ICP 分两步迭代求解刚性点云配准问题：（1）对空间上最近点对应关系进行硬赋值，然后（2）求出最小二乘刚性变换。基于空间距离的最近点对应的硬赋值对初始刚性变换和噪声/离群点敏感，往往导致ICP收敛到错误的局部极小值。本文提出了RPM网络，一种对初始化不敏感的基于深度学习的刚性点云配准方法。网络使用可微Sinkhorn层和退火算法从空间坐标和局部几何中学习的混合特征中获得点对应的软分配。为了进一步提高配准性能，论文引入二次网络来预测最优退火参数。与某些现有方法不同，RPM网络可以处理缺少的对应关系和部分可见性的点云。实验结果表明，与现有的非深度学习和最新的深度学习方法相比，本文的RPM网络达到了SOTA

RPM-Net: Robust Point Matching using Learned Features

Zi Jian Yew Gim Hee Lee
Department of Computer Science, National University of Singapore
{zijian.yew, gimhee.lee}@comp.nus.edu.sg

Abstract

Iterative Closest Point (ICP) solves the rigid point cloud registration problem iteratively in two steps: (1) make hard assignments of spatially closest point correspondences, and then (2) find the least-squares rigid transformation. The hard assignments of closest point correspondences based on spatial distances are sensitive to the initial rigid transformation and noisy/outlier points, which often cause ICP to converge to wrong local minima. In this paper, we propose the RPM-Net – a less sensitive to initialization and more robust deep learning-based approach for rigid point cloud registration. To this end, our network uses the differentiable Sinkhorn layer and annealing to get soft assignments of point correspondences from hybrid features learned from both spatial coordinates and local geometry. To further improve registration performance, we introduce a secondary network to predict optimal annealing parameters. Unlike some existing methods, our RPM-Net handles missing correspondences and point clouds with partial visibility. Experimental results show that our RPM-Net achieves state-of-the-art performance compared to existing non-deep learning and recent deep learning methods. Our source code is available at the project website[1].

1. Introduction

Rigid point cloud registration refers to the problem of finding the rigid transformation to align two given point clouds with unknown point correspondences. It has applications in many areas of computer vision and robotics, *e.g.* robot and object pose estimation, point cloud-based odometry and mapping, etc. Rigid point cloud registration is a chicken-and-egg problem that requires solving for both unknown point correspondences and rigid transformation to align the point clouds, and is thus commonly known as the simultaneous pose and correspondence problem [17]. Knowledge of either point correspondences or rigid transformation trivializes the problem.

[1] https://github.com/yewzijian/RPMNet

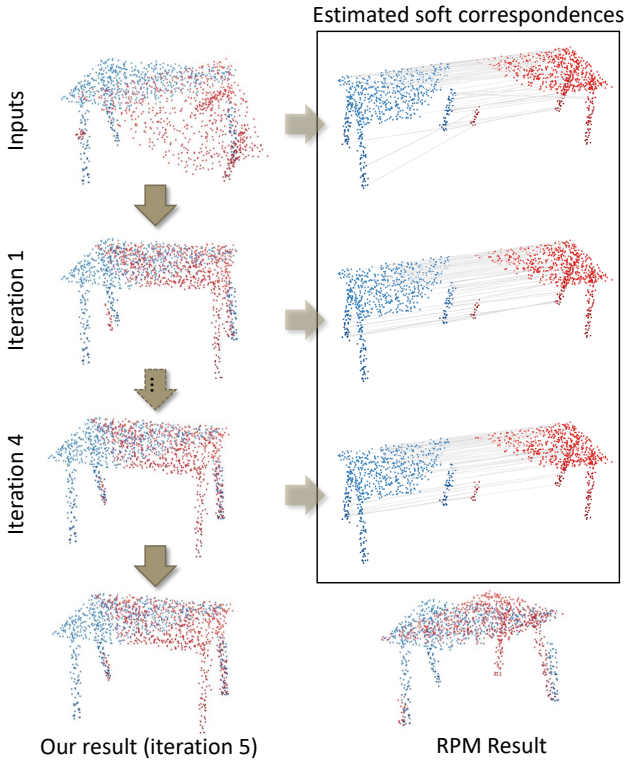

Figure 1. Our RPM-Net estimates soft correspondences from hybrid features learned from both spatial coordinates and local geometry of the points, and converges to the correct solution after 5 iterations. In contrast, RPM [12] gets trapped in a local minima.

ICP [3] is widely accepted to be the de facto algorithm for solving the rigid point cloud registration problem. It solves for both the point correspondences and rigid transformation by alternating between two steps: (1) assigns each point from the reference point cloud to its spatially closest point in the source point cloud, and (2) computes the least-squares rigid transformation between the correspondences. Unfortunately, ICP is highly sensitive to initialization and often converges to wrong local minima. It can also fail in the presence of noisy/outlier points. This limits ICP to relatively low noise and outlier-free scenarios with good initial rigid transforms, and precludes its use in applications such

as registration of noisy scans and global registration. A recent deep learning-based ICP – Deep Closest Point (DCP) [35] computes point correspondences from deep features to desensitize initialization, but remains not robust to outliers and does not work well on partially visible point clouds.

Many works [6, 12, 33] are proposed to mitigate the problems of ICP, and one prominent work is the Robust Point Matching (RPM) [12]. It starts with soft assignments of the point correspondences, and gradually hardens the assignments through deterministic annealing. As we show in the experiments, although RPM is more robust than ICP, it remains sensitive to initialization and local minima as the point correspondences are still obtained solely from spatial distances. On the other hand, feature-based methods [26, 28, 32] avoid initialization and the local minima problem by detecting distinctive keypoints and describing the local geometry of the keypoints using feature descriptors. Feature descriptors in one point cloud can then be matched to those in the other, and the rigid transformation can be solved robustly using a RANSAC scheme. However, these methods only work well for point clouds with distinctive geometric structures [30].

