Unsupervised Deep Probabilistic Approach for Partial Point Cloud Registration –Supplementary Material–

A. Appendix

In this supplementary material, we first describe the detailed feature extractors in Sec. A.1, then we provide the details of the Transformer in Sec. A.2, followed by solving Eq. (6) in Sec. A.3. We also give the definitions of evaluation metrics in Sec. A.4. Finally, we provide more registration results in Sec. A.5.

A.1. Feature Extractor

Our UDPReg adopts a KPConv [41]-based encoder-decoder architecture for feature extraction, where we add a lightweight Transformer for context aggregation. The configurations of KPConv and ResBlock are the same as in [19]. On 3DMatch, following [38], we first downsample the input point clouds with a voxel size of 2.5cm, then send the downsampled point clouds into the feature extractor. The detailed network configurations are shown in Table 6.

Table 6. Network architecture for 3DMatch and ModelNet.

Stage	3DMatch	ModelNet
1	KPConv(1→64) ResBlock(64→128)	KPConv(1→256) ResBlock(256→256)
2	ResBlock(64→128, strided) ResBlock(128→256) ResBlock(256→256)	ResBlock(256→512, strided) - -
3	ResBlock(256→256, strided) ResBlock(256→512) ResBlock(512→512)	ResBlock(512→512, strided) ResBlock(512→512) -
4	ResBlock(512→512, strided) ResBlock(512→1024) ResBlock(1024→1024)	ResBlock(1024→1024, strided) - -
5	- -	ResBlock(1024→1024, strided) ResBlock(1024→1024)
6	-	ResBlock(1024→1024, strided)
7	Conv1D(1024→256) Transformer(256→256)	Conv1D(1024→256) Transformer(256→256)
8	NearestUpsampling UnaryConv(1537→512)	NearestUpsampling UnaryConv(1537→512)
9	NearestUpsampling UnaryConv(768→256)	NearestUpsampling UnaryConv(768→512)
10	Linear(512→257)	Linear(256→129)

A.2. Transformer

The transformer is composed of three main components: self-attention, positional encoding, and cross-attention. Geometric self-attention is utilized to capture long-range dependencies, while positional encoding assigns intrinsic geometric properties to each point feature, thereby increasing differentiation among features in areas where they may be indistinct. The cross-attention module leverages the connections between the source and target point clouds, enabling the encoding of contextual information between partially overlapping point clouds. The individual parts will be described in detail below.

Self-Attention. We use the geometric self-attention provided in GeoTransformer [38] for self-attention.

Positional Encoding. Following [31], incorporating a positional encoding approach, which imparts intrinsic geometric attributes to individual point features through the inclusion of unique positional information, improves differentiation among point features in less distinctive regions. To begin with, we choose the $k = 10$ nearest neighbors $\mathcal{K}_i$ of $\bar{\boldsymbol{p}}_i^s$ and calculate the centroid $\bar{\boldsymbol{p}}_c^s = \sum_{i=1}^{\bar{N}_s} \bar{\boldsymbol{p}}_i^s$ of $\bar{\boldsymbol{\mathcal{P}}}^s$, where $\bar{\boldsymbol{p}}_i^s$ and $\bar{\boldsymbol{p}}_j^s$ represent two superpoints of $\bar{\boldsymbol{\mathcal{P}}}^s$. For each $\bar{\boldsymbol{p}}_x^s \in \mathcal{K}_i$, we denote the angle between two vectors $\bar{\boldsymbol{p}}_i^s - \bar{\boldsymbol{p}}_c^s$ and $\bar{\boldsymbol{p}}_x^s - \bar{\boldsymbol{p}}_c^s$ as α_{ix}. The position encoding $\bar{\boldsymbol{g}}_i^s$ of $\bar{\boldsymbol{p}}_i$ is defined as follows:

$$\bar{\boldsymbol{g}}_i^s = \varphi\left(\|\bar{\boldsymbol{p}}_i^s - \bar{\boldsymbol{p}}_c^s\|_2\right) + \max_{x \in \mathcal{K}_i}\{\phi\left(\alpha_{ix}\right)\}, \tag{11}$$

where φ and ϕ are two MLPs, and each MLP consists of a linear layer and one ReLU nonlinearity function.

