PRNet: Self-Supervised Learning for Partial-to-Partial Registration

Yue Wang
Massachusetts Institute of Technology
yuewangx@mit.edu

Justin Solomon
Massachusetts Institute of Technology
jsolomon@mit.edu

Abstract

We present a simple, flexible, and general framework titled Partial Registration Network (PRNet), for partial-to-partial point cloud registration. Inspired by recently-proposed learning-based methods for registration, we use deep networks to tackle non-convexity of the alignment and partial correspondence problems. While previous learning-based methods assume the entire shape is visible, PRNet is suitable for partial-to-partial registration, outperforming PointNetLK, DCP, and non-learning methods on synthetic data. PRNet is self-supervised, jointly learning an appropriate geometric representation, a keypoint detector that finds points in common between partial views, and keypoint-to-keypoint correspondences. We show PRNet predicts keypoints and correspondences consistently across views and objects. Furthermore, the learned representation is transferable to classification.

1 Introduction

Registration is the problem of predicting a rigid motion aligning one point cloud to another. Algorithms for this task have steadily improved, using machinery from vision, graphics, and optimization. These methods, however, are usually orders of magnitude slower than “vanilla” Iterative Closest Point (ICP), and some have hyperparameters that must be tuned case-by-case. The trade-off between efficiency and effectiveness is steep, reducing generalizability and/or practicality.

Recently, PointNetLK [1] and Deep Closest Point (DCP) [2] show that learning-based registration can be faster and more robust than classical methods, even when trained on different datasets. These methods, however, cannot handle partial-to-partial registration, and their one-shot constructions preclude refinement of the predicted alignment.

We introduce the Partial Registration Network (PRNet), a sequential decision-making framework designed to solve a broad class of registration problems. Like ICP, our method is designed to be applied iteratively, enabling coarse-to-fine refinement of an initial registration estimate. A critical new component of our framework is a keypoint detection sub-module, which identifies points that match in the input point clouds based on co-contextual information. Partial-to-partial point cloud registration then boils down to detecting keypoints the two point clouds have in common, matching these keypoints to one another, and solving the Procrustes problem.

Since PRNet is designed to be applied iteratively, we use Gumbel–Softmax [3] with a straight-through gradient estimator to sample keypoint correspondences. This new architecture and learning procedure modulates the sharpness of the matching; distant point clouds given to PRNet can be coarsely matched using a diffuse (fuzzy) matching, while the final refinement iterations prefer sharper maps. Rather than introducing another hyperparameter, PRNet uses a sub-network to predict the temperature [4] of the Gumbel–Softmax correspondence, which can be cast as a simplified version of the actor-critic method. That is, PRNet learns to modulate the level of map sharpness each time it is applied.
We train and test PRNet on ModelNet40 and on real data. We visualize the keypoints and correspondences for shapes from the same or different categories. We transfer the learned representations to shape classification using a linear SVM, achieving comparable performance to state-of-the-art supervised methods on ModelNet40.

Contributions. We summarize our key contributions as follows:

- We present the Partial Registration Network (PRNet), which enables partial-to-partial point cloud registration using deep networks with state-of-the-art performance.
- We use Gumbel–Softmax with straight-through gradient estimation to obtain a sharp and near-differentiable mapping function.
- We design an actor-critic closest point module to modulate the sharpness of the correspondence using an action network and a value network. This module predicts more accurate rigid transformations than differentiable soft correspondence methods with fixed parameters.
- We show registration is a useful proxy task to learn representations for 3D shapes. Our representations can be transferred to other tasks, including keypoint detection, correspondence prediction, and shape classification.
- We release our code to facilitate reproducibility and future research.

2 Related Work

Rigid Registration. ICP [5] and variants [6, 7, 8, 9] have been widely used for registration. Recently, probabilistic models [10, 11, 12] have been proposed to handle uncertainty and partiality. Another trend is to improve the optimization: [13] applies Levenberg–Marquardt to the ICP objective, while global methods seek a solution using branch-and-bound [14]. Riemannian optimization [15], convex relaxation [16], mixed-integer programming [17], and semidefinite programming [18].

Learning on Point Clouds and 3D Shapes. Deep Sets [19] and PointNet [20] pioneered deep learning on point sets, a challenge problem in learning and vision. These methods take coordinates as input, embed them to high-dimensional space using shared multilayer perceptrons (MLPs), and use a symmetric function (e.g., max or $\sum$) to aggregate features. Follow-up works incorporate local information, including PointNet++ [21], DGCNN [22], PointCNN [23], and PCNN [24]. Another branch of 3D learning designs convolution-like operations for shapes or applies graph convolutional networks (GCNs) [25, 26] to triangle meshes [27, 28], exemplifying architectures on non-Euclidean data termed geometric deep learning [29]. Other works, including SPLATNet [30], SplineCNN [31], KPConv [32], and GWCNN [33], transform 3D shapes to regular grids for feature learning.