In this paper, we propose a deep learning-based RPM, the RPM-Net: an end-to-end differentiable deep network that preserves robustness of RPM against noisy/outlier points while desensitizing initialization with point correspondences from learned feature distances instead of spatial distances. To this end, we design a feature extraction network to compute hybrid features of each point from its spatial coordinates *and* geometric properties, and then use a Sinkhorn [31] layer and annealing to get soft assignments of the point correspondences from these hybrid features. The fusion of spatial coordinates and geometric properties, and learning from data improve point correspondences. This desensitizes initialization and enhance the ability to register point clouds with missing correspondences and partial visibility. Similar to ICP and most of its variants, our RPM-Net refines rigid point cloud registration iteratively. Furthermore, we introduce a secondary network to predict optimal annealing parameters based on the current state of the alignments, *i.e*. our annealing does not follow a fixed schedule. Together with the use of hybrid features, our algorithm can converge in a small number of iterations as illustrated in the example shown in Figure 1. Experiments show that our RPM-Net achieves state-of-the-art performance compared to existing non-deep learning and recent deep learning methods. Our main contributions are:

- Learn hybrid features with a feature extraction network, Sinkhorn layer and annealing to desensitize initialization and enhance robustness of rigid point cloud registration.
- Introduce a secondary network to predict optimal annealing parameters.
- Suggest a modified Chamfer distance metric to improve measurement of registration quality in the presence of symmetry or partial visibility.
- Show state-of-the-art performance compared to other existing works on experimental evaluations under clean, noisy, and partially visible datasets.

2. Related Work

Feature-Based Methods. Feature-based methods tackle the registration problem in a two-step approach: (1) establish point correspondences between the two point clouds, and (2) compute the optimal transformation from these correspondences. The first step is non-trivial and requires well-designed descriptors to describe distinctive keypoints in order to match them between point clouds. A large variety of handcrafted 3D feature descriptors have been proposed and a comprehensive survey can be found in [13]. Generally, these descriptors accumulate measurements (typically number of points) into histograms according to their spatial coordinates [9, 14, 32], or their geometric attributes such as curvature [4] or normals [28]. To orientate the spatial bins, most of these methods require a local reference frame (LRF) which is hard to obtain unambiguously, so other works *e.g*. PFH [27] and FPFH [26] design rotation invariant descriptors to avoid the need for a LRF. More recent works apply deep learning to learn such descriptors. One of the earliest such works, 3DMatch [42], voxelizes the region around each keypoint and compute descriptors with a 3DCNN trained using a contrastive loss. Voxelization results in a loss of quality, so later works such as PPFNet [7] uses a PointNet [23, 24] architecture to learn features directly from raw point clouds. In addition to predicting feature descriptors, 3DFeat-Net [41] and USIP [18] also learn to detect salient keypoints. The main problem with feature based methods is that they require the point clouds to have distinctive geometric structures. Additionally, the resulting noisy correspondences require a subsequent robust registration step (*e.g*. RANSAC) which does not fit well into typical learning frameworks.

Handcrafted Registration Methods. The original ICP algorithms [3, 5] circumvent the need for feature point matching by alternating between estimating point correspondences and finding the rigid transform that minimizes the point-to-point [3] or point-to-plane [5] error. Subsequent works try to improve upon the convergence of ICP by *e.g*., selecting suitable points [10, 25] or weighting point correspondences [11]. An overview of ICP variants can be found in [25]. Nevertheless, most ICP variants still require relatively good initialization to avoid converging to bad local minima. A notable exception, Go-ICP [40] uses a branch-and-bound scheme to search for the globally optimal registration at the trade-off of much longer computation

times. Alternatively, the basin of convergence of ICP can be widened using soft assignment strategies [6, 12, 33]. In particular, RPM [12] uses a soft assignment scheme with a deterministic annealing schedule to gradually "harden" the assignment at each iteration. IGSP [21] uses a different approach and measures the point similarity on a hybrid metric space with the spatial coordinates of the point and handcrafted BSC [8] features. However, the authors do not learn the features, and have to handcraft the weighting scheme between the spatial and feature distances. Our work builds upon the iterative framework of RPM. However, we consider distances between learned hybrid features during its soft assignment stage. Moreover, we do not use a predefined annealing schedule, instead we let the network decide the best settings to use at each iteration.

Learned Registration Methods. Recent works improve existing methods with deep learning. PointNetLK [1] utilizes PointNet [23] to compute a global representation of each point cloud, then optimizes the transforms to minimize the distances between the global descriptors in an iterative fashion analogous to the Lucas-Kanade algorithm [20, 2]. Later, PCRNet [30] improves the robustness against noise by replacing the Lucas-Kanade step with a deep network. Deep Closest Point [35] proposes a different approach. It extracts features for each point to compute a soft matching between the point clouds, before using a differentiable SVD module to extract the rigid transformation. They also utilize a transformer network [34] to incorporate global and inter point cloud information when computing the feature representations. Although shown to be more robust than traditional methods, the above works cannot handle partial-to-partial point cloud registration. A concurrent work, PRNet [36] incorporates keypoint detection to handle partial visibility. Our work uses a simpler approach and is more similar to Deep Closest Point, but unlike [35], our network is able to handle outliers and partial visibility through the use of Sinkhorn normalization [31] from RPM, and uses an iterative inference pipeline to achieve high precision.

3. Problem Formulation

Given two point clouds: $\mathbf{X} = \{\mathbf{x}_j \in \mathbb{R}^3 \mid j = 1, ..., J\}$ and $\mathbf{Y} = \{\mathbf{y}_k \in \mathbb{R}^3 \mid k = 1, ..., K\}$, which we denote as the *source* and *reference*, respectively, our objective is to recover the unknown rigid transformation $\{\mathbf{R}, \mathbf{t}\}$. $\mathbf{R} \in SO(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector that align the two point clouds. We assume the point normals can be easily computed from the points. Unlike the most recent deep learning-based related work [35], we do not assume a one-to-one correspondence between points. The two point clouds can have different number of points, *i.e.*, $J \neq K$ or cover different extents.