Cross-Attention. Let ${}^{(l)}\bar{\boldsymbol{\mathcal{F}}}^s$ be the intermediate representation for $\bar{\boldsymbol{\mathcal{P}}}$ at layer l and let ${}^{(0)}\bar{\boldsymbol{\mathcal{F}}}^s = \{\bar{\boldsymbol{g}}_i^s + \bar{\boldsymbol{f}}_i^s\}_{i=1}^{\bar{N}_s}$. We use a multi-attention layer consisting of four attention heads to update the ${}^{(l)}\bar{\boldsymbol{\mathcal{F}}}^s$ via

$$\begin{aligned} &\boldsymbol{S}^s = {}^{(l)}\boldsymbol{W}_1\,{}^{(l)}\bar{\boldsymbol{\mathcal{F}}}^s + {}^{(l)}\boldsymbol{b}_1,\ \boldsymbol{K}^t = {}^{(l)}\boldsymbol{W}_2\,{}^{(l)}\bar{\boldsymbol{\mathcal{F}}}^t + {}^{(l)}\boldsymbol{b}_2, \\ &\boldsymbol{V}^x = {}^{(l)}\boldsymbol{W}_3\,{}^{(l)}\bar{\boldsymbol{\mathcal{F}}}^t + {}^{(l)}\boldsymbol{b}_3,\ \boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{S}^{s\top}\boldsymbol{K}^t}{\sqrt{b}}\right), \\ &{}^{(l+1)}\bar{\boldsymbol{\mathcal{F}}}^s = {}^{(l)}\bar{\boldsymbol{\mathcal{F}}}^s + {}^{(l)}h\left(\boldsymbol{A}\boldsymbol{V}^x\right). \end{aligned} \tag{12}$$

Here, ${}^{(l)}h\left(\cdot\right)$ is a three-layer fully connected network consisting of a linear layer, instance normalization, and a LeakyReLU activation. The same attention module is also simultaneously performed for all points in point cloud $\bar{\boldsymbol{\mathcal{P}}}^t$. The final outputs of attention module are $\bar{\boldsymbol{\mathcal{F}}}^s$ for $\bar{\boldsymbol{\mathcal{P}}}^s$ and $\bar{\boldsymbol{\mathcal{F}}}^t$ for $\bar{\boldsymbol{\mathcal{P}}}^t$. The latent features $\bar{\boldsymbol{\mathcal{F}}}^s$ have the knowledge of $\bar{\boldsymbol{\mathcal{F}}}^t$ and vice versa.

Overlap Score. After computing $\bar{\boldsymbol{\mathcal{F}}}^s$ and $\bar{\boldsymbol{\mathcal{F}}}^t$, a network acts on them to extract overlap scores $\bar{\boldsymbol{O}}^s = \{\bar{\boldsymbol{o}}_i^s \in [0,1]\}_{i=1}^{\bar{N}}$ and $\bar{\boldsymbol{O}}^t = \{\bar{\boldsymbol{o}}_j^t \in [0,1]\}_{j=1}^{\bar{M}} \in [0,1]$ for $\bar{\boldsymbol{\mathcal{P}}}^s$ and $\bar{\boldsymbol{\mathcal{P}}}^t$, respectively, to identify the overlapping regions [19]. The overlap scores and features are sent to the decoder, which outputs the point-wise feature descriptor $\boldsymbol{\mathcal{F}}^s \in \mathbb{R}^{N_s \times d}$ and $\boldsymbol{\mathcal{F}}^t \in \mathbb{R}^{N_t \times d}$ and overlap scores $\boldsymbol{O}^s = \{o_i^s\} \in \mathbb{R}_+^{N_s}$ and $\boldsymbol{O}^t = \{o_j^t\} \in \mathbb{R}_+^{N_t}$. d is the dimension of features.

