{N_x}\sum_{i=1}^{N_x} log\left(\frac{exp\left(\sum_{j=1}^{N_y} \boldsymbol{C_{pred}}_{j,i}\boldsymbol{C_{gt}}_{j,i}\right)}{\sum_{j=1}^{N_y} exp(\boldsymbol{C_{pred}}_{j,i})}\right) \tag{3.5}$$

where $\boldsymbol{C_{pred}}, \boldsymbol{C_{gt}} \in \mathbb{R}^{N_y \times N_x}$ with N_x being the number of inlier source points and N_y that of target points. Here, $\boldsymbol{C_{pred}}$ is the mean of $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$.

On the other hand, the discrepancy loss is used to maximize the discrepancy between the two predicted correspondence matrices $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$ so that the outliers can be filtered easily. Only outlier source points are passed into this loss function as our aim is to increase discrepancy only for outliers. Its mathematical formulation is given by

$$L_{disc}(\boldsymbol{C_{pred1}}, \boldsymbol{C_{pred2}}) = \max\left(mrg - \frac{1}{N'_x}\sum_{i=1}^{N'_x} disc(\boldsymbol{C_{pred1}}_{:,i}, \boldsymbol{C_{pred2}}_{:,i}), 0\right) \tag{3.6}$$

where N'_x is the number of outlier points in the source and $disc(\cdot)$ is given by

$$disc(\boldsymbol{C_{pred1}}_{:,i}, \boldsymbol{C_{pred2}}_{:,i}) = H(\boldsymbol{C_{pred1}}_{:,i}) - H(\boldsymbol{C_{pred2}}_{:,i}) \tag{3.7}$$

where $H(\cdot)$ is the entropy function over column-wise softmax.

$$H(\boldsymbol{C_{pred}}_{:,i}) = -\sum_{j=1}^{N_y}\left(\frac{exp(\boldsymbol{C_{pred}}_{j,i})}{\sum_{j=1}^{N_y} exp(\boldsymbol{C_{pred}}_{j,i})} * log\left(\frac{exp(\boldsymbol{C_{pred}}_{j,i})}{\sum_{j=1}^{N_y} exp(\boldsymbol{C_{pred}}_{j,i})}\right)\right) \tag{3.8}$$

Here, the margin mrg helps to prevent overfitting because if the average discrepancy of the outlier points is greater than this margin, then we can set loss to zero as the discrepancy is sufficiently large to filter the outliers.

Thus, the total loss function is given by

$$L_{total} = L_{CE}(\boldsymbol{C_{pred}}, \boldsymbol{C_{gt}}) + L_{disc}(\boldsymbol{C_{pred1}}, \boldsymbol{C_{pred2}}) \tag{3.9}$$

To sum up, the correspondence loss forces the two classifiers to learn the same features for inlier points (thus reducing their discrepancy), while discrepancy loss forces the two classifiers to learn different features for outlier points (which increases their discrepancy). Hence, the two losses put together helps us classify the points into inliers vs outliers by looking at the difference between $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$. This working mechanism is illustrated in Figure 3.4

3.6 Training Procedure

The training procedure is divided into two steps:

