Iterative Distance-Aware Similarity Matrix Convolution with Mutual-Supervised Point Elimination for Efficient Point Cloud Registration

Jiahao Li[1], Changhao Zhang[2], Ziyao Xu[3], Hangning Zhou[3], and Chi Zhang[3]

[1] Washington University in St. Louis, St. Louis, USA
jiahao.li@wustl.edu
[2] Xi'an Jiaotong University, Xi'an, China
cvchanghao@gmail.com
[3] Megvii Inc., Beijing, China
{xuziyao,zhouhangning,zhangchi}@megvii.com

Abstract. In this paper, we propose a novel learning-based pipeline for partially overlapping 3D point cloud registration. The proposed model includes an iterative distance-aware similarity matrix convolution module to incorporate information from both the feature and Euclidean space into the pairwise point matching process. These convolution layers learn to match points based on joint information of the entire geometric features and Euclidean offset for each point pair, overcoming the disadvantage of matching by simply taking the inner product of feature vectors. Furthermore, a two-stage learnable point elimination technique is presented to improve computational efficiency and reduce false positive correspondence pairs. A novel mutual-supervision loss is proposed to train the model without extra annotations of keypoints. The pipeline can be easily integrated with both traditional (e.g. FPFH) and learning-based features. Experiments on partially overlapping and noisy point cloud registration show that our method outperforms the current state-of-the-art, while being more computationally efficient.

Keywords: Point Cloud Registration

1 Introduction

Point cloud registration is an important task in computer vision, which aims to find a rigid body transformation to align one 3D point cloud (source) to another (target). It has a variety of applications in computer vision, augmented reality and virtual reality, such as pose estimation and 3D reconstruction. The most widely used traditional registration method is Iterative Closest Point (ICP) [3], which is only suitable for estimating small rigid transformation. However, in many real world applications, this assumption does not hold. The task of registering two point clouds with large rotation and translation is called global

registration. Some global registration methods [43, 42] are proposed to overcome the limitation of ICP, but are usually very slow compared to ICP.

In recent years, deep learning models have dominated the field of computer vision [15, 13, 31, 8, 9]. Many computer vision tasks are proven to be solved better using data-driven methods based on neural networks. Recently, some learning-based neural network methods for point cloud registration are proposed [1, 35, 36]. They are capable of dealing with large rotation angles, and are typically much faster than traditional global registration methods. However, they still have major drawbacks. For example, DCP [35] assumes that all the points in the source point cloud have correspondences in the target point cloud. Although promising, learning-based point cloud registration methods are far from perfect.

In this paper, we propose the **Iterative Distance-Aware Similarity Matrix Convolution Network (IDAM)**, a novel learnable pipeline for accurate and efficient point cloud registration. The intuition for IDAM is that while many registration methods use local geometric features for point matching, ICP uses the distance as the only criterion for matching. We argue that incorporating both geometric and distance features into the iterative matching process can resolve ambiguity and have better performance than using either of them. Moreover, point matching involves computing a similarity score, which is usually computed using the inner product or $L2$ distance between feature vectors. This simple matching method does not take into consideration the interaction of features of different point pairs. We propose to use a learned module to compute the similarity score based on the entire concatenated features of the two points of interest. These two intuition can be realized using a single learnable **similarity matrix convolution** module that accepts pairwise inputs in both the feature and Euclidean space.

Another major problem for global registration methods is efficiency. To reduce computational complexity, we propose a novel **two-stage point elimination** technique to keep a balance between performance and efficiency. The first point elimination step, **hard point elimination**, independently filters out the majority of individual points that are not likely to be matched with confidence. The second step, **hybrid point elimination**, eliminates correspondence pairs instead of individual points. It assigns low weights to those pairs that are probable to be false positives while solving the absolute orientation problem. We design a novel **mutual-supervision loss** to train these learned point elimination modules. This loss allows the model to be trained end-to-end without extra annotations of keypoints. This two-stage elimination process makes our method significantly faster than the current state-of-art global registration methods.

Our learned registration pipeline is compatible with both learning-based and traditional point cloud feature extraction methods. We show by experiments that our method performs well with both FPFH [27] and Graph Neural Network (GNN) [37, 17, 40] features. We compare our model to other point cloud registration methods, showing that the power of learning is not only restricted to feature extraction, but is also critical for the registration process.

2 Related Work

Local Registration The most widely used traditional local registration method is Iterative Closest Point (ICP) [3]. It finds for each point in the source the closest neighbor in the target as the correspondence. Trimmed ICP [5] extends ICP to handle partially overlapping point clouds. Other methods [26, 28, 4] are mostly variants to the vanilla ICP.