A.3. Optimization

Now, we introduce how to address the optimization objective presented in Eq. (6) of the main paper:

$$\begin{aligned} &\min_{\boldsymbol{\gamma}} \sum_{i,j} \boldsymbol{\gamma}_{ij} \|\boldsymbol{p}_i - \boldsymbol{\mu}_j\|_2^2, \\ &\text{s.t.}, \sum_i \boldsymbol{\gamma}_{ij} = N\boldsymbol{\pi}_j, \sum_j \boldsymbol{\gamma}_{ij} = 1, \boldsymbol{\gamma}_{ij} \in [0,1]. \end{aligned} \tag{13}$$

The constraint $\sum_j \boldsymbol{\gamma}_{ij}=1$ is imposed based on the property of probability that the sum of all probabilities for all possible events is equal to one. The constraint $\sum_i \boldsymbol{\gamma}_{ij}=N\boldsymbol{\pi}_j$ represents the mixture weights' constraints.

Let $\boldsymbol{\Gamma} = \frac{\boldsymbol{\gamma}}{N}$ with elements defined as $\Gamma_{ij} = \frac{\gamma_{ij}}{N}$. By replacing the variable $\boldsymbol{\gamma}$ with $\boldsymbol{\Gamma}$ in Eq. (13), the joint objective can be formulated as an optimal transport (OT) problem [37] as

$$\min_{\boldsymbol{\Gamma}} \langle \boldsymbol{\Gamma}, \boldsymbol{D} \rangle, \text{ s.t. } \boldsymbol{\Gamma}^\top \mathbf{1}_N = \boldsymbol{\pi}, \boldsymbol{\Gamma}\mathbf{1}_L = \frac{1}{N}\mathbf{1}_N. \tag{14}$$

While the minimization of Eq. (14) can be solved in polynomial time as a linear program, it becomes challenging when dealing with millions of data points and thousands of classes as traditional algorithms do not scale well [11]. To overcome this limitation, we utilize an efficient version of the Sinkhorn-Knopp algorithm [11]. This requires the following regularization term:

$$\begin{aligned} &\min_{\boldsymbol{\Gamma}} \langle \boldsymbol{\Gamma}, \boldsymbol{D} \rangle - \epsilon H(\boldsymbol{\Gamma}), \\ &\text{s.t. } \boldsymbol{\Gamma}^\top \mathbf{1}_N = \boldsymbol{\pi}, \ \ \boldsymbol{\Gamma}\mathbf{1}_L = \frac{1}{N}\mathbf{1}_N, \end{aligned} \tag{15}$$

where $H(\boldsymbol{\Gamma}) = \langle \boldsymbol{\Gamma}, \log \boldsymbol{\Gamma} - 1 \rangle$ represents the entropy of $\boldsymbol{\Gamma}$, and $\epsilon > 0$ is a regularization parameter. When ϵ is very large, optimizing Eq. (15) is equivalent to optimizing Eq. (14), but even for moderate values of ϵ, the objective function tends to have approximately the same optimal solution [11]. Choosing the appropriate value of ϵ involves a trade-off between convergence speed and proximity to the original transport problem [11]. In our scenario, a fixed value of ϵ is suitable since our focus is on obtaining the final clustering and representation learning outcomes rather than solving the transport problem exactly. The solution to Eq. (15) can be expressed as a normalized exponential matrix, as stated in [11],

