Iterative BTreeNet: Unsupervised Learning for Large and Dense 3D Point Cloud Registration

Long Xi[a], Wen Tang[a,*], Tao Xue[b], TaoRuan Wan[c]

[a]*Department of Creative Technology, Bournemouth University, Poole, UK, BH12 5BB.*
[b]*Department of Computer Science, Xi'an Polytechnic University, Xi'an, China, 710048.*
[c]*School of Informatics, University of Bradford, Bradford, UK, BD7 1DP*

Abstract

3D point cloud registration is a computational process that aligns two 3D point clouds through transformation. i.e. finding matching translation and rotation. Existing state-of-the-art learning-based methods require ground-truth transformation as supervision and often perform poorly in dealing with partial point clouds and large scenes that are not trained, resulting in poor generalization for neural networks.

In this paper, we propose a novel unsupervised deep learning network - *Binary Tree Network (BTreeNet)* that consists of a novel forward propagation, which learns features for the rotation separately from the translation and avoids the interference between the estimations of rotation and translation in one single matrix. We then propose an *Iterative Binary Tree Network (IBTreeNet)* to continuously improve the registration accuracy for large and dense 3D point clouds. The Chamfer Distance and the Earth Mover's Distance are adopted as the loss function for unsupervised learning. We show that BTreeNet and IBTreeNet outperform state-of-the-art learning-based and traditional methods on partial and noisy point clouds without training them in such scenarios. Most importantly, the proposed methods exhibit remarkable generalization and robustness to unseen large and dense scenes that are never trained.

*Corresponding author

Email addresses: `lxi@bournemouth.ac.uk` (Long Xi), `wtang@bournemouth.ac.uk` (Wen Tang), `xuetao@xpu.edu.cn` (Tao Xue), `t.wan@bradford.ac.uk` (TaoRuan Wan)

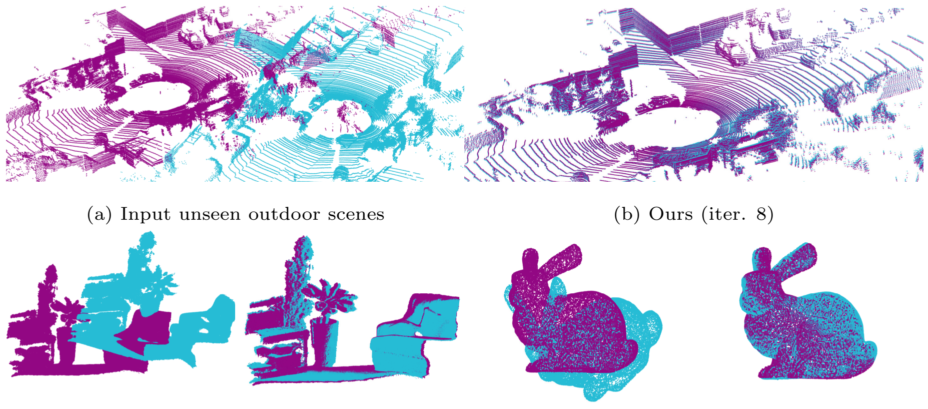
Keywords: 3D Point Cloud Registration, Hierarchical Tree, Computer Vision, Deep Learning

1. Introduction

Point cloud rigid registration is a task that aligns two point clouds, captured by various sensor technologies (i.e. laser and RGB-D scanners and depth cameras), by estimating the rigid transformation between them. Point cloud registration is a well-known problem and has been used in many computer vision applications, for example, 3D reconstruction [1, 2, 3] and localization [4, 5, 6].

Many research attempts have considered 3D point cloud registration as an optimization problem. By solving a least-squares problem to update the alignment and finding correspondences between the 3D point clouds, Iterative Closest Point (ICP) [7] is the most popular 3D point cloud registration method. Normal Distribution Transformation (NDT) [8] uses standard optimization techniques applied to statistical models of 3D point clouds for 3D point cloud registration. Coherent Point Drift (CPD) [9] treats 3D point cloud registration as a probability density estimation problem by maximizing the likelihood. However, these methods [7, 8, 9] are sensitive to the initialization of 3D point clouds and can be computationally expensive and time-consuming.