Global Registration The most important non-learning global registration methods is RANSAC [6]. Usually FPFH [27] or SHOT [33] feature extraction methods are used with RANSAC. However, RANSAC is very slow compared to ICP. Fast Global Registration (FGR) [43] uses FPFH features and an alternating optimization technique to speed up global registration. Go-ICP [42] adopts a brute-force branch-and-bound strategy to find the rigid transformation. There are also other methods [20, 11, 41, 14] that utilize a variety of optimization techniques.

Data-driven Registration PointNetLK [1] pioneers the recent learning-based registration methods. It adapts PointNet [22] and the Lucas & Kanade [18] algorithm into a single trainable recurrent deep neural network. Deep Closest Point (DCP) [35] proposed to use a transformer network based on DGCNN [37] to extract features, and train the network end-to-end by back-propagating through the SVD layer. PRNet [36] tries to extend DCP to an iterative pipeline and deals with partially overlapping point cloud registration.

Learning on Point Cloud Recently a large volume of research papers apply deep learning techniques for learning on point clouds. Volumetric methods [21, 45] apply discrete 3D convolution on the voxel representation. OctNet [24] and O-CNN [34] try to design efficient high-resolution 3D convolution using the sparsity property of point clouds. Other methods [38, 16, 32, 19] try to directly define convolution in the continuous Euclidean space, or convert the point clouds to a new space for implementing easy convolution-like operations [25, 30]. Contrary to the effort of adapting convolution to point clouds, PointNet [22] and PointNet++ [23], which use simple permutation invariant pooling operations to aggregate information from individual points, are widely used recently due to their simplicity. [37, 17] view point clouds as graphs with neighbors connecting to each other, and apply graph neural networks (GNN) [40] to extract features.

3 Model

This section describes the proposed IDAM point cloud registration model. The diagram of the whole pipeline is shown in Fig. 1. The details of each component is explained in the following sections.

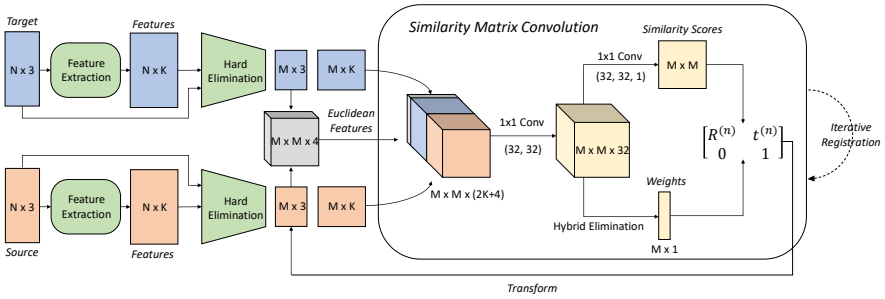

Fig. 1. The overall architecture of the IDAM registration pipeline. Details of hard point elimination and hybrid point elimination are demonstrated in Fig. 2.

3.1 Notation

Here we introduce some notation that will be used throughout the paper. The problem of interest is that for a given source point cloud $\mathcal{S}$ of $N_{\mathcal{S}}$ points and a target point cloud $\mathcal{T}$ of $N_{\mathcal{T}}$ points, we need to find the ground truth rigid body transformation $(\mathbf{R}^*, \mathbf{t}^*)$ that aligns $\mathcal{S}$ to $\mathcal{T}$. Let $\mathbf{p}_i \in \mathcal{S}$ denote the ith point in the source, and $\mathbf{q}_j \in \mathcal{T}$ the jth point in the target.

3.2 Similarity Matrix Convolution

To find the rigid body transformation $\mathbf{R}^*$ and $\mathbf{t}^*$, we need to find a set of point correspondences between the source and target point clouds. Most of the existing methods achieve this by using the inner product (or $L2$ distance) of the point features as a measure of similarity, and directly pick the ones with the highest (or lowest for $L2$) response.

However, this has two shortcomings. First of all, one point in $\mathcal{S}$ may have multiple possible correspondences in $\mathcal{T}$, and one-shot matching is not ideal since the points chosen as correspondences may not be the correct ones due to randomness. Inspired by ICP, we argue that incorporating distance information between points into an iterative matching processing can alleviate this problem, since after an initial registration, correct point correspondences are more likely to be closer to each other.

The second drawback of direct feature similarity computation is that it has limited power of identifying the similarity between two points because the way of matching is the same for different pairs. Instead, we have a learned network that accepts the whole feature vectors and outputs the similarity scores. This way, the network takes into consideration the combinations of features from two points in a pair for matching.