4. Background: Robust Point Matching

As mentioned earlier, our work builds upon the framework of RPM [12]. We briefly describe the algorithm in this section for completeness and interested readers are referred to [12] for further details. We define a match matrix $\mathbf{M} = \{0, 1\}^{J \times K}$ to represent the assignment of point correspondences, where each element

$$m_{jk} = \begin{cases} 1 & \text{if point } \mathbf{x}_j \text{ corresponds to } \mathbf{y}_k \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

Let us first consider the case where there is a one-to-one correspondence between the points. In this case, $\mathbf{M}$ is a square matrix. The registration problem can be formulated as finding the rigid transformation $\{\mathbf{R}, \mathbf{t}\}$ and correspondence matrix $\mathbf{M}$ which best maps points in $\mathbf{X}$ onto $\mathbf{Y}$, *i.e.*,

$$\underset{\mathbf{M},\mathbf{R},\mathbf{t}}{\arg\min} \sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk} (\|\mathbf{R}\mathbf{x}_j + \mathbf{t} - \mathbf{y}_k\|_2{}^2 - \alpha), \quad (2)$$

subject to $\sum_{k=1}^{K} m_{jk} = 1, \forall j$, $\sum_{j=1}^{J} m_{jk} = 1, \forall k$, and $m_{jk} \in \{0, 1\}, \forall jk$. The three constraints enforces $\mathbf{M}$ to be a permutation matrix. α is a parameter to control the number of correspondences rejected as outliers: any pair of points $(\mathbf{x}_j, \mathbf{y}_k)$ with a distance $\|\mathbf{R}\mathbf{x}_j + \mathbf{t} - \mathbf{y}_k\|_2{}^2 < \alpha$ is taken to be an inlier since setting $m_{jk} = 1$ decreases the cost in Eq. 2.

In RPM, the permutation matrix constraint is relaxed to a doubly stochastic constraint, *i.e.*, each $m_{jk} \in [0, 1]$. The minimization of Eq. 2 is then solved using deterministic annealing that iterates between two steps: (1) softassign, and (2) estimation of the rigid transformation. The match matrix $\mathbf{M}$ is estimated in the softassign step. To this end, each element $m_{jk} \in \mathbf{M}$ is first initialized as follows:

$$m_{jk} \leftarrow e^{-\beta(\|\mathbf{R}\mathbf{x}_j + \mathbf{t} - \mathbf{y}_k\|_2{}^2 - \alpha)}, \quad (3)$$

where β is the annealing parameter to be increased over each iteration step: small initial values of β result in soft assignments which help avoid falling into local minima. As β increases, $m_{jk} \to \{0, 1\}$ and the match matrix $\mathbf{M}$ becomes closer to a permutation matrix. Alternate row and column normalizations are then performed to satisfy the doubly stochastic constraints. This is due to a result from Sinkhorn [31], which states that a doubly stochastic matrix can be obtained from any square matrix with all positive entries by repeated applications of alternating row and column normalizations. Note that the assignments are deterministic (hence the term *deterministic* annealing). β controls the "hardness" of correspondences in a deterministic fashion. This contrasts with simulated annealing methods [16] where the decision of whether to accept a certain solution is a stochastic function of the temperature.

Once the correspondences are estimated, the rigid transformation $\{\mathbf{R}, \mathbf{t}\}$ can be computed. Various methods can be used for this purpose, we follow [19, 35] to compute $\{\mathbf{R}, \mathbf{t}\}$ using SVD (Section 5.3) in this paper. Lastly, when $J \neq K$ or in the presence of outlier non-matching points, the equality constraints on $\mathbf{M}$ in Eq. 2 become inequality constraints, but can be converted back into an equality constraint by introducing slack variables:

$$\sum_{k=1}^{K} m_{jk} \leq 1, \ \ \forall j \quad \rightarrow \quad \sum_{k=1}^{K+1} m_{jk} = 1, \ \ \forall j, \tag{4}$$

and likewise for the column constraints. In practice, this is implemented by adding an additional row and column of ones to the input of Sinkhorn normalization, *i.e.*, $\mathbf{M}_{J+1,K}, \mathbf{M}_{J,K+1}$.

5. Our RPM-Net

Figure 2 shows an illustration of our RPM-Net.. We make two main changes to RPM: (1) spatial distances are replaced with learned hybrid feature distances, and (2) our network decides the values of α, β (*c.f.* Eq. 3) at each iteration. At each iteration i, the source point cloud $\mathbf{X}$ is first transformed by the rigid transformation $\{\mathbf{R}^{i-1}, \mathbf{t}^{i-1}\}$ estimated from the previous step into the transformed point cloud $\tilde{\mathbf{X}}^i$. The feature extraction module (Section 5.1) then extracts hybrid features for the two point clouds. Concurrently, a secondary parameter prediction network (Section 5.2) predicts the optimal annealing parameters α, β. The hybrid features and α, β parameters are used to compute the initial match matrix, followed by Sinkhorn normalization to enforce the doubly stochastic constraints to get the final match matrix $\mathbf{M}^i$. Finally, the updated transformation $\{\mathbf{R}^i, \mathbf{t}^i\}$ is computed and used in the next iteration.

5.1. Feature Extraction

We replace the spatial distances in Eq. 3 with distances between learned features, *i.e.*,

$$m_{jk} \leftarrow e^{-\beta(\|F_{\tilde{\mathbf{x}}_j} - F_{\mathbf{y}_k}\|_2{}^2 - \alpha)}, \tag{5}$$

where $F_{\tilde{x}_j}$ and $F_{\mathbf{y}_k}$ are the features for points $\tilde{\mathbf{x}}_j \in \tilde{\mathbf{X}}^i$ and $\mathbf{y}_k \in \mathbf{Y}$, respectively. Replacing spatial coordinates with learned features allows our algorithm to consider additional sources of information, *e.g.* local geometric characteristics, during the computation of the assignments to avoid getting stuck in wrong local minima.