With regard to deep learning-based approaches, PointNetLK [10], Deep Closest Point (DCP) [11], Robust Point Matching (RPM) [12], Feature-metric Registration (FMR) [13], Deep Gaussian Mixture Registration (DeepGMR) [14] and Robust Graph Matching (RGM) [15] are proposed based on the 3D point cloud processing network PointNet [16] for rigid 3D point cloud registration. PointNetLK [10], as the first supervised deep learning network for rigid 3D point cloud registration, combines a PointNet-based encoder [16] with a traditional Lucas & Kanade (LK) [17] algorithm to align two 3D point clouds. DCP [11] utilizes a PointNet-based encoder [16] and an attention module [18] to extract features of 3D point clouds and generates the transformation matrix through a differentiable Singular Value Decomposition (SVD). RPM [12] estimates point


(a) Input unseen outdoor scenes (b) Ours (iter. 8)

(c) Input unseen indoor scenes (d) Ours (iter. 7) (e) Input unseen shapes (f) Ours (iter. 5)

Figure 1: 3D point cloud registration on the unseen KITTI [20], 3DMatch [21] and Stanford Bunny [22] datasets that are not trained. IBTreeNet can iteratively align the source 3D point clouds to the template, even though it was not trained on these scenes and shapes. Refer to Section 4.6 for more details.

correspondences between two 3D point clouds through a DGCNN-based [19] feature extraction module and generates the transformation matrix similarly to DCP [11]. RGM [15] transforms point clouds into graphs and learns point and graph features to calculate the point correspondences. Similarly to DCP [11] and RPM [12], the transformation matrix in RGM [15] is also estimated from a differentiable SVD. FMR [13] presents a feature-metric point cloud registration network, which enforces the optimisation of registration by minimizing a feature-metric projection error without correspondences. DeepGMR [14] is the first learning-based 3D point cloud registration network that explicitly leverages a probabilistic registration paradigm by formulating registration as the minimization of KL-divergence between two probability distributions modelled as mixtures of Gaussians.

These state-of-the-art learning-based methods, PointNetLK [10], DCP [11], RPM [12], FMR [13], DeepGMR [14] and RGM [15], however, share three common problems. First, the ground-truth transformation matrix or point-to-point correspondences are used to supervise the training process in PointNetLK [10], DCP [11], RPM [12], DeepGMR [14] and RGM [15]. Second, these methods [10, 11, 12, 14, 15] suffer on partial 3D point clouds without training in this scenario (Section 4.4), which shows the poor generalization ability of these networks. Third, these methods [10, 11, 12, 13, 15] often perform poorly in dealing with large and dense scenes and shapes that are not trained, resulting in misalignment between two 3D point clouds (see Section 4.6).

In this paper, we propose a novel hierarchical binary tree-based unsupervised network - *BTreeNet* for large and dense 3D point cloud registration. Unlike PointNetLK [10], DCP [11], RPM [12], FMR [13], DeepGMR [14] and RGM [15], BTreeNet consists of a novel forward propagation that learns features for the rotation separately from the translation and estimates the rotation matrix separately from the translation matrix, as shown in Figure 2. Our BTreeNet avoids the interference between the feature extraction of rotation and translation. Specifically, the root node of the tree is a global feature vector generated from a PointNet-based encoder [16], and the BTreeNet follows a binary tree structure that has a left sub-tree and a right sub-tree. The left sub-tree learns features for rotation and the right sub-tree learns features for translation. The left leaf node estimates the rotation matrix, and the right leaf node estimates the translation matrix. The Chamfer Distance [23] is adopted as the loss function for unsupervised learning, which minimizes the Euclidean distance between two 3D point clouds. We further add the Earth Mover's Distance [24] as the second term to maximize the distribution similarity between two 3D point clouds. Both the Chamfer Distance and the Earth Mover's Distance take raw 3D point clouds as input and do not need the ground-truth transformation matrix.

To continuously improve the registration accuracy between two 3D point clouds, we then propose an *Iterative Binary Tree Network (IBTreeNet)*, which iteratively rotates and translates the registration results of BTreeNet to the target 3D point cloud through the reuse of the trained IBTreeNet model. Note that IBTreeNet has an identical architecture to BTreeNet, but it is trained based on the registration results of a trained BTreeNet model. The objective of IBTreeNet is to extract features of the rotated and translated 3D point cloud from the BTreeNet for the next alignment iteration. Once trained, IBTreeNet can be used repeatedly and iteratively to improve the registration accuracy between two 3D point clouds. Our IBTreeNet exhibits remarkable generalization and robustness to unseen large scenes and shapes that are never trained, as shown in Figure 1. This generalization and robustness performance can be attributed to our proposed forward propagation that avoids the interference between the feature extraction of rotation and translation.

The main contributions of this paper are as follows:

- A BTreeNet with a novel forward propagation based on the hierarchical binary tree is proposed to align two 3D point clouds, which learns features for the rotation separately from the translation and estimates the rotation matrix separately from the translation matrix.
- A novel IBTreeNet is proposed to continuously improve the registration accuracy, which iteratively rotates and translates the source 3D point cloud to the target.
- We adopt the Chamfer Distance and the Earth Mover's Distance as the loss function for unsupervised learning of 3D point cloud registration.
- BTreeNet and IBTreeNet are tolerant to partial overlap, noise and large scenes without training them in such scenarios, indicating the remarkable generalization ability.