Keypoints and Correspondence. Correspondence and registration are dual tasks. Correspondence is the approach while registration is the output, or vice versa. Countless efforts tackle the correspondence problem, either at the point-to-point or part-to-part level. Due to the $O(n^2)$ complexity of point-to-point correspondence matrices and $O(n!)$ possible permutations, most methods (e.g., [34, 35, 36, 37, 38, 39, 40, 41]) compute a sparse set of correspondences and extend them to dense maps, often with bijectivity as an assumption or regularizer. Other efforts use more exotic representations of correspondences. For example, functional maps [42] generalize to mappings between functions on shapes rather than points on shapes, expressing a map as a linear operator in the Laplace–Beltrami eigenbasis. Mathematical methods like functional maps can be made ‘deep’ using priors learned from data: Deep functional maps [43, 44] learn descriptors rather than designing them by hand.

For partial-to-partial registration, we cannot compute bijective correspondences, invalidating many past representations. Instead, keypoint detection is more secure. To extract a sparser representation, KeyPointNet [45] uses registration and multiview consistency as supervision to learn a keypoint detector on 2D images; our method performs keypoint detection on point clouds. In contrast to our model, which learns correspondences from registration, [46] uses correspondence prediction as the training objective to learn how to segment parts. In particular, it utilizes PointNet++ [21] to product point-wise features, generates matching using a correspondence proposal module, and finally trains the pipeline with ground-truth correspondences.

Self-supervised Learning. Humans learn knowledge not only from teachers but also by predicting and reasoning about unlabeled information. Inspired by this observation, self-supervised learning usually involves predicting part of an input from another part [37, 48], solving one task using features
learned from another task [45] and/or enforcing consistency from different views/modalities [49, 50]. Self-supervised pretraining is an effective way to transfer knowledge learned from massive unlabeled data to tasks where labeled data is limited. For example, BERT [51] surpasses state-of-the-art in natural language processing by learning from contextual information. ImageNet Pretrain [52] commonly provides initialization for vision tasks. Video-audio joint analysis [53, 54, 55] utilizes modality consistency to learn representations. Our method is also self-supervised, in the sense that no labeled data is needed.

Actor–Critic Methods. Many recent works can be counted as actor–critic methods, including deep reinforcement learning [56], generative modeling [57], and sequence generation [58]. These methods generally involve two functions: taking actions and estimating values. The predicted values can be used to improve the actions while the values are collected when the models interact with environment. PRNet uses a sub-module (value head) to predict the level of granularity at which we should map two shapes. The value adjusts the temperature of Gumbel–Softmax in the action head.

3 Method

We establish preliminaries about the rigid alignment problem and related algorithms in §3.1; then, we present PRNet in §3.2. For ease of comparison to previous work, we use the same notation as [2].

3.1 Preliminaries: Registration, ICP, and DCP

Consider two point clouds $X = \{x_1, \ldots, x_i, \ldots, x_N\} \subset \mathbb{R}^3$ and $Y = \{y_1, \ldots, y_j, \ldots, y_M\} \subset \mathbb{R}^3$. The basic task in rigid registration is to find a rotation $R_{XY}$ and translation $t_{XY}$ that rigidly align $X$ to $Y$. When $M = N$, ICP and its peers approach this task by minimizing the objective function

$$E(R_{XY}, t_{XY}, m) = \frac{1}{N} \sum_{i=1}^{N} \| R_{XY} x_i + t_{XY} - y_{m(x_i)} \|^2.$$  \hspace{1cm} (1)

Here, the rigid transformation is defined by a pair $[R_{XY}, t_{XY}]$, where $R_{XY} \in \text{SO}(3)$ and $t_{XY} \in \mathbb{R}^3$; $m$ maps from points in $X$ to points in $Y$. Assuming $m$ is fixed, the alignment in (1) is given in closed-form by

$$R_{XY} = VU^T \quad \text{and} \quad t_{XY} = -R_{XY} \bar{x} + \bar{y},$$  \hspace{1cm} (2)

where $U$ and $V$ are obtained using the singular value decomposition (SVD) $H = USV^T$, with $H = \sum_{i=1}^{N} (x_i - \bar{x})(y_{m(x_i)} - \bar{y})^T$. In this expression, centroids of $X$ and $Y$ are defined as $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_{m(x_i)}$, respectively.