Based on the above intuition, we propose **distance-aware similarity matrix convolution** for finding point correspondences. Suppose we have the geometric features $\mathbf{u}^{\mathcal{S}}(i)$ for $\mathbf{p}_i \in \mathcal{S}$ and $\mathbf{u}^{\mathcal{T}}(j)$ for $\mathbf{q}_j \in \mathcal{T}$, both with dimension K. We form the **distance-augmented feature tensor** at iteration n as

$$\mathbf{T}^{(n)}(i,j) = [\mathbf{u}^{\mathcal{S}}(i); \mathbf{u}^{\mathcal{T}}(j); \|\mathbf{p}_i - \mathbf{q}_j\|; \frac{\mathbf{p}_i - \mathbf{q}_j}{\|\mathbf{p}_i - \mathbf{q}_j\|}] \tag{1}$$

where $[\cdot;\cdot]$ denotes concatenation. The $(2K+4)$-dimensional vector at the (i,j) location of $\mathbf{T}^{(n)}$ is a combination of the geometric and Euclidean features for the point pair $(\mathbf{p}_i, \mathbf{q}_j)$. The 4-dimensional Euclidean features comprise the distance between $\mathbf{p}_i$ and $\mathbf{q}_j$, and the unit vector pointing from $\mathbf{q}_j$ to $\mathbf{p}_i$. Each augmented feature vector in $\mathbf{T}^{(n)}$ encodes the joint information of the local shapes of the two points and their current relative position, which are useful for computing similarity scores at each iteration.

The distance-augmented feature tensor $\mathbf{T}^{(n)}$ can be seen as a $(2K+4)$-channel 2D image. To extract a similarity score for each point pair, we apply a series of 1×1 2D convolution on $\mathbf{T}^{(n)}$ that outputs a single channel image of the same spatial size at the last layer. This is equivalent to applying a multi-layer perceptron on the augmented feature vector at each position. Then we apply a Softmax function on each row of the single channel image to get the **similarity matrix**, denoted as $\mathbf{S}^{(n)}$. $\mathbf{S}^{(n)}(i,j)$ represents the "similarity score" (the higher the more similar) for $\mathbf{p}_i$ and $\mathbf{q}_j$. Each row of $\mathbf{S}^{(n)}$ defines a normalized probability distribution over all the points in $\mathcal{T}$ for some $\mathbf{p} \in \mathcal{S}$. As a result, $\mathbf{S}^{(n)}(i,j)$ can also be interpreted as the probability of $\mathbf{q}_j$ being the correspondence of $\mathbf{p}_i$. The 1×1 convolutions learn their weights using the **point matching loss** described in Section 3.4. They learn to take into account the interaction between the shape and distance information to output a more accurate similarity score compared to simple inner product.

To find the correspondence pairs, we take the argmax of each row of $\mathbf{S}^{(n)}$. The results are a set of correspondence pairs $\{(\mathbf{p}_i, \mathbf{p}'_i) \mid \forall \mathbf{p}_i \in \mathcal{S}\}$, with which we solve the following optimization problem to find the estimated rigid transformation $(\mathbf{R}^{(n)}, \mathbf{t}^{(n)})$

$$\mathbf{R}^{(n)}, \mathbf{t}^{(n)} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{p}'_i\|^2 \tag{2}$$

This is a classical absolute orientation problem [10], which can be efficiently solved with the orthogonal Procrustes algorithm [7, 2] using Singular Value Decomposition (SVD). $\mathbf{R}^{(n)}$ and $\mathbf{t}^{(n)}$ are then used to transform the source point cloud to a new position before entering the next iteration. The final estimate for $(\mathbf{R}^*, \mathbf{t}^*)$ is the composition of the intermediate $(\mathbf{R}^{(n)}, \mathbf{t}^{(n)})$ for all the iterations.

3.3 Two-stage Point Elimination

Although similarity matrix convolution is powerful in terms of matching, it is computationally expensive to apply convolution on the large $N_{\mathcal{S}} \times N_{\mathcal{T}} \times (2K+4)$ tensor, because $N_{\mathcal{S}}$ and $N_{\mathcal{T}}$ are typically more than a thousand. However, if we randomly down-sample the point clouds, the performance of the model would

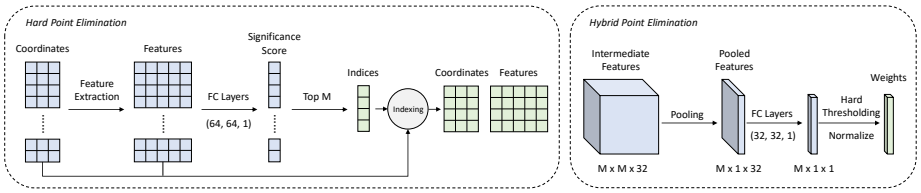

Fig. 2. Comparison of hard point elimination and hybrid point elimination. Hard point elimination filters points based on the features extracted independently for each point, while hybrid point elimination utilizes the joint information of the point pairs to compute weights for the orthogonal Procrustes algorithm.

degrade drastically since many points no longer have correspondences. To tackle this dilemma, we propose a **two-stage point elimination** process. It consists of **hard point elimination** and **hybrid point elimination** (Fig. 2), which targets on improving efficiency and accuracy respectively. While manually labelling keypoints for point clouds is not practical, we propose a **mutual-supervision loss**, that uses the information in the similarity matrices $\mathbf{S}^{(n)}$ to supervise the point elimination modules. The details of the mutual-supervision loss is described in Section 3.4. In this section, we present the point elimination process for inference.