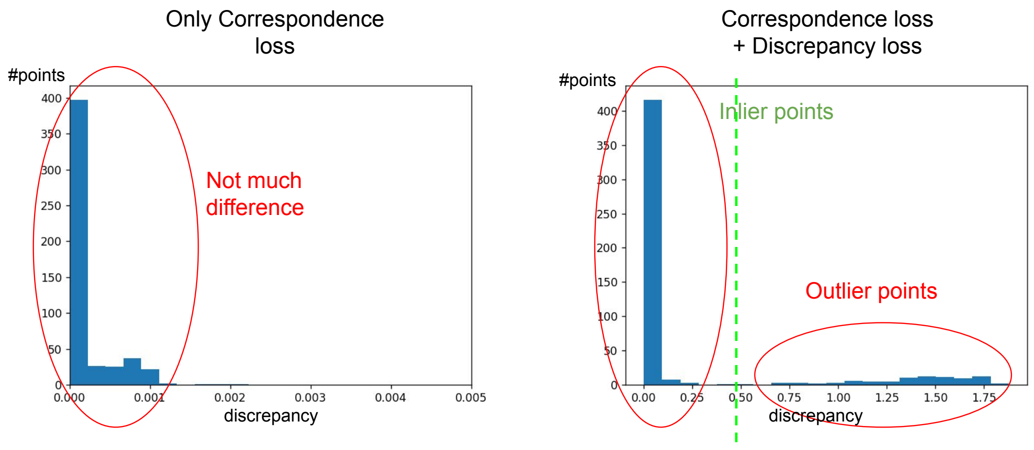
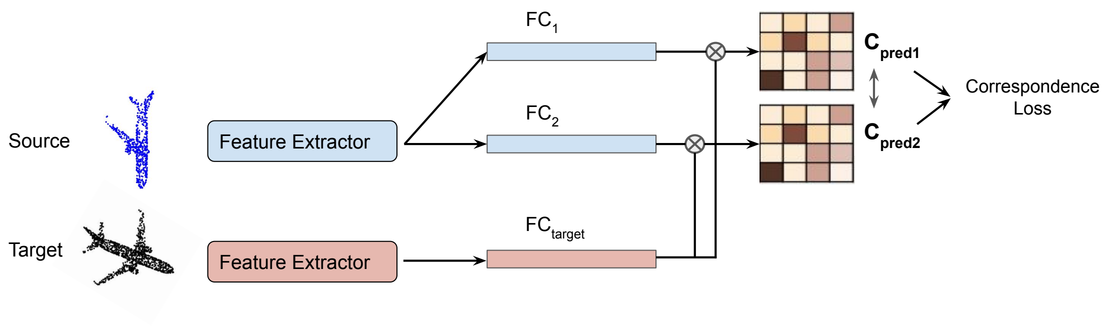


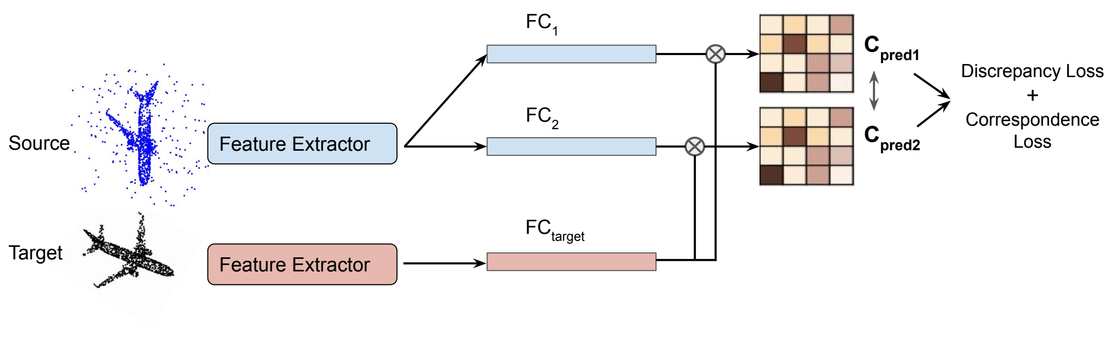
Figure 3.4: Motivation for supervising our network with the discrepancy loss function along with the correspondence loss. Both the graphs are histograms representing the number of points with a certain discrepancy value between the two predictions $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$. Left: The histogram when the network is trained only using the correspondence loss. It can be seen that all the points (inlier and outliers) have negligible discrepancy between the predictions, making it hard to separate the outliers. Right: The histogram when the network is supervised with discrepancy loss in addition to the correspondence loss. The inlier points remain to have low discrepancy, but the outlier predictions are pushed apart to have a high discrepancy. Thus, they can now be easily separated by thresholding at a certain discrepancy value.

Step 1: Pre-training: The full network architecture (feature extractor and the fully connected layers) is trained for registration using the correspondence loss. This training is similar to the one we discussed in Chapter 2. In this step, we train our network to learn point correspondences from intrinsic geometrical properties of the point clouds (extracted by DGCNN) and the cross co-relation between the source and the target points (extracted by transformer). In this training, we only pass inlier point clouds to the network as it learns the similarity between the source and the target points. Here, both the fully connected classifiers $\boldsymbol{FC_1}$ and $\boldsymbol{FC_2}$ will learn similar features without any discrepancy. This is because the point correspondences extracted by the network will be in the inlier distribution space. (Ref. Fig. 3.5a)