2. Related Work

Research on 3D point cloud registration can be categorized into two classes of approaches: traditional or deep learning-based methods. Traditional 3D point cloud registration methods consider the registration an optimization problem, applying the least square regression or maximizing the likelihood of a probability density function. Deep learning-based methods are successful at learning 3D point cloud representations and features by estimating the rigid transformation in the alignment. Our work belongs to deep learning-based methods for 3D point cloud registration.

Traditional 3D Point Cloud Registration Methods: Iterative closest point (ICP) [7] and its variants [25, 26] are well-known traditional methods for 3D point cloud registration by finding point cloud correspondences and solving a least-squares problem to update the alignment. Normal Distribution Transformation (NDT) [8] uses the statistical models of 3D point clouds in the alignment. The key element in NDT [8] is its representation of the 3D point clouds. Instead of using each individual point in the 3D point cloud, NDT [8] converts the 3D point clouds into voxel grids. The grids are represented by a combination of normal distributions, describing the probability of finding a point at a certain position. NDT [8] uses the representation of normal distributions to apply standard numerical optimization methods for registration. Coherent Point Drift (CPD) [9] considers the alignment of two 3D point clouds a probability density estimation problem, where one 3D point cloud represents the Gaussian Mixture Model (GMM) centroids that need to align the other 3D point cloud. CPD [9] moves the GMM centroids coherently as a group to the other 3D point cloud by maximizing the likelihood. However, ICP-based methods [7, 25, 26], NDT [8] and CPD [9] are time-consuming and prone to local minima when two 3D point clouds whose initial positions are far from aligned.

Discriminative Optimization (DO) [27] and its variant Reweighted Discriminative Optimization (RDO) [28] are the supervised sequential update methods that learn the update steps for solving the least-squares problem to obtain the transformation matrix. The learning processes of these optimization methods [27, 28] focus on the sequence of update maps for each individual 3D point cloud and need to be retrained on each individual data, whereas deep learning-based methods [10, 11, 12, 13, 14, 15] learn the generalized features from a large 3D point cloud dataset, and the trained features can be used to unseen 3D point clouds that are not trained.

Deep Learning-based Methods on 3D Point Cloud Registration: PointNet [16] is the first deep neural network to process raw 3D point cloud on 3D point cloud processing. The basic idea of PointNet [16] is to learn a spatial encoding of each point and then aggregate all individual point features to a global feature vector of a 3D point cloud. PointNet [16] has been proven to be useful for tasks including 3D point cloud classification [16, 29, 30, 19], object recognition [31, 32], object detection [33, 34], object segmentation [16, 29, 35, 36, 37], object reconstruction [38, 39], shape completion [3, 40, 41] and registration [42, 10, 11, 12, 13, 14, 15].

Recently proposed PointNetLK [10] is a pioneer in the task of 3D point cloud registration. PointNetLK [10] combines a deep learning method PointNet [16] and a traditional registration method Lucas-Kanade algorithm [17] to achieve features automatically and minimize the distances between the global feature descriptors in the alignment. DCP [11] utilizes DGCNN [19] and an attention module [18] to extract features of two 3D point clouds and replaces the Lucas-Kanade [17] algorithm in PointNetLK [10] with a proposed differentiable Singular Value Decomposition (SVD) module to reduce feature dimension. The SVD module in DCP [11] estimates a transformation matrix with the size of 7, where the first three output values represent the translation matrix and the last four values represent the rotation quaternion. RPM [12] uses raw 3D point clouds and normals as input for the DGCNN-based [19] feature extraction module to estimate point correspondences between two 3D point clouds. Similarly to DCP [11], the SVD module at the end of the network estimates a transformation matrix with the size of 7 from the point correspondences. RGM [15] transforms 3D point clouds into graphs and learns point and graph features via a graph feature extractor to calculate the point correspondences. Similarly to DCP [11] and RPM [12], the transformation matrix in RGM [15] is also estimated from a differentiable SVD. FMR [13] uses the Chamfer distance [23] as a loss function for unsupervised learning and proposes a feature-metric projection error for updating the transformation parameters during each iteration. DeepGMR [14] proposes a network that extracts pose-invariant correspondences between 3D point clouds and Gaussian Mixture Model (GMM) parameters. Two differentiable compute blocks are proposed in DeepGMR [14] to recover the optimal transformation from matched GMM parameters, which achieves favourable performance on 3D point clouds with point-to-point correspondences and large transformations, respectively.

However, these state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] learn the rotation and translation features together and generate the rotation and the translation in one matrix. As a result, the learning of rotation features and translation features interfere with each other, which leads to lower precision of registration results. Moreover, these methods use the ground-truth transformation or point correspondence matrix as supervision. Our work avoids the interference between the feature extraction of rotation and translation and does not need the ground-truth transformation matrix as supervision. The registration results of our BTreeNet and IBTreeNet have been compared with traditional methods [7, 8, 9] and state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15]. The comparison experiments are evaluated on testing datasets, including clear data (Section 4.3), partially visible data (Section 4.4), data with Gaussian noise (Section 4.5) and data with large rotations (Section 4.7). Moreover, the comparison experiments are also tested on unseen scenes and shapes that are not trained to evaluate the generalization and robustness of each method (Section 4.6).