We can understand ICP and the more recent learning-based DCP method [2] as providing different choices of $m$:

**Iterative Closest Point.** ICP chooses $m$ to minimize (1) with $[R_{XY}, t_{XY}]$ fixed, yielding:

$$m(x_i, Y) = \arg \min_j \| R_{XY} x_i + t_{XY} - y_j \|_2$$  \hspace{1cm} (3)

ICP approaches a fixed point by alternating between (2) and (3); each step decreases the objective (1). Since (1) is non-convex, however, there is no guarantee that ICP reaches a global optimum.
**Deep Closest Point.** DCP uses deep networks to learn \( m \). In this method, \( \mathcal{X} \) and \( \mathcal{Y} \) are embedded using learned functions \( \mathcal{F}_\mathcal{X} \) and \( \mathcal{F}_\mathcal{Y} \) defined by a Siamese DGCNN \([22]\); these lifted point clouds are optionally contextualized by a Transformer module \([59]\), yielding embeddings \( \Phi_{\mathcal{X}} \) and \( \Phi_{\mathcal{Y}} \). The mapping \( m \) is then

\[
m(x_i, \mathcal{Y}) = \text{softmax}(\Phi_{\mathcal{Y}} \Phi_{x_i}^\top).
\]

This formula is applied in one shot followed by \([2]\) to obtain the rigid alignment. The loss used to train this pipeline is mean-squared error (MSE) between ground-truth rigid motion from synthetically-rotated point clouds and prediction; the network is trained end-to-end.

### 3.2 Partial Registration Network

DCP is a one-shot algorithm, in that a single pass through the network determines the output for each prediction task. Analogously to ICP, PRNet is designed to be *iterative*: multiple passes of a point cloud through PRNet refine the alignment. The steps of PRNet, illustrated in Figure 1, are as follows:

1. take as input point clouds \( \mathcal{X} \) and \( \mathcal{Y} \);
2. detect keypoints of \( \mathcal{X} \) and \( \mathcal{Y} \);
3. predict a mapping from keypoints of \( \mathcal{X} \) to keypoints of \( \mathcal{Y} \);
4. predict a rigid transformation \([R_{\mathcal{XY}}, t_{\mathcal{XY}}]\) aligning \( \mathcal{X} \) to \( \mathcal{Y} \) based on the keypoints and map;
5. transform \( \mathcal{X} \) using the obtained transformation;
6. return to 1 using the pair \((R_{\mathcal{XY}}, \mathcal{X} + t_{\mathcal{XY}}, \mathcal{Y})\) as input.

When predicting a mapping from keypoints in \( \mathcal{X} \) to keypoints in \( \mathcal{Y} \), PRNet uses Gumbel–Softmax \([3]\) to sample a matching matrix, which is sharper than \([4]\) and approximately differentiable. It has a value network to predict a temperature for Gumbel–Softmax, so that the whole framework can be seen as an actor-critic method. We present details of and justifications behind the design below.

**Notation.** Denote by \( \mathcal{X}^p = \{x_i^p, \ldots, x_N^p\} \) the rigid motion of \( \mathcal{X} \) to align to \( \mathcal{Y} \) after \( p \) applications of PRNet; \( \mathcal{X}^1 \) and \( \mathcal{Y}^1 \) are initial input shapes. We will use \([R_{\mathcal{XY}}^p, t_{\mathcal{XY}}^p]\) to denote the \( p \)-th rigid motion predicted by PRNet for the input pair \((\mathcal{X}, \mathcal{Y})\).

Since our training pairs are synthetically generated, before applying PRNet we know the ground-truth \([R_{\mathcal{XY}}^*, t_{\mathcal{XY}}^*]\) aligning \( \mathcal{X} \) to \( \mathcal{Y} \). From these values, during training we can compute “local” ground-truth \([R_{\mathcal{XY}}^*, t_{\mathcal{XY}}^*] \) on-the-fly, which maps the current \((\mathcal{X}^p, \mathcal{Y})\) to the best alignment:

\[
R_{\mathcal{XY}}^p = R_{\mathcal{XY}}^* R_{\mathcal{XY}}^{1:p-1} \quad \text{and} \quad t_{\mathcal{XY}}^p = t_{\mathcal{XY}}^* - R_{\mathcal{XY}}^p t_{\mathcal{XY}}^{1:p-1}.
\]

where

\[
R_{\mathcal{XY}}^{1:p-1} = R_{\mathcal{XY}}^{p-1} \ldots R_{\mathcal{XY}}^1 \quad \text{and} \quad t_{\mathcal{XY}}^{1:p-1} = R_{\mathcal{XY}}^{p-1} t_{\mathcal{XY}}^{1:p-1} + t_{\mathcal{XY}}^{p-1}.
\]

We use \( m^p \) to denote the mapping function in \( p \)-th step.