Step 2: Fine-tuning: Here, the network (especially the fully connected classifier) is trained to increase the discrepancy between $\boldsymbol{FC_1}$ and $\boldsymbol{FC_2}$ with the supervision of the discrepancy loss. In this step, the outlier points are fed to the network along with the inlier points. Discrepancy loss trains the network to increase the distance between the predictions of $\boldsymbol{FC_1}$ and $\boldsymbol{FC_2}$. This difference is then reflected in the correspondence matrices $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$. The higher this difference, the more is the probability of the point being outlier. By thresholding this difference, we can filter out the outlier points accurately. (Ref. Fig. 3.5b)



(a) Step 1: Pre-training



(b) Step 2: Fine-tuning

Figure 3.5: Training procedure. Top: Pre-training step. Supervision only on the inlier points using the correspondence loss. Bottom: Fine-tuning Step: Supervision on all points using the discrepancy loss along with the correspondence loss.

3.7 Experiments and Results

We evaluated our outlier filtering method against various types of outlier settings and different backbone architectures. All our experiments are conducted on the ModelNet40 [20] dataset.

3.7.1 Different types of outlier settings:

In this experiment, we evaluate our approach against three types of outlier scenarios mentioned in Figure 3.7), which we believe are most commonly found in real-world examples.

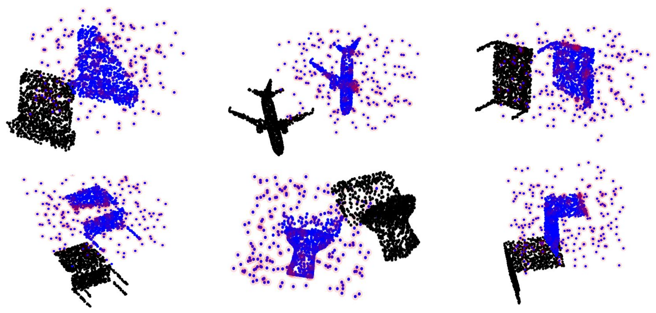
The first scenario (Fig. 3.7a) is the presence of random outliers with partial source points. It occurs when the scanner has a random noise associated with it and some part of the scanned 3D object is occluded. Thus, it results in random outliers present in the scene and the source points are only partially available.

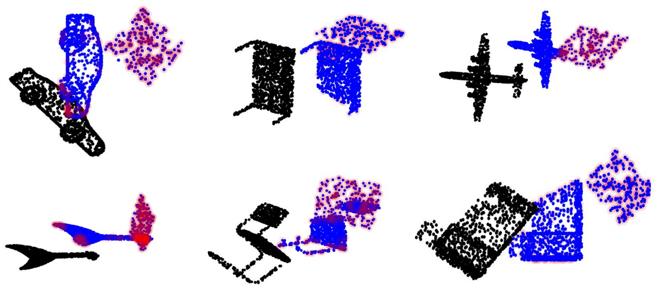
The second scenario (Fig. 3.7b) is a hypothetical case in which a plane is also a part of the scanned source object. (e.g. nearby wall, tabletop, etc). Evaluation on this case will verify that our approach can not only filter out random noise, but also structured outliers such as a plane. In ModelNet40 classes which inherits a planar shape in its structure, such as tables, chairs, laptops, etc. this experiment shows that our model does not confuse between the outlier plane and object's inherent plane, and thus filters the outliers correctly.

The third scenario (Fig. 3.7c) is the most common scenario in real-world, and perhaps the most challenging one. In this case, both the source and the target point clouds are only partially visible. This occurs in real-world autonomous driving scenarios as well as in 3D scanning and stitching. For instance, when a room is scanned from two different angles, only partial parts of the room at overlapping each other. In other words, only partials scans of the room are available in both the source and the target point clouds.

We tested our model on each of the three cases. In these experiments, we fixed the percentage of outlier points out of total number of points to 20%. In the ablation studies, we further vary this percentage to test the robustness of our model.

The Quantitative results on each of these experiments can be found in Table 3.1, while


(a) Random outliers with partial source points


(b) Planar outliers with partial source points

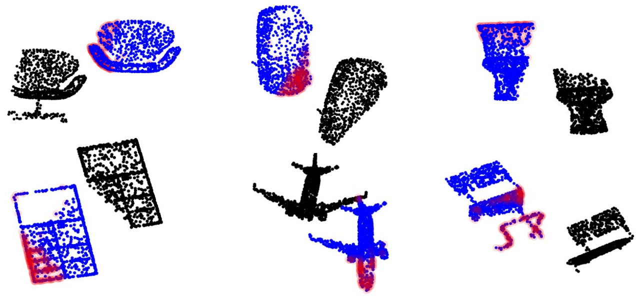

(c) Partial source and partial target points

Figure 3.6: Qualitative results of our outlier filtering technique. Blue: Source points including outliers. Black: Target points. Red Circles: Outliers detected by our method.