3. Methods

Let P_T and P_S define the target and the source 3D point clouds, respectively, where each point in a 3D point cloud is defined as $P_i = (x, y, z)$. The purpose of our method is to estimate the rigid transformation that best rotates and translates P_S to P_T. The rigid transformation includes a rotation matrix R and a translation matrix t, where $R \in SO(3)$ and $t \in \mathbb{R}^3$. BTreeNet (Section 3.1), as shown in Figure 2, is used to estimate R and t that align P_T and P_S. An iterative BTreeNet (IBTreeNet) (Section 3.2), as shown in Figures 3 and 4, is to iteratively improve the registration accuracy between P_T and P_S. The BTreeNet and IBTreeNet are trained using a Chamfer distance [23] and a Earth Mover's

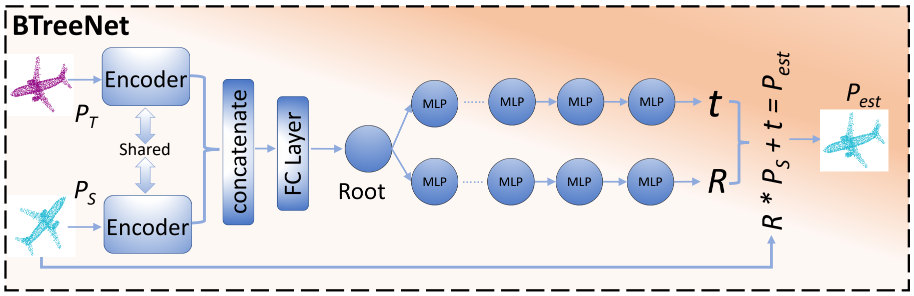


Figure 2: BTreeNet architecture. The source and the target 3D point clouds are given as input through a shared PointNet-based encoder. The global features from the encoder are concatenated and provided as an input to a fully connected layer to achieve the root node of the binary tree. The binary tree learns features for rotation separately form the translation through MLP models and generates the rotation matrix separately from the translation matrix to align two 3D point clouds.

Distance [24] as loss function (Section 3.3) for unsupervised learning.

3.1. BTreeNet

The novelty of our BTreeNet compared to the latest state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] is the hierarchical binary tree-based forward propagation that leans features for the rotation separately from the translation and estimates the rotation matrix separately from the translation matrix. Specifically, in these state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15], the combination of rotation and translation in the output layer makes the network deal with two tasks (e.g. estimating the rotation and the translation) simultaneously. Thus, the two tasks interfere with each other on feature learning based on all different initial rotations along any arbitrary axes and any random translations. As a result, these methods converge to a local optimum, which leads to lower precision of registration results.

Let $P_T \in \mathbb{R}^{N\times 3}$ and $P_S \in \mathbb{R}^{N\times 3}$ are two rigid 3D point clouds that need to be aligned. Both P_T and P_S are given as input to a shared PointNet-based encoder that consists of several multi-layer perception (MLP) models and a symmetric max-pooling function at the end to extract global features. The MLP models extract point-wise features $f_T \in \mathbb{R}^{N \times d}$ and $f_S \in \mathbb{R}^{N \times d}$, where d is the dimension for point-wise features. Weights and biases are shared in MLP models for P_T and P_S. The symmetric max-pooling function aggregates point-wise features f_T and f_S to extract the global features $F_T \in \mathbb{R}^{1 \times d}$ and $F_S \in \mathbb{R}^{1 \times d}$, as defined in Equation 1.

$$F = \underset{p_i \in P}{MAX} \{h(p_1), ..., h(p_n)\} \, (i = 1, ..., n) \tag{1}$$

where F represents either F_T or F_S. p_i is a point in either P_T or P_S. h is the MLP models. The MAX represents the max-pooling that returns a new vector of the element-wise maximum and guarantees that the input 3D point clouds are invariant to any permutations.

The state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] adopt the standard forward propagation, as defined in Equation 2, to learn the features of rotation and translation from the global features F in the same neuron at each layer. The final output in these methods is a transformation matrix with the size of 7, where the first three output values represent the translation matrix and the last four values represent the rotation quaternion.

$$z_i^{(l+1)} = \boldsymbol{w}_i^{(l+1)} y_i^{(l)} + \boldsymbol{b}_i^{(l+1)}; \qquad y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{2}$$

where l indexes the hidden layer and i indexes the hidden neuron in each layer. $y_i^{(l)}$ is the i-th features of P_T and P_S at the layer l. $\boldsymbol{w}_i^{(l+1)}$ and $\boldsymbol{b}_i^{(l+1)}$ denote the i-th weight and bias at layer $l+1$. $z_i^{(l+1)}$ denotes the i-th feature vector of inputs at the layer $l+1$. $f(\cdot)$ is any activation function, e.g. ReLu.