Hard Point Elimination To reduce the computational burden of similarity matrix convolution, we first propose the **hard point elimination** (Fig. 2 Left). Given the extracted local shape features for each point, we apply a multi-layer perceptron on the feature vector, and output a **significance score**. A high score means a more prominent point, such as a corner point, that can be matched with high confidence later (see the Appendix for visualization). It filters out those points in the "flat" regions that are ambiguous during matching. This process is done on individual points, and does not take into account the point pair information as in similarity matrix convolution. As a result, it is efficient to compute the significance score. We preserve the M points for each point cloud with highest significance scores, and eliminate the remaining points. In our network, we choose $M = \lceil \frac{N}{6} \rceil$, where N can be $N_{\mathcal{S}}$ or $N_{\mathcal{T}}$. Denote the set of points in $\mathcal{S}$ preserved by hard point elimination as $\mathcal{B}_{\mathcal{S}}$, and that for the target as $\mathcal{B}_{\mathcal{T}}$.

Hybrid Point Elimination While hard point elimination improves the efficiency significantly, it has negative effect on the performance of the model. The correct corresponding point in the target point cloud for a point in the source point cloud may be mistakenly eliminated in hard elimination. Therefore, similarity matrix convolution will never be able to find the correct correspondence. However, since we always try to find the correspondence with the maximal similarity score for every point in the source, these "negative" point pairs can make

the rigid body transformation obtained by solving Eq. 2 inaccurate. This problem is especially severe when the model is dealing with two point clouds that only partially overlap with each other. In this case, even without any elimination, some points will not have any correspondence whatsoever.

To alleviate this problem, we propose a **hybrid point elimination** (Fig. 2 Right) process, applied after similarity matrix convolution. Hybrid point elimination is a mixture of both hard elimination and soft elimination, and operates on point pairs instead of individual points. It uses a permutation-invariant pooling operation to aggregate information across all possible correspondences for a given point in the source, and outputs the **validity score**, for which a higher score means higher probability of having a true correspondence. Formally, let $\mathbf{F}$ be the intermediate output (see Fig. 1) of the similarity matrix convolution of shape $M \times M \times K'$. Hybrid point elimination first computes the validity score

$$v(i) = \sigma(f(\bigoplus_j(\mathbf{F}(i,j)))) \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, $\bigoplus$ is an element-wise permutation invariant pooling method, such as "mean" or "max", and f is a multi-layer perceptron that takes the pooled features as input and outputs the scores. This permutation invariant pooling technique is used in a variety point cloud processing [22, 23] and graph neural network [37, 17] models. Following [22, 23] we use element-wise max for $\bigoplus$. This way, we have a validity score for each point in the source, and thus for each point pair. It can be seen as the probability that a correspondence pair is correct.

With this validity score, we then compute the **hybrid elimination weights**. The weight for the ith point pair is defined as

$$w_i = \frac{v(i) \cdot \mathbb{1}[\![v(i) \geq \text{median}_k(v(k))]\!]}{\sum_i v(i) \cdot \mathbb{1}[\![v(i) \geq \text{median}_k(v(k))]\!]} \tag{4}$$

where $\mathbb{1}[\![\cdot]\!]$ is the indicator function. What this weighting process does is that it gives 0 weight to those points with lowest validity scores (hard elimination), and weighs the rest proportionally to the validity scores (soft elimination). With this elimination weight vector, we can obtain the $(\mathbf{R}^{(n)}, \mathbf{t}^{(n)})$ with a slightly different objective function from Eq. 2

$$\mathbf{R}^{(n)}, \mathbf{t}^{(n)} = \underset{\mathbf{R},\mathbf{t}}{\text{argmin}} \sum_i w_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{p}'_i\|^2 \tag{5}$$

This can still be solved using SVD with little overhead. Ideally, the hybrid point elimination can eliminate those point pairs that are not correct due to noise and incompletion, giving better performance on estimating $\mathbf{R}^{(n)}$ and $\mathbf{t}^{(n)}$ (see the Appendix for visualization).

3.4 Mutual-Supervision Loss

In this section, we describe in detail the **mutual-supervision loss** that is used to train the network. With this loss, we can train the similarity matrix convolution, along with the two-stage elimination module, without extra annotations of keypoints. The loss is the sum of three parts, which will be explained in the following.