the qualitative outlier filtering results are shown in Figure 3.7. The results table shows the registration performance of DCP with and without our outlier filtering approach on all the three scenarios mentioned in Figure 3.7. One can observe that the registration errors drop significantly after adopting our outlier filtering method. From the last column, it is evident that our approach can indeed filter out around 99% of the outliers correctly, thus eliminating all the incorrect correspondence pairs which results in an increased registration

Scenario	**Results**				
	Without Outlier Filtering		With Outlier Filtering		
	Rotation Error (deg)	Translation Error (cm)	Rotation Error (deg)	Translation Error (cm)	% Outliers correctly filtered out
Fig. 3.7a	3.54	0.024	**0.78**	**0.003**	99.2%
Fig. 3.7b	8.31	0.071	**0.15**	**0.0007**	99.5%
Fig. 3.7c	2.52	0.033	**0.11**	**0.0007**	98.6%

Table 3.1: Registration performance of our approach on the three outlier scenarios. First column is the scenario mentioned in Fig. 3.7. Columns 2-3 gives the registration RMSE (rotation and translation error) without applying any outlier filtering technique. Columns 4-5 gives the same after applying our technique. Column 6 gives the % of the outliers that were correctly filtered out by our technique.

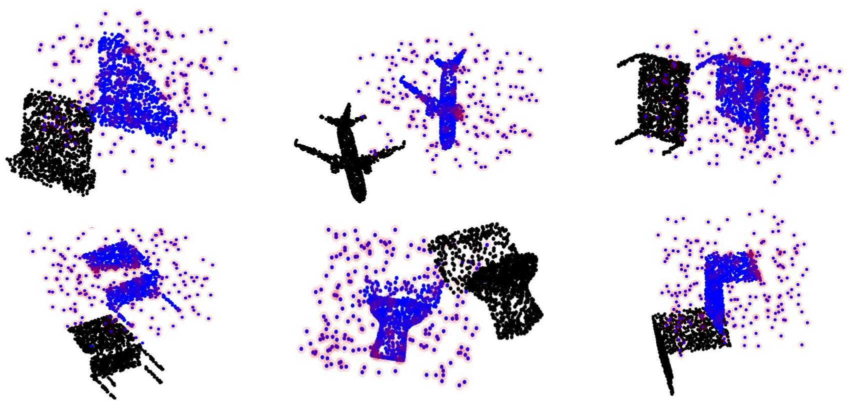
performance. From Figure 3.7, it is evident that our approach can filter outliers effectively in all the different scenarios.

3.7.2 Ablation on varying the discrepancy threshold

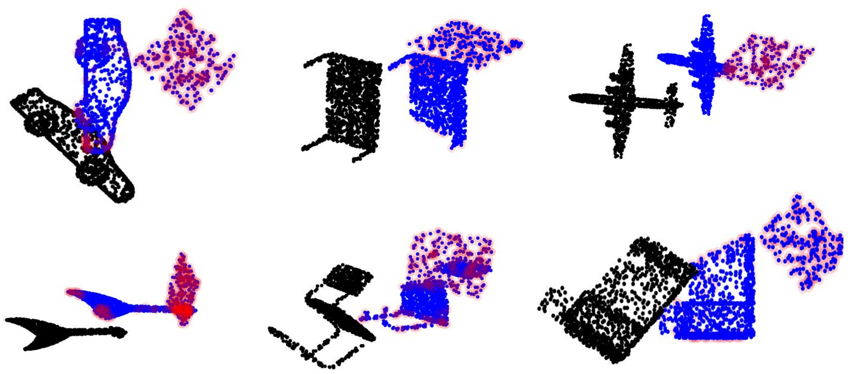
We also analysed the effect of the discrepancy threshold on registration accuracy. It is the threshold value of the distance between two classifier's predictions $\boldsymbol{C_{pred1}}$ and $\boldsymbol{C_{pred2}}$ at which we can separate the outliers from inliers. The results are given in the Table 3.2.