Synthesizing the notation above, \( \mathcal{X}^p \) is given by

\[
x_i^p = R_{\mathcal{XY}}^{p-1} x_i^{p-1} + t_{\mathcal{XY}}^{p-1}.
\]

where

\[
R_{\mathcal{XY}}^p = V^p U^p \quad \text{and} \quad t_{\mathcal{XY}}^p = -R_{\mathcal{XY}}^p \bar{u}^p + \bar{y}.
\]

In this equation, \( U^p \) and \( V^p \) are computed using \([2]\) from \( \mathcal{X}^p \), \( \mathcal{Y}^p \), and \( m^p \).

**Keypoint Detection.** For partial-to-partial registration, usually \( N \neq M \) and only subsets of \( \mathcal{X} \) and \( \mathcal{Y} \) match to one another. To detect these mutually-shared patches, we design a simple yet efficient keypoint detection module based on the observation that the \( L^2 \) norms of features tend to indicate whether a point is important.

Using \( X_k^p \) and \( Y_k^p \) to denote the \( k \) keypoints for \( \mathcal{X}^p \) and \( \mathcal{Y}^p \), we take

\[
\begin{align*}
X_k^p &= \mathcal{X}^p(\text{topk}(\|\Phi_{x_i}^p\|_2, \ldots, \|\Phi_{x_i}^p\|_2, \ldots, \|\Phi_{x_i}^p\|_2)) \\
Y_k^p &= \mathcal{Y}^p(\text{topk}(\|\Phi_{y_i}^p\|_2, \ldots, \|\Phi_{y_i}^p\|_2, \ldots, \|\Phi_{y_i}^p\|_2))
\end{align*}
\]
where topk(·) extracts the indices of the k largest elements of the given input. Here, \( \Phi \) denotes embeddings learned by DGCNN and Transformer.

By aligning only the keypoints, we remove irrelevant points from the two input clouds that are not shared in the partial correspondence. In particular, we can now solve the Procrustes problem that matches keypoints of \( \mathcal{X} \) and \( \mathcal{Y} \). We show in [3.4] that although we do not provide explicit supervision, PRNet still learns how to detect keypoints reasonably.

**Gumbel–Softmax Sampler.** One key observation in ICP and DCP is that (3) usually is not differentiable with respect to the map \( m \) but by definition yields a sharp correspondence between the points in \( \mathcal{X} \) and the points in \( \mathcal{Y} \). In contrast, the smooth function (4) in DCP is differentiable, but in exchange for this differentiability the mapping is blurred. We desire the best of both worlds: A potentially sharp mapping function that admits backpropagation.

To that end, we use Gumbel–Softmax (10) to sample a matching matrix. Using a straight-through gradient estimator, this module is approximately differentiable. In particular, the Gumbel–Softmax mapping function is given by

\[
m^p(x_i, y_j) = \text{one hot} \left[ \text{arg max} \text{softmax}(\Phi_x^p \Phi_y^p + g_{ij}) \right],
\]

where \( (g_{i1}, \ldots, g_{ij}, \ldots, g_{iN}) \) are i.i.d. samples drawn from Gumbel(0, 1). The map in (10) is not differentiable due to the discontinuity of \( \text{arg max} \), but the straight-through gradient estimator yields (biased) subgradient estimates with low variance. Following their methodology, on backward evaluation of the computational graph, we use [4] to compute \( \frac{\partial L}{\partial \theta} \), ignoring the one hot operator and the \( \text{arg max} \) term.

**Actor-Critic Closest Point (ACP).** The mapping functions (4) and (10) have fixed “temperatures,” that is, there is no control over the sharpness of the mapping matrix \( m^p \). In PRNet, we wish to adapt the sharpness of the map based on the alignment of the two shapes. In particular, for low values of \( p \) (the initial iterations of alignment) we may satisfied with high-entropy approximate matchings that obtain a coarse alignment; later during iterative evaluations, we can sharpen the map to align individual pairs of points.