Unlike PointNetLK [10], DCP [11], RPM [12], FMR [13], DeepGMR [14] and RGM [15], our proposed novel forward propagation, as defined in Equation 3, aims to learn features for the rotation separately from the translation and avoids the interference of feature extraction between them. Thus, a hierarchical binary tree based structure is proposed for the novel forward propagation. The global features of $F_T \in \mathbb{R}^{1 \times d}$ and $F_S \in \mathbb{R}^{1 \times d}$ are concatenated and given as an input to a fully connected layer to achieve the root node $F_{root} \in \mathbb{R}^{2 \times d}$ of the binary

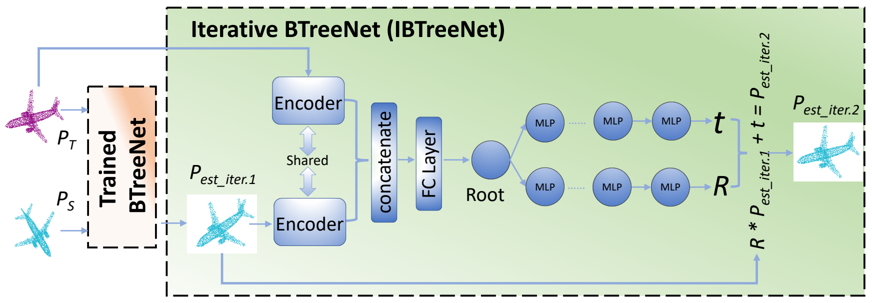


Figure 3: Iterative BTreeNet training mode. The BTreeNet needs to be completely trained before training IBTreeNet. The input 3D point clouds are the P_{est} from the trained BTreeNet and the P_T. The architecture of IBTreeNet is identical to BTreeNet.

tree. The binary tree has two sub-trees, including left and right sub-trees.

$$\begin{aligned} F_{lst} &= F_{root_{idx}}(idx = 0, ..., d); \quad & F_{rst} &= F_{root_{idx}}(idx = d, ..., 2d) \\ z_{i\_lst}^{(l+1)} &= \boldsymbol{w}_{i\_lst}^{(l+1)} F_{lst}^{(l)} + \boldsymbol{b}_{i\_lst}^{(l+1)}; \quad & y_{i\_lst}^{(l+1)} &= f(z_{i\_lst}^{(l+1)}) \\ z_{i\_rst}^{(l+1)} &= \boldsymbol{w}_{i\_rst}^{(l+1)} F_{rst}^{(l)} + \boldsymbol{b}_{i\_rst}^{(l+1)}; \quad & y_{i\_rst}^{(l+1)} &= f(z_{i\_rst}^{(l+1)}) \end{aligned} \tag{3}$$

where $F_{lst} \in \mathbb{R}^{1\times d}$ and $F_{rst} \in \mathbb{R}^{1\times d}$ represent the features are given as input to the left sub-tree and right sub-tree, respectively. l indexes the hidden layer of the binary tree and i indexes the hidden neuron in each layer. $\boldsymbol{w}_{i\_lst}^{(l+1)}$ and $\boldsymbol{b}_{i\_lst}^{(l+1)}$ denote the i-th weight and bias for left sub-tree at layer $l+1$, and $\boldsymbol{w}_{i\_rst}^{(l+1)}$ and $\boldsymbol{b}_{i\_rst}^{(l+1)}$ denote the i-th weight and bias for right sub-tree at layer $l+1$. $z_{i\_lst}^{(l+1)}$ denotes the i-th feature vector of inputs for left sub-tree at the layer $l+1$, and $z_{i\_rst}^{(l+1)}$ denotes the i-th feature vector of inputs for right sub-tree at the layer $l+1$. $f(\cdot)$ is any activation function, e.g. ReLu. $y_{i\_lst}^{(l+1)}$ denotes the i-th feature vector of left sub-tree outputs at the layer $l+1$, and $y_{i\_rst}^{(l+1)}$ denotes the i-th feature vector of right sub-tree outputs at the layer $l+1$.