In our work, $F_{(\cdot)}$ is a hybrid feature containing information on both the point's spatial coordinates and local geometry. For a point $\mathbf{x}_c$ in either point cloud, we first define a local neighborhood $\mathcal{N}(\mathbf{x}_c)$ containing points within a distance of τ_{rad} from it. Its feature $F_{\mathbf{x}_c}$ is then given by:

$$F_{\mathbf{x}_c} = f_\theta(\mathbf{x}_c, \{\Delta\mathbf{x}_{c,i}\}, \{\text{PPF}(\mathbf{x}_c, \mathbf{x}_i)\}), \tag{6}$$

where f_θ is a deep network parameterized by θ, and $x_i \in \mathcal{N}(\mathbf{x}_c)$. $\Delta\mathbf{x}_{c,i}$ denotes the neighboring points translated into a local frame by subtracting away the coordinates of the centroid point [24]:

$$\Delta\mathbf{x}_{c,i} = \mathbf{x}_i - \mathbf{x}_c. \tag{7}$$

$\text{PPF}(\mathbf{x}_c, \mathbf{x}_i)$ are 4D point pair features (PPF) [26, 7] that describe the surface between the centroid point $\mathbf{x}_c$ and each neighboring point $\mathbf{x}_i$ in a rotation invariant manner:

$$\begin{aligned}\text{PPF}(\mathbf{x}_c, \mathbf{x}_i) = (&\angle(\mathbf{n}_c, \Delta\mathbf{x}_{c,i}), \angle(\mathbf{n}_i, \Delta\mathbf{x}_{c,i}), \\ &\angle(\mathbf{n}_c, \mathbf{n_i}), \|\Delta\mathbf{x}_{c,i}\|_2),\end{aligned} \tag{8}$$

where $\mathbf{n}_c$ and $\mathbf{n}_i$ are the normals of points $\mathbf{x}_c$ and $\mathbf{x}_i$. The above two inputs describe the local geometry, but do not contain information about the absolute positions of the points. In this work, we also include the absolute position of the centroid point $\mathbf{x}_c$. This gives our network the ability to refine the registration iteratively as in the original RPM.

We implement f_θ using a PointNet [23] which is able to pool an arbitrary number of orderless points into a single descriptor. Specifically, to obtain the feature for $\mathbf{x}_c$, we first concatenate the raw features of each neighboring point $\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_c)$ into a 10-D input vector $[\mathbf{x}_c, \Delta\mathbf{x}_{c,i}, \text{PPF}(\mathbf{x}_c, \mathbf{x}_i)]$. We then feed them into a series of shared dense layers, a max-pooling and additional dense layers, followed by ℓ^2 normalization to obtain a single feature vector $F_{\mathbf{x}_c}$.

5.2. Parameter Prediction Network

In the original RPM algorithm, the value of the outlier parameter α and the annealing schedule for β (Eq. 5) are manually set for each dataset. These parameters are dataset dependent and have to be tuned for each dataset. In our RPM-Net, these parameters are difficult to set manually since they are dependent on the learned features. We argue that a fixed annealing schedule is unnecessary as the parameters can be chosen based on the current state of alignment instead of the iteration count. Correspondingly, we use a secondary network that takes both point clouds as input and predicts the parameters for the current iteration. In particular, we concatenate the two point clouds to form a $(J+K, 3)$ matrix, augment it with a fourth column containing 0 or 1 depending on which point cloud the point originates from, and feed it into a PointNet that outputs α and β. To ensure that the predicted α and β are positive, we use a softplus activation for the final layer.

5.3. Estimating the Rigid Transformation

Once the soft assignments are estimated, the final step is to estimate the rigid transformation. For each point $\mathbf{x}_j$ in $\mathbf{X}$, we compute the corresponding coordinate in $\mathbf{Y}$:

$$\hat{\mathbf{y}}_j = \frac{1}{\sum_k^K m_{jk}} \sum_k^K m_{jk} \cdot \mathbf{y}_k. \tag{9}$$