Note that training on all the points during each forward-backward loop is inefficient and unnecessary. However, since hard point elimination does not function properly during training yet, we do not have direct access to $\mathcal{B}_\mathcal{S}$ and $\mathcal{B}_\mathcal{T}$ (see the definitions in Section 3.3). Therefore, we need some way to sample points from the source and the target for training. This sampling technique is described in Section 3.5. In this section we accept that as given, and abuse the notation $\mathcal{B}_\mathcal{S}$ for the **source sampled set** and $\mathcal{B}_\mathcal{T}$ for the **target sampled set**, which both contain the M sampled points for training. Let $\mathbf{p}_i$ denote the ith point in $\mathcal{B}_\mathcal{S}$ and $\mathbf{q}_j$ denote the jth point in $\mathcal{B}_\mathcal{T}$

Point Matching Loss The point matching loss is used to supervise the similarity matrix convolution. It is a standard cross-entropy loss. The point matching loss for the nth iteration is defined as

$$\mathcal{L}_{\text{match}}^{(n)}(\mathcal{S}, \mathcal{T}, \mathbf{R}^*, \mathbf{t}^*) = \frac{1}{M} \sum_{i=1}^{M} -\log(\mathbf{S}^{(n)}(i, j^*)) \cdot \mathbb{1}[\![\|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}_{j^*}\|^2 \leq r^2]\!] \tag{6}$$

where

$$j^* = \underset{1 \leq j \leq M}{\operatorname{argmin}} \|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}_j\|^2 \tag{7}$$

is the index of the point in the target sampled set $\mathcal{B}_\mathcal{T}$ that is closest to the ith point in the source sampled set $\mathcal{B}_\mathcal{S}$ under the ground truth transformation. r is hyper-parameter that controls the minimal radius within which two points are considered close enough. If the distance of $\mathbf{p}_i$ and $\mathbf{q}_{j^*}$ is larger than r, they can not be seen as correspondences, and no supervision signal is applied on them. This happens frequently when the model is dealing with partially overlapping point clouds. The total point matching loss is the average of those for all the iterations.

Negative Entropy Loss This loss is used for training hard point elimination. The problem for training hard point elimination is that we do not have direct access to annotations of keypoints. Therefore, we propose to use a **mutual supervision** technique, which uses the result of the point matching loss to supervise hard point elimination. This mutual supervision is based on the intuition that if a point $\mathbf{p}_i \in \mathcal{B}_\mathcal{S}$ is a prominent point (high significance score),

the probability distribution defined by the ith row of $\mathbf{S}^{(n)}$ should have low entropy because it is confident in matching. On the other hand, the supervision on the similarity matrices has no direct relationship to hard point elimination. Therefore, the **negative entropy** of the probability distribution can be seen as a supervision signal for the significance scores. Mathematically, the **negative entropy loss** for the nth iteration can be defined as

$$\mathcal{L}_{\text{hard}}^{(n)}(\mathcal{S}, \mathcal{T}, \mathbf{R}^*, \mathbf{t}^*) = \frac{1}{M} \sum_{i=1}^{M} |s(i) - \sum_{j=1}^{M} \mathbf{S}^{(n)}(i,j) \log(\mathbf{S}^{(n)}(i,j))|^2 \tag{8}$$

where $s(i)$ is the significant score for the ith point in $\mathcal{B}_{\mathcal{S}}$. Although this loss can be defined for any iteration, we only use the one for first iteration, because in the early stages of registration the shape features are more important than the Euclidean features. We want the hard point elimination module learns to filter points only based on shape information. We cut the gradient flow from the negative entropy loss to $\mathbf{S}^{(n)}$ to prevent interference with the training of similarity matrix convolution.

Hybrid Elimination Loss A similar mutual supervision idea can also be used for training the hybrid point elimination. The difference is that hybrid elimination takes into account the point pair information, while hard point elimination only looks at individual points. As a result, the mutual supervision signal is much more obvious for hybrid point elimination. We simply use the probability that there exists a point in $\mathcal{B}_{\mathcal{T}}$ which is the correspondence of point $\mathbf{p}_i \in \mathcal{B}_{\mathcal{S}}$ as the supervision signal for v_i (validity score). Instead of computing the probability explicitly, the **hybrid elimination loss** for the nth iteration is defined as

$$\mathcal{L}_{\text{hybrid}}^{(n)}(\mathcal{S}, \mathcal{T}, \mathbf{R}^*, \mathbf{t}^*) = \frac{1}{M} \sum_{i=1}^{M} -\mathbb{I}_i \cdot \log(v_i) - (1 - \mathbb{I}_i) \cdot \log(1 - v_i) \tag{9}$$

where

$$\mathbb{I}_i = \mathbb{1}[\![\|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}_{\text{argmax}_j \mathbf{S}^{(n)}(i,j)}\|^2 \leq r^2]\!] \tag{10}$$

In effect, this loss assigns a positive label 1 to those points in $\mathcal{B}_{\mathcal{S}}$ that correctly finds its correspondence, and a negative label 0 to those that do not. In the long run, those point pairs with high probability of correct matching will have higher validity scores.