The final output in left sub-tree $y_{i\_lst}^{(l+1)}$ is a rotation quaternion q with the size of 1×4, and the final output in right sub-tree $y_{i\_rst}^{(l+1)}$ is a translation matrix t with the size of 1×3. The rotation quaternion q, defined by Equation 4, is transformed into a rotation matrix R with the size of 3×3, as defined in

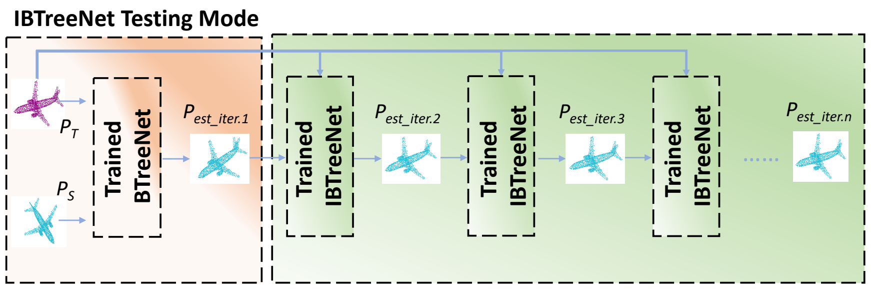


Figure 4: Iterative BTreeNet testing mode. Once BTreeNet and IBTreeNet have been trained, the trained IBTreeNet model can be repeatedly used to iteratively rotates and translates P_S to P_T.

Equation 5. Finally, the P_S is rotated and translated using $P_{est} = R * P_S + t$.

$$\begin{aligned} & q = [q_0, q_1, q_2, q_3]^T \\ & q_0 = cos\frac{\theta}{2}; q_1 = n_x sin\frac{\theta}{2}; q_2 = n_y sin\frac{\theta}{2}; q_3 = n_z sin\frac{\theta}{2} \\ & n = [n_x, n_y, n_z]^T \end{aligned} \tag{4}$$

where θ is the rotation angle (i.e. 45 degrees) and n is the axis of rotation.

$$R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \tag{5}$$

3.2. Iterative BTreeNet

Although three random Euler angles and translations on each axis are sampled and applied to source point clouds during the training, networks still cannot extract the generalized and effective features for every pose of point cloud pairs. As shown in Figure 11, eight iterations have been applied for all networks to reuse their trained models repeatedly for further alignments. These networks cannot achieve accurate registration in the following iterations once it fails to align two 3D point clouds in the first iteration, which indicates that the trained models cannot extract the generalized and effective features of P_T and P_{est} for the next alignment. Refer to Section 4.6.1 for more details. Thus, further alignments are needed to improve the registration accuracy.

To extract effective features of P_T and P_{est} for further alignment and continuously improve the registration accuracy in the following iterations, an iterative BTreeNet (IBTreeNet) is trained based on the result of BTreeNet, as shown in Figure 3. The architecture of IBTreeNet is the same as BTreeNet, but they are trained under different conditions. The differences between BTreeNet and IBTreeNet can be divided into two parts, including the training and testing processes. During the training, IBTreeNet is trained based on the registration results from a pre-trained BTreeNet model. Thus BTreeNet needs to be trained in advance before training IBTreeNet. As shown in Figure 3, the first transformation for P_S from the trained BTreeNet model gives a transformed 3D point cloud $P_{est\_iter.1}$. The transformed point cloud $P_{est\_iter.1}$ and the target 3D point cloud P_T are given as input to IBTreeNet that learns features of P_T and $P_{est\_iter.1}$ for the next alignment. Thus, two iterations are adopted in the training process. During the testing, the trained BTreeNet is used for the first iteration, and the trained IBTreeNet can be used repeatedly and iteratively to improve the registration accuracy between two 3D point clouds, as shown in Figure 4. Thus, infinite iterations can be adopted during the testing. In practice, four iterations are used for data similar to the training data (Section 4.3), and ten iterations are used for unseen datasets that are not trained. (Section 4.6).

The objective of IBTreeNet is to extract effective features of the rotated and translated 3D point cloud from the BTreeNet for the next alignment iteration. Once trained, IBTreeNet can be used repeatedly and iteratively to improve the registration accuracy between two 3D point clouds. Our IBTreeNet exhibits remarkable generalization and robustness to unseen large scenes and shapes that are never trained (Section 4.6). This generalization and robustness performance can be attributed to our proposed forward propagation that avoids the interference between the feature extraction of rotation and translation.

3.3. Loss Function

The loss function for 3D point cloud registration measures the difference between the target 3D point cloud P_T and the transformed 3D point cloud P_{est}.

The loss is defined to be invariant to any permutation of 3D point clouds in both P_T and P_{est}. We use Chamfer distance (CD) [23] in the loss function.

$$CD(P_T,P_{est})=\frac{1}{P_T}\sum_{x\in P_T}\underset{y\in P_{est}}{min}\|x-y\|_2 + \frac{1}{P_{est}}\sum_{y\in P_{est}}\underset{x\in P_T}{min}\|y-x\|_2 \tag{6}$$

Chamfer distance calculates the average nearest point distance between P_T and P_{est} by finding the closest neighbour with $O(nlogn)$ complexity. In addition, P_T and P_{est} can be the different sizes of 3D point clouds.