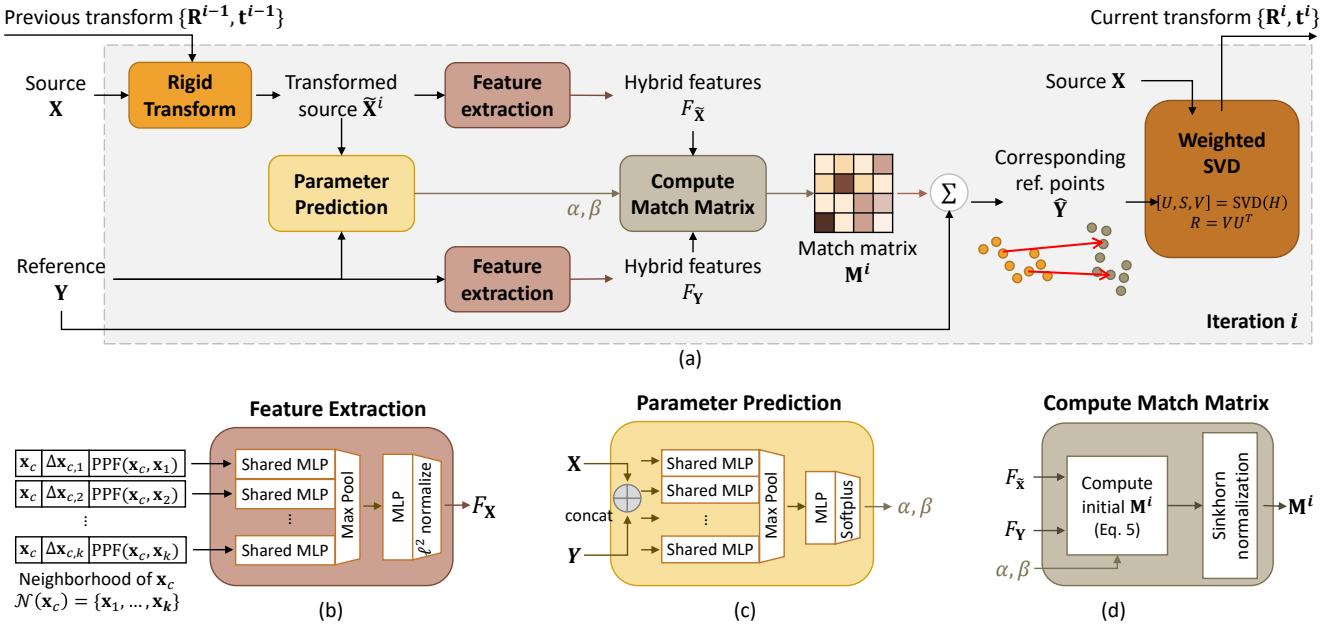

Figure 2. (a) Overview of our RPM-Net, (b) feature extraction network, (c) parameters prediction network, and (d) computation of match matrix $\mathbf{M}$. Superscripts denote the iteration count.

We then follow [19, 35] and use the SVD to solve for the rigid transformation, which has been shown to be differentiable in [22]. Since not every point $\mathbf{x}_j$ might have a correspondence, we weigh each correspondence $(\mathbf{x}_j, \hat{\mathbf{y}}_j)$ by $w_j = \sum_k^K m_{jk}$ when computing the rigid transformation.

5.4. Loss Functions

Our primary loss function is the ℓ^1 distance between the source point cloud $\mathbf{X}$ transformed using the groundtruth transformation $\{\mathbf{R}_{gt}, \mathbf{t}_{gt}\}$ and the predicted transformation $\{\mathbf{R}_{pred}, \mathbf{t}_{pred}\}$ [19]:

$$\mathcal{L}_{reg} = \frac{1}{J} \sum_{j}^{J} |(\mathbf{R}_{gt}\mathbf{x}_j + \mathbf{t}_{gt}) - (\mathbf{R}_{pred}\mathbf{x}_j + \mathbf{t}_{pred})| \tag{10}$$

We notice empirically that the network has the tendency to label most points as outliers with only the above registration loss. To alleviate this issue, we add a secondary loss on the computed match matrix $\mathbf{M}$ to encourage inliers:

$$\mathcal{L}_{inlier} = -\frac{1}{J} \sum_{j}^{J} \sum_{k}^{K} m_{jk} - \frac{1}{K} \sum_{k}^{K} \sum_{j}^{J} m_{jk}. \tag{11}$$

The overall loss is the weighted sum of the two losses:

$$\mathcal{L}_{total} = \mathcal{L}_{reg} + \lambda \mathcal{L}_{inlier}, \tag{12}$$

where we use $\lambda = 0.01$ in all our experiments. We compute the loss for every iteration i, but weigh the losses by $\frac{1}{2}^{(N_i - i)}$ to give later iterations higher weights, where N_i is the total number of iterations during training.

5.5. Implementation Details

The overall network is implemented as a recurrent neural network with an inner loop for the Sinkhorn normalization. We follow [29] in our implementation of the Sinkhorn normalization by unrolling it for a fixed number of steps (set to 5). Although gradients can flow from one iteration to the other, in practice, that does not improve performance and causes training instability. We adopt a simple solution of stopping the $\{\mathbf{R}, \mathbf{t}\}$ gradients at the start of each iteration. This means every iteration becomes independent and we can just execute one iteration during training. Nevertheless, we run $N_i = 2$ iterations of alignment during training since this allows the network to see data with smaller misalignments more often and consequently learn how to refine the registration in subsequent iterations. During test time, we use $N_i = 5$ iterations to achieve more precise registration. For both feature extraction and parameter prediction networks, we use ReLU activation with group normalization [38] on all layers except the last. Our feature extraction network considers a neighborhood of $\tau_{rad} = 0.3$, and outputs features of dimension 96. We train the network using ADAM optimizer [15] with a learning rate of 0.0001.

6. Experiments

6.1. ModelNet40 Dataset

We evaluate our algorithm on the ModelNet40 [39] dataset, which contains CAD models from 40 man-made object categories. We make use of the processed data from

[23], which contains 2,048 points sampled randomly from the mesh faces and normalized into a unit sphere. The dataset contains official train/test splits for each category. To evaluate the ability of our network to generalize to different object categories, we use the train and test splits for the first 20 categories for training and validation respectively, and the test split of the remaining categories for testing. This results in 5,112 train, 1,202 validation, and 1,266 test models. Following [35], we sample rotations by sampling three Euler angle rotations in the range $[0, 45°]$ and translations in the range $[-0.5, 0.5]$ on each axis during training and testing. We transform the source point cloud $\mathbf{X}$ using the sampled rigid transform and the task is to register it to the unperturbed reference point cloud $\mathbf{Y}$.

6.2. Evaluation Metrics

We evaluate the registration by computing the mean isotropic rotation and translation errors:

$$\text{Error}(\mathbf{R}) = \angle(\mathbf{R}_{GT}^{-1}\hat{\mathbf{R}}), \quad \text{Error}(\mathbf{t}) = \|\mathbf{t}_{GT} - \hat{\mathbf{t}}\|_2, \quad (13)$$

where $\{\mathbf{R}_{GT}, \mathbf{t}_{GT}\}$ and $\{\hat{\mathbf{R}}, \hat{\mathbf{t}}\}$ denote the groundtruth and estimated transformation, respectively. $\angle(\mathbf{X}) = \arccos(\frac{tr(\mathbf{X})-1}{2})$ returns the angle of rotation matrix $\mathbf{X}$ in degrees. For consistency with previous work [35], we also provide the mean absolute errors over euler angles and translation vectors. Note however that these metrics are anisotropic.