Scenario	$Threshold = 0.05$	$Threshold = 0.15$	$Threshold = 0.25$
Fig. 3.7a	**0.79**	0.90	1.16
Fig. 3.7b	**0.51**	0.53	0.72
Fig. 3.7c	**0.11**	0.25	0.53

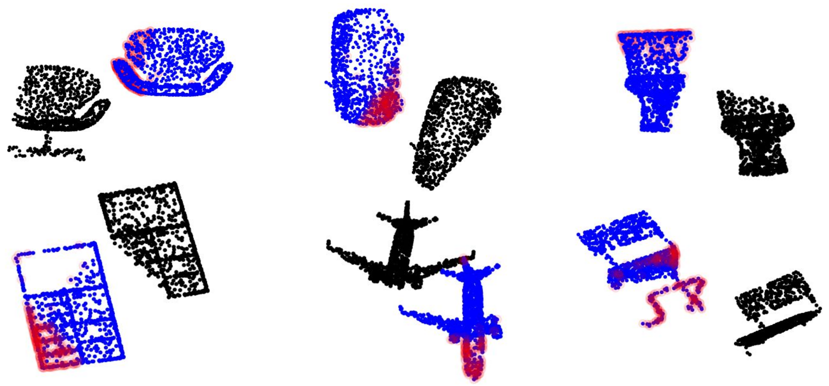
Table 3.2: Ablation study for varying the discrepancy threshold value. First column is the scenario as mentioned in Fig. 3.7. Columns 2-4 gives the rotation error (RMSE) values in degrees for the registration in which the outliers are filtered by the discrepancy threshold values given in the column head.



(a) Random outliers with partial source points



(b) Planar outliers with partial source points



(c) Partial source and partial target points

Figure 3.7: Qualitative results of our outlier filtering technique. Blue: Source points including outliers. Black: Target points. Red Circles: Outliers detected by our method.

As one can observe, $threshold = 0.05$ gave the best registration results. Decreasing threshold further resulted in an error due to insufficient inlier points available for registration in some test cases. Hence, we decided to go with $threshold = 0.05$ in all our experiments.

3.7.3 Ablation on percentage of outliers

In addition to the above experiments, we also conducted ablation study to test our method's sensitivity to the percentage of outlier points. The qualitative and quantitative results for each of the scenarios can be found in Figures 3.8, 3.9, 3.10. As it is observed, for varying percentage of outliers, registration performance after outlier filtering is always better than without filtering. One can also observe that as we increase the number of outliers, the drop in performance is very low in the random outliers scenario and negligible in the planar outliers case, as against the one without outlier filtering which suffers from high drop in performance. In the case of partial-to-partial registration Figure 3.10, drop in performance quite high as the percentage of outliers is increases, but this is mainly because the overlapping source and target portion decreases significantly as we increase outlier percentage (especially in the case of outliers = 40% or 50% where there are hardly any points in the overlapped region.)

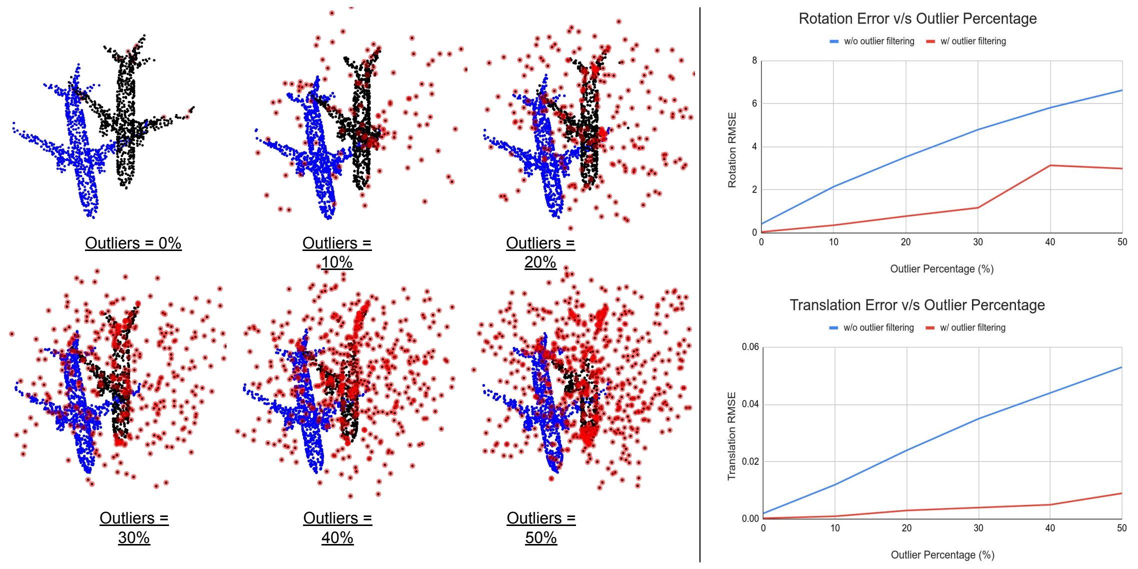


Figure 3.8: Ablation on varying percentage of outliers for Scenario: Random outliers with partial source points