We adopt the Earth Mover's distance (EMD) [24] as the second term in the loss function.

$$EMD(P_T,P_{est})=\underset{\phi:P_{est}\to P_T}{min}\sum_{y\in P_{est}}\|y-\phi(y)\|_2 \tag{7}$$

where $\phi : P_{est} \to P_T$ is bijection. The EMD finds a bijection ϕ and minimizes the distance between corresponding points based on ϕ with $O(n^2)$ complexity.

The final loss function, as defined in Equation 8, consists of two terms, CD and EMD. Note that both CD and EMD only require the 3D point clouds as input for unsupervised learning.

$$Loss(P_T,P_{est}) = CD(P_T,P_{est}) + EMD(P_T,P_{est}) \tag{8}$$

3.4. Implementation Details

We train BTreeNet and IBTreeNet for 300 epochs with a batch size of 32, a learning rate of 0.005, and an Adagrad optimizer. The filter sizes for PointNet-based encoder are $[64, 64, 64, 128, 256, 512]$. The features extracted from P_T and P_S are concatenated and given as an input to a fully connected layer to generate the root of the binary tree with the size of $[1 \times 1024]$. The filter sizes for the left sub-tree in each level are $[512, 256, 128, 64, 4]$. The filter sizes for the right sub-tree in each level are $[512, 256, 128, 64, 3]$. BTreeNet and IBTreeNet are trained with the input size of $N \times 3$, where N can be any number and we set $N = 1024$ during the training. Once trained, the size of the input 3D point cloud is not constrained to 1024. For example, 1024, 20,480 and 121,210 points have been used when evaluated on the testing dataset (Sections 4.3 and 4.6).

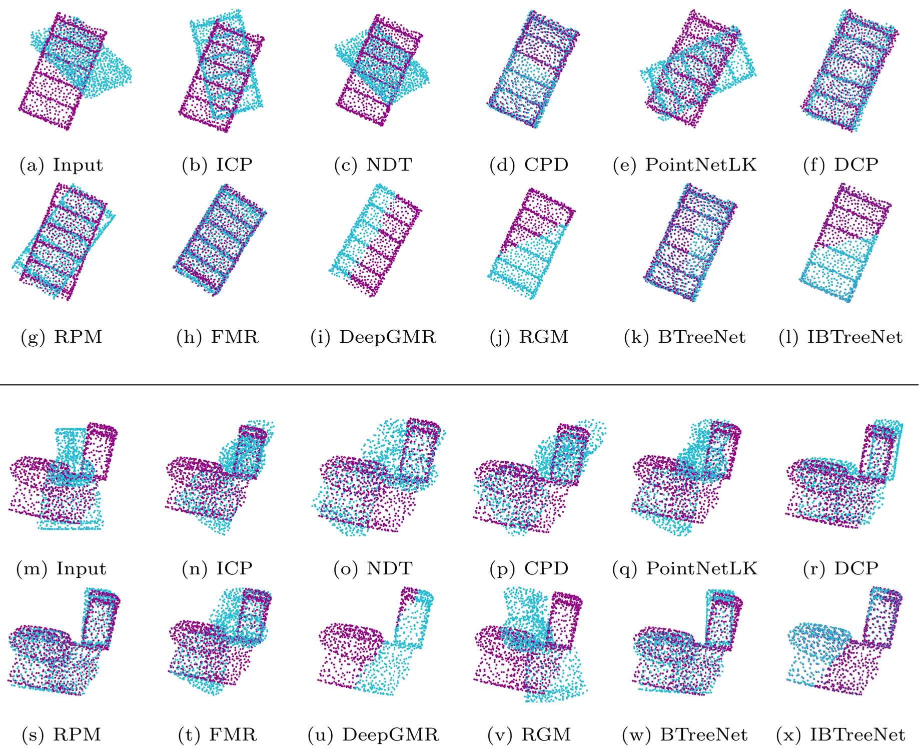


Figure 5: Registration results on two clean 3D point clouds. Each 3D point cloud contains 1,024 points.

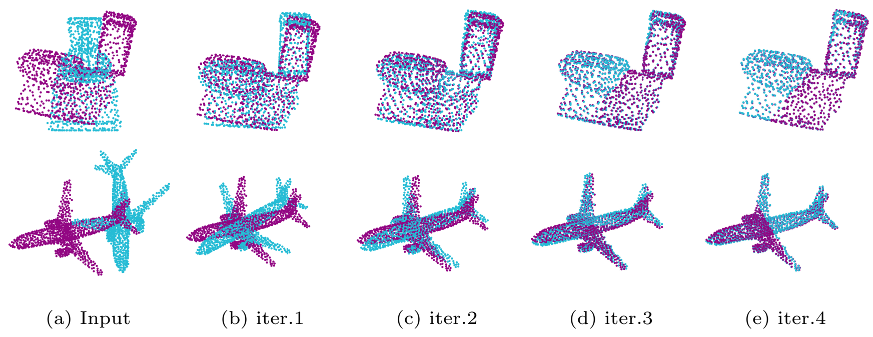


Figure 6: IBTeeNet illustration. (a) Input 3D point clouds. (b) - (e) Iteration 1 to 4.