$$\boldsymbol{\Gamma} = \operatorname{diag}(\boldsymbol{\mu}) \exp\left(\boldsymbol{D}/\epsilon\right) \operatorname{diag}(\boldsymbol{\nu}), \tag{16}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_N)$ and $\boldsymbol{\nu} = (\nu_1, \nu_2, \cdots, \nu_L)$ are renormalization vectors in $\mathbb{R}^N$ and $\mathbb{R}^L$. Iterating the updates via $\boldsymbol{\mu}_i = \left[\exp\left(\boldsymbol{D}/\epsilon\right)\boldsymbol{\nu}\right]_i^{-1}$ and $\boldsymbol{\nu}_j = \left[\exp\left(\boldsymbol{D}/\epsilon\right)^\top \boldsymbol{\mu}\right]_j^{-1}$ with initial values $\boldsymbol{\mu} = \frac{1}{N}\mathbf{1}_N$ and $\boldsymbol{\nu} = \boldsymbol{\pi}$, respectively, yields the vectors $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. Although any distribution can be used for the initialization of $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, setting them as the constraints results in faster convergence [11]. In our experiments, we used 20 iterations as it worked well in practice. After solving Eq. (16), we obtain the probability matrix $\boldsymbol{\gamma}$ as

$$\boldsymbol{\gamma} = N \cdot \boldsymbol{\Gamma}. \tag{17}$$

Eqs. (8), (4), and (5) can be solved in a similar way.

A.4. Metrics

Following Predator [19] and CoFiNet [52], we use three metrics, *Registration Recall* (RR), *Relative Rotation Error* (RRE), and *Relative Translation Error* (RTE), to evaluate the performance of the proposed registration algorithm. RRE and RTE are respectively defined as

$$\begin{aligned} RRE &= \arccos\left(\frac{\mathbf{Tr}\left(\boldsymbol{R}^\top \boldsymbol{R}^\star\right) - 1}{2}\right), \\ RTE &= \|\boldsymbol{t} - \boldsymbol{t}^\star\|_2, \end{aligned} \tag{18}$$

where $\boldsymbol{R}^\star$ and $\boldsymbol{t}^\star$ denote the ground-truth rotation matrix and the translation vector, respectively. *Registration Recall* (RR), the fraction of point cloud pairs whose root mean square error (RMSE) of transformation is smaller than a certain threshold (i.e., $RMSE < 0.2m$). Specifically, we denote the set of ground truth correspondences as $\mathcal{H}$ and the estimated transformation T, their root mean square error are calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{H}|} \sum_{(\boldsymbol{p},\boldsymbol{q}) \in \mathcal{H}} \|T(\boldsymbol{p}) - \boldsymbol{q}\|_2^2}. \tag{19}$$

Follow [19], Chamfer distance (CD) is used to measure the registration quality on ModelNet40. We use the modified Chamfer distance metric:

$$\begin{aligned} CD(\mathcal{P}, \mathcal{Q}) = &\frac{1}{|\mathcal{P}|} \sum_{\boldsymbol{p} \in \mathcal{P}} \min_{\boldsymbol{q} \in \mathcal{Q}} \|T(\boldsymbol{p}) - \boldsymbol{q}\| \\ &+ \frac{1}{|\mathcal{Q}|} \sum_{\boldsymbol{q} \in \mathcal{Q}} \min_{\boldsymbol{p} \in \mathcal{P}} \|T(\boldsymbol{p}) - \boldsymbol{q}\|, \end{aligned} \tag{20}$$

where $\mathcal{P}$ and $\mathcal{Q}$ are input source and target point clouds.

Figure 6. Coloring the points using learned GMM labels.

A.5. More Results

Visualize the Gaussian mixtures. Fig. 6 shows the visual results by coloring the points using GMM labels. We use different colors to differentiate clusters.

KITTI results. Table 7 shows the generalization results from 3DMatch to KITTI. UDPReg outperforms baselines, showing its robustness and generalization.

Table 7. Results of generalization from 3DMatch to KITTI.

Method	RTE(↑)	RRE(↑)	Success(↑)	Time (↓)
Predator	16.5	1.38	46.13	0.44
SGP	13.8	0.49	62.22	**0.12**
UDPReg(Ours)	**8.81**	**0.41**	**64.59**	0.26

Complexity Analysis. $O(N \times L), L < N$ complexity for clustering and $O(N^2)$ for attention represents the memory bottleneck of UDPReg. N, L are point and cluster numbers, respectively. Table 7 reports the inference time on a Tesla V100 GPU (32G) and two Intel(R) 6226 CPUs.