The above metrics unfairly penalizes the alignment to an alternative solution in the case of symmetry commonly found in ModelNet40 models, so we also propose a modified Chamfer distance metric between the transformed source point cloud $\mathbf{X}$ and the reference point cloud $\mathbf{Y}$:

$$\tilde{CD}(\mathbf{X}, \mathbf{Y}) = \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} \min_{\mathbf{y} \in \mathbf{Y}_{\text{clean}}} \|\mathbf{x} - \mathbf{y}\|_2{}^2 + \frac{1}{|\mathbf{Y}|} \sum_{\mathbf{y} \in \mathbf{Y}} \min_{\mathbf{x} \in \mathbf{X}_{\text{clean}}} \|\mathbf{x} - \mathbf{y}\|_2{}^2, \quad (14)$$

where we modified the Chamfer distance to compare with the *clean* and *complete* versions of the other point cloud.

6.3. Baseline Algorithms

We compare the performance of our RPM-Net with the following handcrafted registration algorithms: ICP [3], FGR [43], and RPM [12], as well as recent deep learning based registration works: PointNetLK [1] and Deep Closest Point (DCP-v2) [35]. We use the implementations of ICP and FGR in Intel Open3D [44], and our own implementation of RPM. For PointNetLK and Deep Closest Point, we use the implementation provided by the authors but retrained the networks since both works do not provide the required pretrained models[2].

Method	Anisotropic err. (Rot.)	Anisotropic err. (Trans.)	Isotropic err. (Rot.)	Isotropic err. (Trans.)	$\tilde{CD}$
ICP	3.114	0.02328	6.407	0.0506	0.002975
RPM	1.121	0.00636	2.426	0.0141	0.000360
FGR	**0.010**	**0.00011**	**0.022**	**0.0002**	*0.000012*
PointNetLK	0.418	0.00241	0.847	0.0054	0.000138
DCP-v2	2.074	0.01431	3.992	0.0292	0.001777
Ours	*0.028*	*0.00016*	*0.056*	*0.0003*	**0.000003**

Table 1. Performance on Clean Data. Bold and italics denote best and second best performing measures. Note: DCP-v2's results are based on our trained model and are marginally worse than its reported [35] performance of an anisotropic error of 2.007° (rot) and 0.0037 (trans).

6.4. Clean Data

We follow the protocol in [35] and evaluate the registration performance on the clean data. We randomly sample the same 1,024 points for the source and reference point clouds from the 2,048 points in ModelNet40 dataset, and then apply a random rigid transformation on the source point cloud and shuffle the point order. Under this setting, each point in the source point cloud $\mathbf{X}$ has a exact correspondence in the reference point cloud $\mathbf{Y}$. All learned models including ours are trained on the clean data. Table 1 shows the performance of the various algorithms on clean data. Our method achieves very accurate registration and ranks first or second in all measures. It outperforms all learned and handcrafted methods except FGR. However, as we will see in subsequent sections, FGR is highly susceptible to noise. A qualitative comparison of the registration results can be found in Figure 3(a).

6.5. Gaussian Noise

In this experiment, we evaluate the performance in the presence of noise and sampling differences, which are present in real world point clouds. We randomly sample 1,024 points from the models, but *independently* for the source and reference point clouds. After applying the random rigid transform to the source point cloud, we randomly jitter the points in both point clouds by noises sampled from $\mathcal{N}(0, 0.01)$ and clipped to [-0.05, 0.05] on each axis. This experiment is significantly more challenging due to noise and non one-to-one correspondences. We train all learned models on the noisy data with the exception of PointNetLK, which we reuse the model from the previous section since that gives better performance. The results are shown in Table 2. Our network outperforms all handcrafted and learned methods as we explicitly handle for points with no correspondences. On the other hand, Deep Closest Point requires

[2]Deep Closest Point provides pretrained models but not for matching of unseen categories and noisy data.

Method	Anisotropic err. (Rot.)	(Trans.)	Isotropic err. (Rot.)	(Trans.)	$\tilde{CD}$
ICP	3.414	0.0242	6.999	0.0514	0.00308
RPM	1.441	0.0094	2.994	0.0202	0.00083
FGR	1.724	0.0120	2.991	0.0252	0.00130
PointNetLK	1.528	0.0128	2.926	0.0262	0.00128
DCP-v2	4.528	0.0345	8.922	0.0707	0.00420
Ours	**0.343**	**0.0030**	**0.664**	**0.0062**	**0.00063**

Table 2. Performance on data with Gaussian noise. The Chamfer distance using groundtruth transformations is 0.00055.

Method	Anisotropic err. (Rot.)	(Trans.)	Isotropic err. (Rot.)	(Trans.)	$\tilde{CD}$
ICP	13.719	0.132	27.250	0.280	0.0153
RPM	9.771	0.092	19.551	0.212	0.0081
FGR	19.266	0.090	30.839	0.192	0.0119
PointNetLK	15.931	0.142	29.725	0.297	0.0235
DCP-v2	6.380	0.083	12.607	0.169	0.0113
Ours	**0.893**	**0.0087**	**1.712**	**0.018**	**0.00085**

Table 3. Performance on partially visible data with noise. The Chamfer distance using groundtruth transformations is 0.00055.

every point to have a correspondence and does not perform well when this condition is violated. A qualitative example of registration on noisy data can be found in Figure 3(b).

6.6. Partial Visibility

We evaluate the performance on partially visible point clouds, where the two point clouds do not fully overlap in extent. Depending on the acquisition method, real world point cloud data are often partial, *e.g.* RGD-D scans contain only points that are visible to the camera. Consequently, handling partial point clouds is an important requirement for many applications. We simulate partial visibility in the following manner. For each point cloud, we sample a half-space with a random direction $\in \mathcal{S}^2$ and shift it such that approximately 70% of the points are retained. Similar to the previous section, points are jittered and sampled independently. For this experiment, we downsample to 717 points instead of 1,024 to maintain a similar point density as the previous sections. We train DCP-v2 and our method on the partially visible data. For PointNetLK, we adopt similar procedure suggested by the authors to sample visible points in the other point cloud. However, we sample visible points from the points with a respective nearest point in the other point cloud within a distance of $\tau = 0.02$ during each iteration since only the partial point cloud is available for both point clouds in our setting. This procedure improves inference performance but did not work well during training. Consequently, we continue to use the clean model of PointNetLK in this experiment. Table 3 shows the performance on partially visible data. Our approach significantly outperforms all baseline methods. Interestingly, despite our best efforts at tuning the parameters in RPM, it also performs poorly. Since both RPM and our approach share a similar scheme for rejecting outliers, this highlights the benefit of our learned feature distances. Example results on partially visible data are shown in Figures 1 and 3(c-e).