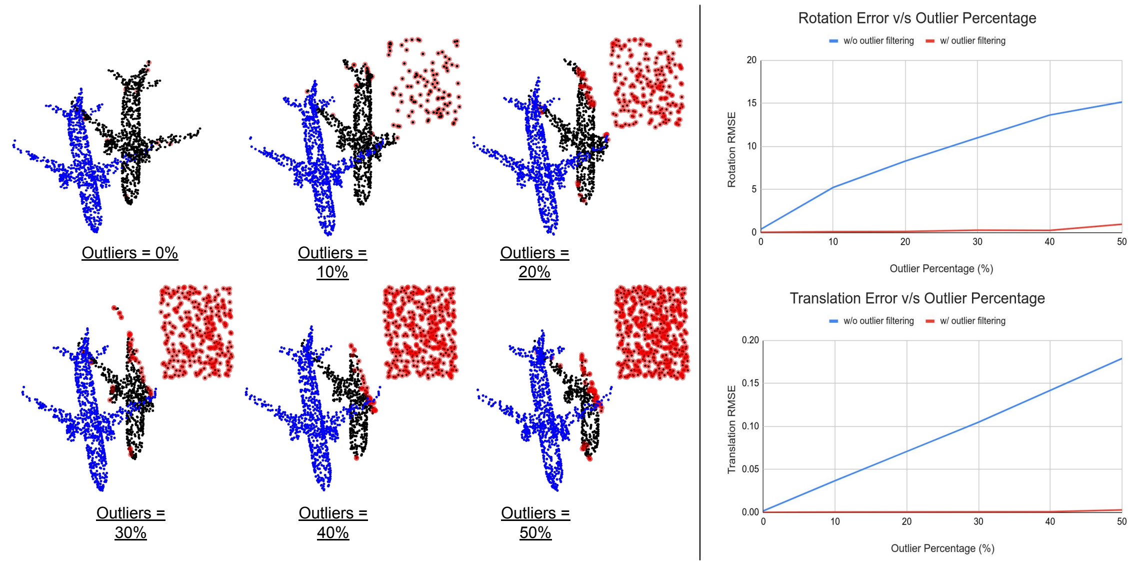


Figure 3.9: Ablation on varying percentage of outliers for Scenario: Planar outliers with partial source points