3.5 Balanced Sampling for Training

In this section, we describe a balanced sampling technique to sample points for training our network. We first sample $\lceil \frac{M}{2} \rceil$ points from $\mathcal{S}$ with the following unnormalized probability distribution

$$p_{\text{pos}}(i) = \mathbb{1}[\![(\min_{\mathbf{q}\in\mathcal{T}} \|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}\|^2) \leq r^2]\!] + \epsilon \tag{11}$$

where $\epsilon = 10^{-6}$ is some small number. This sampling process aims to randomly sample "positive" points from $\mathcal{S}$, in the sense that they indeed have correspondences in the target. It introduces the ϵ to avoid errors when encountering the singularity cases where no points in the source have correspondences in the target.

Similarly, we sample $(M - \lceil \frac{M}{2} \rceil)$ "negative" points from $\mathcal{S}$ using the unnormalized distribution

$$p_{\text{neg}}(i) = \mathbb{1}[\![(\min_{\mathbf{q}\in\mathcal{T}} \|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}\|^2) > r^2]\!] + \epsilon \tag{12}$$

This way, we have a set $\mathcal{B}_\mathcal{S}$ of points of size M, with both positive and negative instances. To sample points from the target, we simply find the closest points of each point from $\mathcal{B}_\mathcal{S}$ in the target

$$\mathcal{B}_\mathcal{T} = \{\operatorname*{argmin}_{\mathbf{q}} \|\mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}\| \mid i \in \mathcal{B}_\mathcal{S}\} \tag{13}$$

This balanced sampling technique randomly samples points from $\mathcal{S}$ and $\mathcal{T}$, while keeping a balance between points that have correspondences and points that do not.

4 Experiments

This section shows the experimental results to demonstrate the performance and efficiency of our method. We also conduct ablation study to show the effectiveness of each component of our model.

4.1 Experimental Setup

We train our model with the Adam [12] optimizer for 40 epochs. The initial learning rate is 1×10^{-4}, and is multiplied by 0.1 after 30 epochs. We use a weight decay of 1×10^{-3} and no Dropout [29]. We use the FPFH implementation from the Open3D [44] library and a very simple graph neural network (GNN) for feature extraction. The details of the GNN architecture are described in the supplementary material. For both FPFH and GNN features, the number of iterations is set to 3.

Following [36], all the experiments are done on the ModelNet40 [39] dataset. ModelNet40 includes 9843 training shapes and 2468 testing shapes from 40 object categories. For a given shape, we randomly sample 1024 points to form a point cloud. For each point cloud, we randomly generate rotations within $[0°, 45°]$

and translation in $[-0.5, 0.5]$. The original point cloud is used as the source, and the transformed point cloud as the target. To generate partially overlap point clouds, we follow the same method as [36], which fixes a random point far away from the two point clouds, and preserve 768 points closest to the far point for each point cloud.

We compare our method to ICP, Go-ICP, FGR, FPFH+RANSAC, PointNetLK, DCP and PRNet. All the data-driven methods are trained on the same training set. We use the same metrics as [36, 35] to evaluate all these methods. For the rotation matrix, the root mean square error (RMSE($\mathbf{R}$)) and mean absolute error (MAE($\mathbf{R}$)) in degrees are used. For the translation vector, the root mean square error (RMSE($\mathbf{t}$)) and mean absolute error (MAE($\mathbf{t}$)) are used.

4.2 Results

In this section, we show the results for three different experiments to demonstrate the effectiveness and robustness of our method. These experimental settings are the same as those in [36]. We also include in the supplementary material some visualization results for these experiments.

Unseen Shapes First, we train our model on the training set of ModelNet40 and evaluate on the test set. Both the training set and test set of ModelNet40 contain point clouds from all the 40 categories. This experiment evaluates the ability to generalize to unseen point clouds. Table 1 shows the results.

Table 1. Results for testing on point clouds of unseen shapes in ModelNet40.

Model	RMSE($\mathbf{R}$)	MAE($\mathbf{R}$)	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
ICP	33.68	25.05	0.29	0.25
FPFH+RANSAC	2.33	1.96	**0.015**	0.013
FGR	11.24	2.83	0.030	0.008
Go-ICP	14.0	3.17	0.033	0.012
PointNetLK	16.74	7.55	0.045	0.025
DCP	6.71	4.45	0.027	0.020
PRNet	3.20	1.45	0.016	0.010
FPFH+IDAM	**2.46**	**0.56**	0.016	**0.003**
GNN+IDAM	2.95	0.76	0.021	0.005

We can see that local registration method ICP performs poorly because the initial rotation angles are large. FPFH+RANSAC is the best performing traditional method, which is comparable to many learning-based methods. Note that both RANSAC and FGR use FPFH methods, and our method with FPFH features outperforms both of them. Neural network models have a good balance between performance and efficiency. Our method outperforms all the other methods with both hand-crafted (FPFH) and learned (GNN) features.