6.7. Ablation Studies

We perform ablation studies to better understand how various choices affect the performance of the algorithm. All studies in this section are evaluated on the partial visibility setting, and we only show the isotropic and Chamfer distance metrics for conciseness.

$\mathbf{x}$	$\Delta\mathbf{x}$	PPF	Anneal	Isotropic err. (Rot.)	(Trans.)	$\tilde{CD}$
✓	✓		✓	3.302	0.0350	0.00153
✓		✓	✓	2.781	0.0273	0.00123
	✓	✓	✓	5.501	0.0496	0.00351
✓	✓	✓		2.220	0.0238	0.00103
✓	✓	✓	✓	**1.712**	**0.0183**	**0.00085**

Table 4. Effects of each component on the registration performance. $\mathbf{x}$ and $\Delta\mathbf{x}$ denote the absolute centroid center coordinates and the local coordinates of the neighboring points respectively.

Effects of different components. Comparing rows 1-3 and 5 of Table 4, we observe that all of $\mathbf{x}$, $\Delta\mathbf{x}$ and PPF are required to achieve the highest performance. Excluding the absolute positions of the points (row 3) results in a significant drop in performance. This indicates the importance of considering point positions at each iteration when performing iterative refinement. It is also noteworthy that even without PPF features (row 1), the algorithm still outperforms DCP-v2. This is despite DCP-v2 using a more sophisticated Dynamic Graph CNN [37] architecture and an attention [34] network. We attribute this to our outlier handling and iterative registration scheme. To understand the importance of our parameter prediction network, we lastly compare with a variant of our network in Table 4 (row 4) where we replace our parameter prediction network with two learnable parameters for α and β. These parameters are trained with the network weights, and the same values are used for each iteration. We can see that the learned annealing schedule from the parameter prediction network improves registration performance.

How many iterations are needed? Figure 4 shows the average Chamfer distance after each iteration. Most performance gains are in the first two iterations, and the registration mostly converges after 5 iterations (which we use for all experiments).

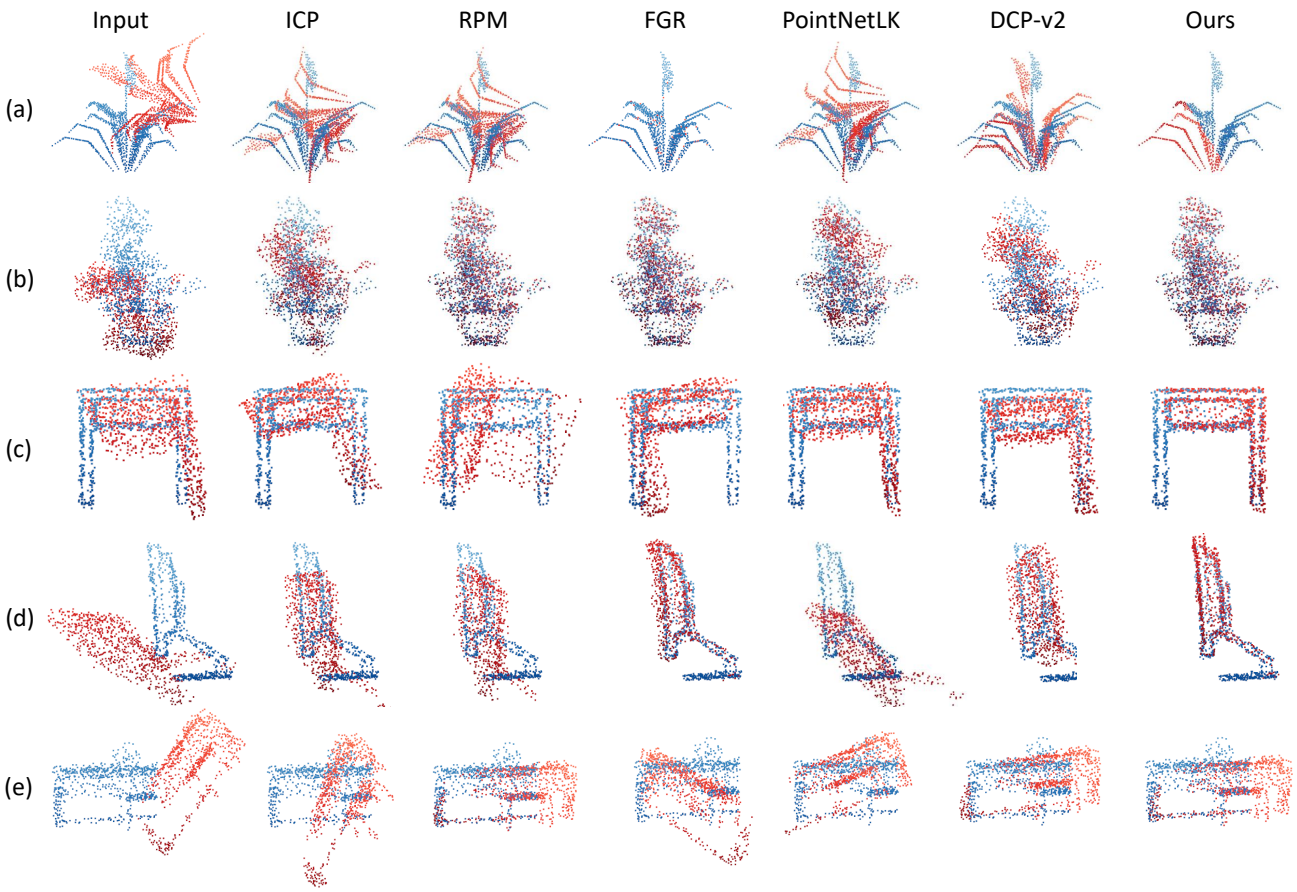

Figure 3. Qualitative registration examples on (a) Clean data, (b) Noisy data, and (c, d, e) Partially visible data.