To make this intuition compatible with PRNet’s learning-based architecture, we add a parameter \( \lambda \) to (10) to yield a generalized Gumbel–Softmax matching matrix:

\[
m^p(x_i, y_j) = \text{one hot} \left[ \text{arg max} \text{softmax} \left( \frac{\Phi_x^p \Phi_y^p + g_{ij}}{\lambda} \right) \right] \tag{11}
\]

When \( \lambda \) is large, the map matrix \( m^p \) is smoothed out; as \( \lambda \to 0 \) the map approaches a binary matrix.

It is difficult to choose a single \( \lambda \) that suffices for all \( (\mathcal{X}, \mathcal{Y}) \) pairs; rather, we wish \( \lambda \) to be chosen adaptively and automatically to extract the best alignment for each pair of point clouds. Hence, we use a small network \( \Theta \) to predict \( \lambda \) based on global features \( \Psi_x^p \) and \( \Psi_y^p \) aggregated from \( \Phi_x^p \) and \( \Phi_y^p \) channel-wise by global pooling (averaging). In particular, we take \( \lambda = \Theta(\Psi_x^p, \Psi_y^p) \), where \( \Psi_x^p = \text{avg}_t \Phi_x^p \), and \( \Psi_y^p = \text{avg}_t \Phi_y^p \). In the parlance of reinforcement learning, this choice can be seen as a simplified version of actor-critic method. \( \Phi_x^p \) and \( \Phi_y^p \) are learned jointly with DGCNN, and Transformer; then an actor head outputs a rigid motion, where (11) uses the \( \lambda \) predicted from a critic head.

**Loss Function.** The final loss \( L \) is the summation of several terms \( L_p \), indexed by the number \( p \) of passes through PRNet for the input pair. \( L_p \) consists of three terms: a rigid motion loss \( L^m_p \), a cycle consistency loss \( L^c_p \), and a global feature alignment loss \( L_g \). We also introduce a discount factor \( \gamma < 1 \) to promote alignment within the first few passes through PRNet; during training we pass each input pair through PRNet \( P \) times.

Combining the terms above, we have

\[
L = \sum_{p=1}^{P} \gamma^{p-1} L_p, \quad \text{where} \quad L_p = L^m_p + \alpha L^c_p + \beta L^g_p \tag{12}
\]

The rigid motion loss \( L^m_p \) is,

\[
L^m_p = \| R_{X \rightarrow Y}^p R_{X \rightarrow Y}^p \|_2^2 + \| t_{X \rightarrow Y}^p - t_{X \rightarrow Y}^p \|_2^2 \tag{13}
\]
Table 1: Test on unseen point clouds

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE(R)</th>
<th>RMSE(R)</th>
<th>MAE(R)</th>
<th>R²(R)</th>
<th>MSE(t)</th>
<th>RMSE(t)</th>
<th>MAE(t)</th>
<th>R²(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICP</td>
<td>1134.552</td>
<td>33.683</td>
<td>25.045</td>
<td>-5.696</td>
<td>0.0856</td>
<td>0.293</td>
<td>0.250</td>
<td>-0.037</td>
</tr>
<tr>
<td>Go-ICP</td>
<td>195.985</td>
<td>13.999</td>
<td>3.165</td>
<td>-0.157</td>
<td>0.0011</td>
<td>0.033</td>
<td>0.012</td>
<td>0.987</td>
</tr>
<tr>
<td>FGR</td>
<td>126.288</td>
<td>11.238</td>
<td>2.832</td>
<td>0.256</td>
<td>0.0009</td>
<td>0.030</td>
<td>0.008</td>
<td>0.989</td>
</tr>
<tr>
<td>PointNetLK</td>
<td>280.044</td>
<td>16.735</td>
<td>7.550</td>
<td>-0.654</td>
<td>0.0007</td>
<td>0.027</td>
<td>0.020</td>
<td>0.975</td>
</tr>
<tr>
<td>DCP-v2</td>
<td>45.005</td>
<td>6.709</td>
<td>4.448</td>
<td>0.732</td>
<td>0.0007</td>
<td>0.027</td>
<td>0.020</td>
<td>0.991</td>
</tr>
<tr>
<td>PRNet (Ours)</td>
<td>10.235</td>
<td>3.199257</td>
<td>1.457</td>
<td>0.939</td>
<td>0.0003</td>
<td>0.016</td>
<td>0.010</td>
<td>0.997</td>
</tr>
<tr>
<td>PRNet (Ours*)</td>
<td>15.624</td>
<td>3.953</td>
<td>1.712</td>
<td>0.850</td>
<td>0.0003</td>
<td>0.017</td>
<td>0.011</td>
<td>0.996</td>
</tr>
</tbody>
</table>

Table 2: Test on unseen categories. PRNet (Ours) denotes the model trained on ShapeNetCore and tested on ModelNet40 held-out categories. Others are trained on first 20 ModelNet40 categories and tested on ModelNet40 held-out categories.