Surprisingly, FPFH+IDAM has better performance than GNN+IDAM. One possibility is that the GNN overfits to the point clouds in the training set, and does not generalize well to unseen shapes. However, as will be shown in later sections, GNN+IDAM is more robust to noise and also more efficient that FPFH+IDAM.

Unseen Categories In the second experiment, we use the first 20 categories in the training set of ModelNet40 for training, and evaluate on the other 20 categories on the test set. This experiment tests the capability to generalize to point clouds of unseen categories. The results are summarized in Table 2. We can see that without training on the testing categories, all the learning-based methods perform worse consistently. Traditional methods are not affected that much as expected. Based on different evaluation metrics, FPFH+RANSAC and FPFH+IDAM are the best performing methods.

Table 2. Results for testing on point clouds of unseen categories in ModelNet40.

Model	RMSE($\mathbf{R}$)	MAE($\mathbf{R}$)	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
ICP	34.89	25.46	0.29	0.25
FPFH+RANSAC	**2.11**	1.82	**0.015**	0.013
FGR	9.93	1.95	0.038	0.007
Go-ICP	12.53	2.94	0.031	0.010
PointNetLK	22.94	9.66	0.061	0.033
DCP	9.77	6.95	0.034	0.025
PRNet	4.99	2.33	0.021	0.015
FPFH+IDAM	3.04	**0.61**	0.019	**0.004**
GNN+IDAM	3.42	0.93	0.022	0.005

Gaussian Noise In the last experiment, we add random Gaussian noise with standard deviation 0.01 to all the shapes, and repeat the first experiment (unseen shapes). The random noise is clipped to $[-0.05, 0.05]$. As shown in Table 3, both traditional methods and IDAM based on FPFH features perform much worse than the noise-free case. This demonstrates that FPFH is not very robust to noise. The performance of data-driven methods are comparable to the noise-free case, thanks to the powerful feature extraction networks. Our method based on GNN features has the best performance compared to others.

4.3 Efficiency

We test the speed of our method, and compare it to ICP, FGR, FPFH+RANSAC, PointNetLK, DCP and PRNet. We use the Open3D implementation of ICP, FGR

Table 3. Results for testing on point clouds of unseen shapes in ModelNet40 with Gaussian noise.

Model	RMSE(**R**)	MAE(**R**)	RMSE(**t**)	MAE(**t**)
ICP	35.07	25.56	0.29	0.25
FPFH+RANSAC	5.06	4.19	0.021	0.018
FGR	27.67	13.79	0.070	0.039
Go-ICP	12.26	2.85	0.028	0.029
PointNetLK	19.94	9.08	0.057	0.032
DCP	6.88	4.53	0.028	0.021
PRNet	4.32	2.05	**0.017**	0.012
FPFH+IDAM	14.21	7.52	0.067	0.042
GNN+IDAM	**3.72**	**1.85**	0.023	**0.011**

and FPFH+RANSAC, and the official implementation of PointNetLK, DCP and PRNet released by the authors. The experiments are done on a machine with 2 Intel Xeon Gold 6130 CPUs and a single Nvidia GeForce RTX 2080 Ti GPU. We use a batch size of 1 for all the neural network based models. The speed is measured in seconds per frame.

We test the speed on point clouds with 1024, 2048 and 4096 points, and the results are summarized in Table 4. It can be seen that neural network based methods are generally faster than traditional methods. When the number of points is small, IDAM with GNN features is only slower than DCP. But as the number of points increases, IDAM+GNN is much faster than all the other methods. Although FPFH+RANSAC has the best performance among non-learning methods, it is also the slowest. Note that our method with FPFH features is 2× to 5× faster than the other two methods (FGR and RANSAC) which also use FPFH.

Table 4. Comparison of speed of different models. IDAM(G) and IDAM(F) represent GNN+IDAM and FPFH+IDAM respectively. RANSAC also uses FPFH. Speed is measured in seconds-per-frame.

	IDAM(G)	IDAM(F)	ICP	FGR	RANSAC	PointNetLK	DCP	PRNet
1024 points	0.026	0.050	0.095	0.123	0.159	0.082	0.015	0.022
2048 points	0.038	0.078	0.185	0.214	0.325	0.085	0.030	0.048
4096 points	0.041	0.175	0.368	0.444	0.685	0.098	0.084	0.312

4.4 Ablation Study

In this section, we present the results of ablation study of IDAM to show the effectiveness of each component. We examine three key components of our model:

distance-aware similarity matrix convolution (denoted as SM), hard point elimination (HA) and hybrid point elimination (HB). We use BS to denote the model that does not contain any of the three components mentioned above. Since hard point elimination is necessary for similarity matrix convolution due to memory constraints, we replace it with random point elimination in BS. We use inner-product of features in BS when similarity matrix convolution is disabled. As a result, BS is just a simple model that uses the inner-product of features as similarity scores to find correspondences, and directly solves the absolute orientation problem (Eq. 2). We add the components one by one and compare their performance for GNN features. We conducted the experiments under the settings of "unseen categories" as described in Section 4.2. The results are summarized in Table 5.