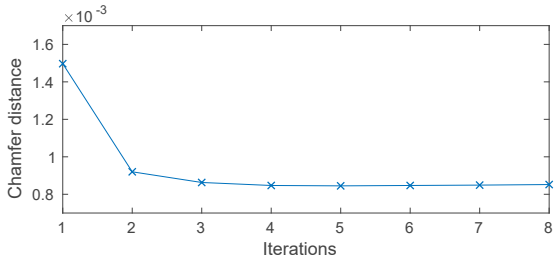

Figure 4. Chamfer distance over registration iterations. The initial average Chamfer distance of the inputs is 0.1899, which is not shown in graph for clarity.

6.8. Computational Efficiency

We compare the inference time of various algorithms in Table 5, averaged over the entire test set. We perform this experiment on a 3.0GHz Intel i7-6950X and a Nvidia Titan RTX. For our algorithm, we provide the timings for 5 iterations as used in previous experiments. Note that ICP and FGR are executed on CPU and the remaining algorithms on a GPU. Our algorithm is significantly faster than RPM which requires a large number of iterations. It is however slower than ICP as well as the non-iterative DCP-v2.

# points	ICP	RPM	FGR	PointNetLK	DCP-v2	Ours
512	8	66	22	161	5	25
1024	18	144	84	176	9	52
2048	28	447	148	209	21	178

Table 5. Average time required for registering a point cloud pair of various sizes (in milliseconds).

7. Conclusion

We present the RPM-Net for rigid point cloud registration. Our approach is a deep learning-based RPM that desensitizes initialization and improves convergence behavior with learned fusion features. Furthermore, the use of the differentiable Sinkhorn normalization with slack variables to enforce partial doubly stochastic constraints allows our method to explicitly handle outliers. We also propose a secondary network to predict optimal annealing parameters that further improves performance. Experimental results show our method yields state-of-the-art performance on the ModelNet40 dataset over various evaluation criteria.

Acknowledgement. This work was partially supported by the Singapore MOE Tier 1 grant R-252-000-A65-114.

References

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & efficient point cloud registration using pointnet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7163–7172, 2019. 3, 6

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. 3

[3] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992. 1, 2, 6

[4] Hui Chen and Bir Bhanu. 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007. 2

[5] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2724–2729 vol.3, 1991. 2

[6] Haili Chui and Anand Rangarajan. A feature registration framework using mixture models. In *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. MMBIA-2000 (Cat. No. PR00737)*, pages 190–197. IEEE, 2000. 2, 3

[7] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4

[8] Zhen Dong, Bisheng Yang, Yuan Liu, Fuxun Liang, Bijun Li, and Yufu Zang. A novel binary shape context for 3D local surface description. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:431 – 452, 2017. 3

[9] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision (ECCV)*, pages 224–237. Springer, 2004. 2

[10] Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically stable sampling for the icp algorithm. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 260–267. IEEE, 2003. 2

[11] Guy Godin, Marc Rioux, and Rejean Baribeau. Three-dimensional registration using range and intensity information. In Sabry F. El-Hakim, editor, *Videometrics III*, volume 2350, pages 279 – 290. International Society for Optics and Photonics, SPIE, 1994. 2

[12] Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness. New algorithms for 2D and 3D point matching: pose estimation and correspondence. *Pattern Recognition*, 31(8):1019 – 1031, 1998. 1, 2, 3, 6

[13] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3D local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016. 2

[14] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):433–449, 1999. 2

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[16] Scott Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220:671–80, 06 1983. 3

[17] Hongdong Li and Richard Hartley. The 3D-3D registration problem revisited. In *International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007. 1

[18] Jiaxin Li and Gim Hee Lee. USIP: Unsupervised stable interest point detection from 3d point clouds. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[19] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepICP: An end-to-end deep neural network for 3D point cloud registration. In *International Conference on Computer Vision (ICCV)*, 2019. 4, 5

[20] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, 1981. 3

[21] Y. Pan, B. Yang, F. Liang, and Z. Dong. Iterative global similarity points: A robust coarse-to-fine integration solution for pairwise 3d point cloud registration. In *International Conference on 3D Vision (3DV)*, pages 180–189, Sep. 2018. 3

[22] Théodore Papadopoulo and Manolis IA Lourakis. Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision (ECCV)*, pages 554–570. Springer, 2000. 5

[23] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2, 3, 4, 6

[24] Charles R. Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 2, 4

[25] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 145–152. IEEE, 2001. 2

[26] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009. 2, 4

[27] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3384–3391, 2008. 2

[28] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014. 2

[29] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. DeepPermNet: Visual permutation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3949–3957, 2017. 5

[30] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. PCRNet: Point cloud registration network using pointnet encoding. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 3

[31] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964. 2, 3

[32] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3D data description. In *ACM Workshop on 3D Object Retrieval*, 3DOR '10, pages 57–62. ACM, 2010. 2

[33] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *European Conference on Computer Vision (ECCV)*, pages 558–569. Springer, 2004. 2, 3

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 3, 7

[35] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 3, 4, 5, 6

[36] Yue Wang and Justin M Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *Advances in Neural Information Processing Systems 32*, pages 8814–8826. Curran Associates, Inc., 2019. 3

[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)*, 2019. 7

[38] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, pages 3–19. Springer, 2018. 5

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 5

[40] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(11):2241–2254, 2015. 2

[41] Zi Jian Yew and Gim Hee Lee. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 2

[42] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 199–208, 2017. 2

[43] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016. 6

[44] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 6