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE(R)</th>
<th>RMSE(R)</th>
<th>MAE(R)</th>
<th>R²(R)</th>
<th>MSE(t)</th>
<th>RMSE(t)</th>
<th>MAE(t)</th>
<th>R²(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICP</td>
<td>1217.618</td>
<td>34.894</td>
<td>25.455</td>
<td>-6.253</td>
<td>0.086</td>
<td>0.293</td>
<td>0.251</td>
<td>-0.038</td>
</tr>
<tr>
<td>Go-ICP</td>
<td>157.072</td>
<td>12.533</td>
<td>2.940</td>
<td>0.063</td>
<td>0.0009</td>
<td>0.030</td>
<td>0.010</td>
<td>0.989</td>
</tr>
<tr>
<td>FGR</td>
<td>98.635</td>
<td>9.932</td>
<td>1.952</td>
<td>0.414</td>
<td>0.0014</td>
<td>0.038</td>
<td>0.007</td>
<td>0.983</td>
</tr>
<tr>
<td>PointNetLK</td>
<td>526.401</td>
<td>22.943</td>
<td>9.655</td>
<td>-2.137</td>
<td>0.0037</td>
<td>0.061</td>
<td>0.033</td>
<td>0.955</td>
</tr>
<tr>
<td>DCP-v2</td>
<td>95.431</td>
<td>9.769</td>
<td>6.954</td>
<td>0.427</td>
<td>0.0010</td>
<td>0.034</td>
<td>0.025</td>
<td>0.986</td>
</tr>
<tr>
<td>PRNet (Ours)</td>
<td>24.857</td>
<td>4.986</td>
<td>2.329</td>
<td>0.850</td>
<td>0.0004</td>
<td>0.021</td>
<td>0.015</td>
<td>0.995</td>
</tr>
<tr>
<td>PRNet (Ours*)</td>
<td>15.624</td>
<td>3.953</td>
<td>1.712</td>
<td>0.850</td>
<td>0.0003</td>
<td>0.017</td>
<td>0.011</td>
<td>0.996</td>
</tr>
</tbody>
</table>

Equation (5) gives the “localized” ground truth values for $R^p_{X,Y}$, $t^p_{X,Y}$. Denoting the rigid motion from $Y$ to $X$ in step $p$ as $[R^p_{YX}, t^p_{YX}]$, the cycle consistency loss is

$$L_p^c = \|R^p_{X,Y}R^p_{YX} - I\|^2 + \|t^p_{X,Y} - t^p_{YX}\|^2.$$  \(14\)

Our last loss term is a global feature alignment loss, which enforces alignment of global features $\Psi^p_X$ and $\Psi^p_Y$. Mathematically, the global feature alignment loss is

$$L_p^g = \|\Psi^p_X - \Psi^p_Y\|.$$ \(15\)

This global feature alignment loss also provides signal for determining $\lambda$. When two shapes are close in global feature space, $\lambda$ should be small, yielding a sharp matching matrix; when two shapes are far from each other, $\lambda$ increases and the map is blurry.

4 Experiments

Our experiments are divided into four parts. First, we show performance of PRNet on a partial-to-partial registration task on synthetic data in §4.1. Then, we show PRNet can generalize to real data in §4.2. Third, we visualize the keypoints and correspondences predicted by PRNet in §4.3. Finally, we show a linear SVM trained on representations learned by PRNet can achieve comparable results to supervised learning methods in §4.4.

4.1 Partial-to-Partial Registration on ModelNet40

We evaluate partial-to-partial registration on ModelNet40 \[62\]. There are 12,311 CAD models spanning 40 object categories, split to 9,843 for training and 2,468 for testing. Point clouds are sampled from the CAD models by farthest-point sampling on the surface. During training, a point cloud with 1024 points $X$ is sampled. Along each axis, we randomly draw a rigid transformation; the rotation along each axis is sampled in $[0, 45]$ and translation is in $[0, 0.5, 0.5]$. We apply the rigid transformation to $X$, leading to $Y$. We simulate partial scans of $X$ and $Y$ by randomly placing a point in space and computing its 768 nearest neighbors in $X$ and $Y$ respectively.