It can be seen that even with random sampling, similarity matrix convolution already outperforms the baseline (BS) by a large margin. The two-stage point elimination (HA and HB) further boosts the performance significantly.

Table 5. Comparison of the performance of different model choices for IDAM. These experiments examine the effectiveness of similarity matrix convolution (SM), hard point elimination (HA) and hybrid point elimination (HB).

Model	RMSE($\mathbf{R}$)	MAE($\mathbf{R}$)	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
BS	7.77	5.33	0.055	0.047
BS+SM	5.08	3.58	0.056	0.042
BS+HA+SM	4.31	2.89	0.029	0.019
BS+HA+SM+HB	**3.42**	**0.93**	**0.022**	**0.005**

5 Conclusions

In this paper, we propose a novel data-driven pipeline named IDAM for partially overlapping 3D point cloud registration. We present a novel distance-aware similarity matrix convolution to augment the network's ability of finding correct correspondences in each iteration. Moreover, a novel two-stage point elimination method is proposed to improve performance while reducing computational complexity. We design a mutual-supervised loss for training IDAM end-to-end without extra annotations of keypoints. Experiments show that our method performs better than the current state-of-the-art point cloud registration methods and is robustness to noise.

Acknowledgements This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700800.

References

1. Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S.: Pointnetlk: Robust & efficient point cloud registration using pointnet. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7163–7172 (2019)
2. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-d point sets. IEEE Transactions on pattern analysis and machine intelligence (5), 698–700 (1987)
3. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. International Society for Optics and Photonics (1992)
4. Bouaziz, S., Tagliasacchi, A., Pauly, M.: Sparse iterative closest point. In: Computer graphics forum. vol. 32, pp. 113–123. Wiley Online Library (2013)
5. Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P.: The trimmed iterative closest point algorithm. In: Object recognition supported by user interaction for service robots. vol. 3, pp. 545–548. IEEE (2002)
6. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
7. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU press (2012)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
10. Horn, B.K.: Closed-form solution of absolute orientation using unit quaternions. Josa a **4**(4), 629–642 (1987)
11. Izatt, G., Dai, H., Tedrake, R.: Globally optimal object pose estimation in point clouds with mixed-integer programming. In: Robotics Research, pp. 695–710. Springer (2020)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
14. Le, H.M., Do, T.T., Hoang, T., Cheung, N.M.: Sdrsac: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 124–133 (2019)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
16. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in neural information processing systems. pp. 820–830 (2018)
17. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8895–8904 (2019)
18. Lucas, B.D., Kanade, T., et al.: An iterative image registration technique with an application to stereo vision (1981)

19. Mao, J., Wang, X., Li, H.: Interpolated convolutional networks for 3d point cloud understanding. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1578–1587 (2019)
20. Maron, H., Dym, N., Kezurer, I., Kovalsky, S., Lipman, Y.: Point registration via efficient convex relaxation. ACM Transactions on Graphics (TOG) **35**(4), 1–12 (2016)
21. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 922–928. IEEE (2015)
22. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)
24. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3577–3586 (2017)
25. Rippel, O., Snoek, J., Adams, R.P.: Spectral representations for convolutional neural networks. In: Advances in neural information processing systems. pp. 2449–2457 (2015)
26. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: Proceedings Third International Conference on 3-D Digital Imaging and Modeling. pp. 145–152. IEEE (2001)
27. Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In: 2009 IEEE international conference on robotics and automation. pp. 3212–3217. IEEE (2009)
28. Segal, A., Haehnel, D., Thrun, S.: Generalized-icp. In: Robotics: science and systems. vol. 2, p. 435. Seattle, WA (2009)
29. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
30. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2530–2539 (2018)
31. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
32. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6411–6420 (2019)
33. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: European conference on computer vision. pp. 356–369. Springer (2010)
34. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions on Graphics (TOG) **36**(4), 1–11 (2017)
35. Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3523–3532 (2019)

36. Wang, Y., Solomon, J.M.: Prnet: Self-supervised learning for partial-to-partial registration. In: Advances in Neural Information Processing Systems. pp. 8812–8824 (2019)
37. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 1–12 (2019)
38. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9621–9630 (2019)
39. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
40. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
41. Yang, H., Carlone, L.: A polynomial-time solution for robust registration with extreme outlier rates. arXiv preprint arXiv:1903.08588 (2019)
42. Yang, J., Li, H., Jia, Y.: Go-icp: Solving 3d registration efficiently and globally optimally. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1457–1464 (2013)
43. Zhou, Q.Y., Park, J., Koltun, V.: Fast global registration. In: European Conference on Computer Vision. pp. 766–782. Springer (2016)
44. Zhou, Q.Y., Park, J., Koltun, V.: Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847 (2018)
45. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4490–4499 (2018)