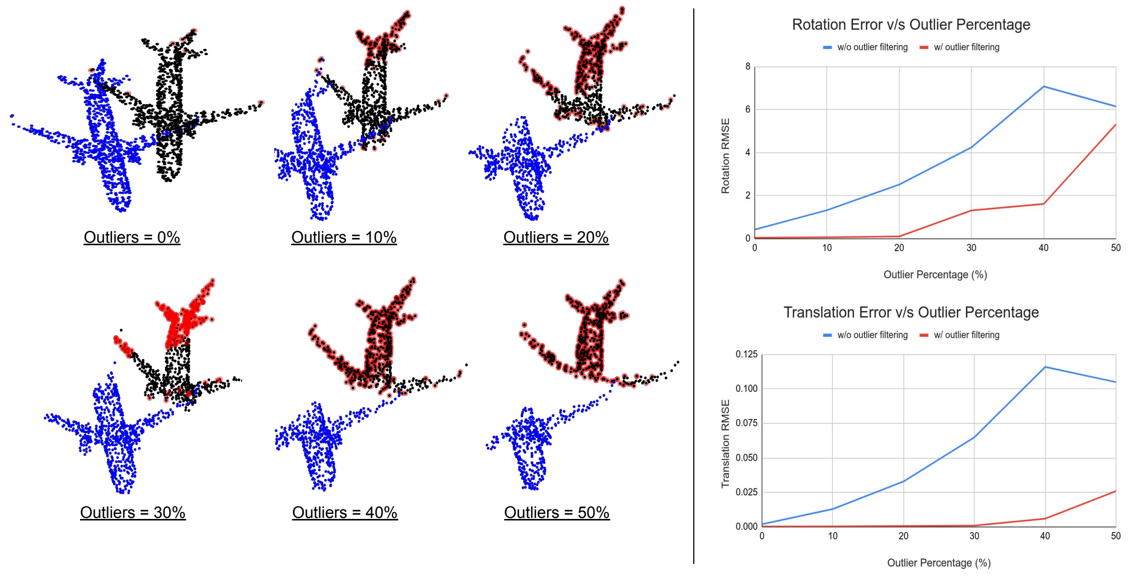


Figure 3.10: Ablation on varying percentage of outliers for Scenario: Partial source and partial target points

3.7.4 Comparison with RPMNet's Sinkhorn Normalization

In this experiment, we conduct an apples-to-apples comparison of our outlier filtering approach with RPMNet's Sinkhorn Normalization approach (discussed in Section **??**). For a fair comparison, since the RPMNet authors implement Sinkhorn normalization on their RPMNet backbone, we also implemented our approach using the same backbone for this experiment.

Method	**Registration Iterations**	**Anisotropic Error**		**Isotropic Error**	
		Rotation MAE (deg)	Translation MAE (cm)	Rotation MAE (deg)	Translation MAE (cm)
RPMNet + Sinkhorn	(5 iter)	0.381	0.0040	0.739	0.0087
RPMNet + Ours	(1 iter)	0.248	0.0021	0.481	0.0043
RPMNet + Ours	(5 iter)	**0.198**	**0.0018**	**0.396**	**0.0038**

Table 3.3: Comparison with RPMNet's Sinkhorn Normalization-based outlier filtering approach. First column states the method used. Column 2 states the number of times registration is applied iteratively by the method (RPMNet's default value is 5 iterations). Columns 3-4 gives the anisotropic registration error as mentioned in the RPMNet paper. Columns 5-6 gives the isotropic registration error.

Table 3.3 shows the comparison of our approach with RPMNet. By default, RPMNet uses 5 registration iterations to register the point clouds accurately. As one can observe, compared to using Sinkhorn normalization for outlier filtering, instead if we use our discrepancy-based approach, we get improved registration performance in just 1 iteration. Needless to say, our 5 iteration version performs even better. Thus, our approach is a 1 shot approach which does not require iterative registration.

3.8 Discussion and future work

Currently, we have limited ourselves to results on ModelNet40, we plan to extend it to real world datasets such as 3DMatch [42] and Sun3d [43].

Other approaches like SuperGlue [44] in the domain of 2D to 2D registration problem, and OPRNet [45] for 3D to 3D PCR, treat the correspondence assignemnt as an optimal transport problem. We would like to study the pros and cons of treating PCR as an optimal transport problem vs a classification problem. Inherently, treating PCR as a classification problem allows us to assign multiple target points to the same source point cloud, while optimal assignment enforces that, one source point is matched to only one target point. We believe that this restriction can hamper the performance of optimal transport approach in case of sparse to dense PCR.

In this thesis, we have demonstrated the application of cross-entropy loss and discrepancy loss. We want to further explore other, more recently developed loss functions such as ring loss [46], LGM loss [47] for classification and KL divergence [48], and MCDD loss [49] for outlier filtering.

So far, we have shown that PCR research can be benefited from the classification research, it would be interesting to see if we can apply some registration literature in the context of classification.

Bibliography

[1] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. *CoRR*, abs/1905.03304, 2019.

[2] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] Qing Yu and Kiyoharu Aizawa. Unsupervised out-of-distribution detection by maximum classifier discrepancy. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9517–9525, 2019.

[4] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[5] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. *CoRR*, abs/1903.05711, 2019.

[6] Aitor Aldoma, Federico Tombari, Radu Bogdan Rusu, and Markus Vincze. Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 113–122. Springer, 2012.

[7] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter 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.

[8] Radu Bogdan Rusu, Andreas Holzbach, Michael Beetz, and Gary Bradski. Detecting and segmenting objects for mobile manipulation. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 47–54. IEEE.

[9] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: Science and Systems*, volume 2, 2009.

[10] K.S. Arun, T.S Huang, and S.D. Bolstein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

[11] Rangaprasad Arun Srivatsan, Prasad Vagdargi, and Howie Choset. Sparse point registration. In *18th International Symposium on Robotics Research*, 2017.