We measure mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and coefficient of determination ($R^2$). Angular measurements are in units of degrees. MSE, RMSE and MAE should be zero while $R^2$ should be one if the rigid alignment is perfect. We compare our model to ICP, Go-ICP \[14\], Fast Global Registration (FGR) \[61\], and DCP \[2\].
\begin{table}[h]
\begin{center}
\begin{tabular}{lcccccccc}
Model & $\text{MSE}(R)$ & $\text{RMSE}(R)$ & $\text{MAE}(R)$ & $R^2(R)$ & $\text{MSE}(t)$ & $\text{RMSE}(t)$ & $\text{MAE}(t)$ & $R^2(t)$ \\
ICP & 1229.670 & 35.067 & 25.564 & -6.252 & 0.0860 & 0.294 & 0.250 & -0.045 \\
Go-ICP \cite{11} & 150.320 & 12.261 & 2.845 & 0.112 & 0.0008 & 0.028 & 0.029 & 0.991 \\
FGR \cite{61} & 397.575 & 19.939 & 9.076 & -1.343 & 0.0032 & 0.0572 & 0.032 & 0.960 \\
PointNetLK \cite{1} & 47.378 & 6.883 & 4.534 & 0.718 & 0.0008 & 0.028 & 0.021 & 0.991 \\
DCP-v2 \cite{2} & 18.691 & 4.323 & 2.051 & 0.889 & 0.0003 & 0.017 & 0.012 & 0.995 \\
PRNet (Ours) & & & & & & & & \\
\hline
\end{tabular}
\caption{Test on unseen point clouds with Gaussian noise.}
\end{center}
\end{table}

Table 3: Test on unseen point clouds with Gaussian noise.

**Figure 1** shows the architecture of ACP. We use DGCNN with 5 dynamic EdgeConv layers and a Transformer to learn co-contextual representations of $x$ and $y$. The number of filters in each layer of DGCNN are (64, 64, 128, 256, 512). In the Transformer, only one encoder and one decoder with 4-head attention are used. The embedding dimension is 1024. We train the network for 100 epochs using Adam \cite{63}. The initial learning rate is 0.001 and is divided by 10 at epochs 30, 60, and 80.

**Table 4: ModelNet40: transfer learning**

<table>
<thead>
<tr>
<th>Model</th>
<th>$z/z$ ↑</th>
<th>SO(3)/SO(3) ↑</th>
<th>input size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet \cite{20}</td>
<td>89.2</td>
<td>83.6</td>
<td>2048 × 3</td>
</tr>
<tr>
<td>PointNet++ \cite{21}</td>
<td>89.3</td>
<td>85.0</td>
<td>1024 × 3</td>
</tr>
<tr>
<td>DGCNN \cite{22}</td>
<td>92.2</td>
<td>87.2</td>
<td>1024 × 3</td>
</tr>
<tr>
<td>VoxNet \cite{64}</td>
<td>83.0</td>
<td>73.0</td>
<td>30^3</td>
</tr>
<tr>
<td>SubVolSup \cite{65}</td>
<td>88.5</td>
<td>82.7</td>
<td>30^3</td>
</tr>
<tr>
<td>SubVolSup MO \cite{65}</td>
<td>89.5</td>
<td>85.0</td>
<td>20 × 30^3</td>
</tr>
<tr>
<td>MVCNN 12X \cite{66}</td>
<td>89.5</td>
<td>77.6</td>
<td>12 × 224^2</td>
</tr>
<tr>
<td>MVCNN 80X \cite{66}</td>
<td>90.2</td>
<td>86.0</td>
<td>80 × 224^2</td>
</tr>
<tr>
<td>RotationNet20 \cite{67}</td>
<td>92.4</td>
<td>80.0</td>
<td>20 × 224^2</td>
</tr>
<tr>
<td>Spherical CNNs \cite{68}</td>
<td>88.9</td>
<td>86.9</td>
<td>2 × 64^2</td>
</tr>
<tr>
<td>PRNet (Ours)</td>
<td>85.2</td>
<td>80.5</td>
<td>1024 × 3</td>
</tr>
</tbody>
</table>

Table 4: ModelNet40: transfer learning

**Partial-to-Partial Registration on Unseen Objects.** We first evaluate on the ModelNet40 train/test split. We train on 9,843 training objects and test on 2,468 testing objects. Table 1 shows performance. Our method outperforms its counterparts in all metrics.