[12] Rangaprasad Arun Srivatsan and Howie Choset. Multiple start branch and prune filtering algorithm for nonconvex optimization. In *The 12th International Workshop on The Algorithmic Foundations of Robotics*. Springer, 2016.

[13] Rangaprasad Arun Srivatsan, Tejas Zodage, and Howie Choset. Globally optimal registration of noisy point clouds. *arXiv preprint arXiv:1908.08162*, 2019.

[14] P.J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[15] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-ICP: Solving 3D Registration Efficiently and Globally Optimally. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 1457–1464, Dec 2013.

[16] Chanki Yu and Da Young Ju. A maximum feasible subsystem for globally optimal 3d point cloud registration. *Sensors*, 18(2):544, 2018.

[17] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. PCRNet: Point Cloud Registration Network using PointNet Encoding. *arXiv e-prints*, page arXiv:1908.07906, August 2019.

[18] Zan Gojcic, Caifa Zhou, Jan Dirk Wegner, and Wieser Andreas. The perfect match: 3d point cloud matching with smoothed densities. In *International conference on computer vision and pattern recognition (CVPR)*, 2019.

[19] G. Hinton, P. Dayan, B. Frey, and R. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268 5214:1158–61, 1995.

[20] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[21] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, 1987.

[22] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[23] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[24] Seth D. Billings, Emad M. Boctor, and Russell H. Taylor. Iterative Most-Likely Point Registration (IMLP): A Robust Algorithm for Computing Optimal Shape Alignment. *PLoS ONE*, 10, 2015.

[25] Lena Maier-Hein, Thiago R dos Santos, Alfred M Franz, and Hans-Peter Meinzer. Iterative closest point algorithm in the presence of anisotropic noise. *Bildverarbeitung für die Medizin*, 2010:231–235, 2010.

[26] A. Myronenko and Xubo Song. Point Set Registration: Coherent Point Drift. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 32(12):2262–2275, Dec 2010.

[27] Anand Rangarajan, Haili Chui, and James S Duncan. Rigid point feature registration using mutual information. *Medical Image Analysis*, 3(4):425–440, 1999.

[28] Ioannis Stamos and Marius 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.

[29] Luciano Silva, Olga Regina Pereira Bellon, and Kim L Boyer. Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 27(5):762–776, 2005.

[30] Mark P Wachowiak, Renata Smolíková, Yufeng Zheng, Jacek M Zurada, and Adel Said Elmaghraby. An approach to multimodal biomedical image registration utilizing particle swarm optimization. *IEEE Transactions on evolutionary computation*, 8(3):289–301, 2004.

[31] Romeil Sandhu, Samuel Dambreville, and Allen Tannenbaum. Point set registration via particle filtering and stochastic dynamics. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1459–1473, 2009.

[32] Gregory Izatt, Hongkai Dai, and Russ Tedrake. Globally optimal object pose estimation in point clouds with mixed-integer programming. In *Robotics Research*, pages 695–710. Springer, 2020.

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[34] Yue Wang and Justin M Solomon. Prnet: Self-supervised learning for partial-to-partial

registration. In *Advances in Neural Information Processing Systems*, pages 8814–8826, 2019.

[35] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[36] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[37] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[38] Zan Gojcic, Caifa Zhou, Jan D. Wegner, Leonidas J. Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration, 2020.

[39] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8958–8966, 2019.

[40] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

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

[42] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas 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.

[43] Jianxiong Xiao, Andrew Owens, and Antonio 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.

[44] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[45] Zheng Dang, Fei Wang, and Mathieu Salzmann. Learning 3D-3D Correspondences for One-shot Partial-to-partial Registration. *arXiv e-prints*, page arXiv:2006.04523, June 2020.

[46] Yutong Zheng, Dipan K Pal, and Marios Savvides. Ring loss: Convex feature normalization for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5089–5097, 2018.

[47] Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9117–9126, 2018.

[48] Sachin Vernekar, Ashish Gaurav, Taylor Denouden, Buu Phan, Vahdat Abdelzad, Rick Salay, and Krzysztof Czarnecki. Analysis of confident-classifiers for out-of-distribution detection. *arXiv preprint arXiv:1904.12220*, 2019.

[49] Dongha Lee, Sehun Yu, and Hwanjo Yu. Multi-class data description for out-of-distribution detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1362–1370, 2020.