**Partial-to-Partial Registration on Unseen Categories.** We follow the same testing protocol as \cite{2} to compare the generalizability of different models. ModelNet40 is split evenly by category into training and testing sets. PRNet and DCP are trained on the first 20 categories, and then all methods are tested on the held-out categories. Table 2 shows PRNet behaves more strongly than others. To further test generalizability, we train it on ShapeNetCore dataset \cite{69} and test on ModelNet40 held-out categories. ShapeNetCore has 57,448 objects, and we do the same preprocessing as on ModelNet40. The last row in Table 2, denoted as PRNet (Ours*), surprisingly shows PRNet performs much better than when trained on ModelNet40. This supports the intuition that data-driven approaches work better with more data.

**Partial-to-Partial Registration on Unseen Objects with Gaussian Noise.** We further test robustness to noise. The same preprocessing is done as in the first experiment, except that noise independently sampled from $\mathcal{N}(0, 0.01)$ and clipped to $[-0.05, 0.05]$ is added to each point. As in Table 3, learning-based methods, including DCP and PRNet, are more robust. In particular, PRNet exhibits stronger performance and is even comparable to the noise-free version in Table 1.

**4.2 Partial-to-Partial on Real Data**

We test our model on the Stanford Bunny dataset \cite{70}. Since the dataset only has 10 real scans, we fine tune the model used in Table 1 for 10 epochs with learning rate 0.0001. For each scan, we generate 100 training examples by randomly transforming the scan in the same way as we do in §4.1. This training procedure can be viewed as inference time fine-tuning, in contrast to optimization-based methods that perform one-time inference for each test case. Figure 2 shows the results. We further test our model on more scans from Stanford 3D Scanning Repository \cite{71} using a similar methodology; Figure 3 shows the registration results.
4.3 Keypoints and Correspondences

We visualize keypoints on several objects in Figure 4 and correspondences in Figure 6. The model detects keypoints and correspondences on partially observable objects. We overlay the keypoints on top of the fully observable objects. Also, as shown in Figure 5, the keypoints are consistent across different views.
4.4 Transfer to Classification

Representations learned by PRNet can be transferred to object recognition. We use the model trained on ShapeNetCore in §4.1 and train a linear SVM on top of the embeddings predicted by DGCNN. In Table 4, we show classification accuracy on ModelNet40. $z/z$ means the model is trained when only azimuthal rotations are present on both during and testing. $\text{SO}(3)/\text{SO}(3)$ means the object is moved with a random motion in $\text{SO}(3)$ during both training and testing. Although other methods are supervised, ours achieves comparable performance.

5 Conclusion

PRNet tackles a general partial-to-partial registration problem, leveraging self-supervised learning to learn geometric priors directly from data. The success of PRNet verifies the sensibility of applying learning to partial matching as well as the specific choice of Gumbel–Softmax, which we hope can inspire additional work linking discrete optimization to deep learning. PRNet is also a reinforcement learning-like framework; this connection between registration and reinforcement learning may provide inspiration for additional interdisciplinary research related to rigid/non-rigid registration.

Our experiments suggest several avenues for future work. For example, as shown in Figure 6, the matchings computed by PRNet are not bijective, evident e.g. in the point clouds of cars and chairs. One possible extension of our work to address this issue is to use Gumbel–Sinkhorn to encourage bijectivity. Improving the efficiency of PRNet when applied to real scans also will be extremely valuable. As described in §4.2, PRNet currently requires inference-time fine-tuning on real scans to learn useful data-dependent representations; this makes PRNet slow during inference. Seeking universal representations that generalize over broader sets of registration tasks will improve the speed and generalizability of learning-based registration. Another possibility for future work is to improve the scalability of PRNet to deal with large-scale real scans captured by LiDAR.

Finally, we hope to find more applications of PRNet beyond the use cases we have shown in the paper. A key direction bridging PRNet to applications will involve incorporating our method into SLAM or structure-from-motion can demonstrate its value for robotics applications and robustness to realistic species of noise. Additionally, we can test the effectiveness of PRNet for registration problems in medical imaging and/or high-energy particle physics.

6 Acknowledgements

The authors acknowledge the generous support of Army Research Office grant W911NF1710068, Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grant IIS-1838071, from an Amazon Research Award, from the MIT-IBM Watson AI Laboratory, from the Toyota-CSAIL Joint Research Center, from a gift from Adobe Systems, and from the Skoltech-MIT Next Generation Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these organizations. The authors also thank members of MIT Geometric Data Processing group for helpful discussion and feedback on the paper.

References


[71] The Stanford 3d scanning repository.