4. Evaluation of Registration Performance

An Nvidia Geforce 2080Ti GPU with 12G memory is used for network training. We train BTreeNet, IBTreeNet and the state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] on ModelNet40 training dataset. Based on these trained models, we conduct comparison experiments to compare our BTreeNet and IBTreeNet with the traditional registration methods [7, 8, 9] and state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15]. First, we conduct comparison experiments on ModelNet40 clean testing datasets (Section 4.3). Second, we evaluate the generalization of these methods on partial and noisy point clouds without training them in such scenarios (Sections 4.4 and 4.5). Third, we evaluate the generalization and robustness ability of learning-based methods and ours on large and dense unseen KITTI [20], Whu-TLS [43, 44, 45], 3DMatch [21] and Stanford 3D datasets [22] that are not trained (Section 4.6). Fourth, we conduct comparison experiments on 3D point clouds with large initial rotations (Section 4.7). Finally, we conduct the ablation studies to evaluate the effectiveness of each component in our BTreeNet (Section 4.8).

4.1. Datasets

ModelNet40 Clean Training Data. For a fair comparison, we use the ModelNet40 [46] dataset as the training dataset following PointNetLK [10], DCP [11], RPM [12], FMR [13], DeepGMR [14] and RGM [15]. ModelNet40 [46] is composed of 12,311 meshed CAD models from 40 object categories, which contains official train/test splits (9,843 for train and 2,468 for test) for each category. All networks in our experiments are trained on the official train split in ModelNet40 [46]. Specifically, we sample 2,048 points as the source point cloud P_S from each object model in the train split and normalize P_S into a unit sphere. The source point clouds P_S are randomly rotated and translated to obtain the target point clouds P_T. For the rotation and translation applied, we randomly sample three Euler angles in the range of $[0, 45]$ degrees and translations in the range of $[-0.5, 0.5]$ on each axis during the training. Finally, we shuffle the point order in P_S and P_T, respectively. During the training, all point cloud pairs are clean data without noise and missing data. Thus, exact point-to-point correspondences exist between P_S and P_T in training data.

Testing Data. To evaluate the robustness and generalization ability of a network, the training and the testing sets should be in different scenarios. Thus, we train all networks only on ModelNet40 Clean Training Data and test them on clean, partial, noisy and unseen point clouds, respectively. The testing data contains seven different groups of 3D point cloud datasets, including *ModelNet40 Clean* (Section 4.3), *ModelNet40 Partial* (Section 4.4), *ModelNet40 Noise* (Section 4.5), *Unseen KITTI* (Section 4.6.1), *Unseen Whu-TLS* (Section 4.6.2), *Unseen 3DMacth* (Section 4.6.3) and *Unseen Shapes* (Section 4.6.4).

Note that *Unseen KITTI* and *Unseen Whu-TLS* are large-scale, outdoor and real-world scenes captured from different 3D scanner systems with realistic sensor noise and variations in the point density, field of view, clutter and occlusion. *Unseen 3DMacth* is an RGBD-reconstruction indoor dataset, which also contains the different point density and distribution compared with *Unseen KITTI* and *Unseen Whu-TLS*. *Unseen Shapes* dataset consists of the Stanford 3D scanning models from different 3D scanners, and the shapes are totally different from the ModelNet40 Clean Training Data. These four unseen datasets are not trained and have been used to evaluate the generalization ability of each network.

4.2. Evaluation Metrics

We evaluate the registration by computing six evaluation metrics. First, we adopt the Chamfer Distance, as defined in Equation 6, to measure how close the two point clouds are brought to each other. Second, we further evaluate the transformation matrix through Frobenius norm based on the Special Euclidean Group for the overall registration errors, as defined in Equation 9.

$$F(\mathbf{T}) = \|\mathbf{T}_{gt} - \mathbf{T}_{est}\|_F \, ; \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{9}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$.

Table 1: Evaluations on the *ModelNet40 Clean* testing dataset with $[0, 45]$ degrees of initial rotations. Bold denotes top four performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.053823	0.408176	15.926107	0.095590	8.106933	0.048669
NDT	0.060876	0.520965	21.684867	0.043825	10.500771	0.022000
CPD	0.015127	0.090626	3.749195	*0.002207*	1.784223	*0.001123*
PNLK	0.016350	0.235991	10.447890	0.013864	5.581651	0.006875
DCP	0.016620	0.071662	2.874782	0.008266	1.469114	0.004150
RPM	**0.000709**	**0.007587**	**0.309407**	**0.002072**	**0.160725**	**0.000947**
FMR	0.014335	0.075151	3.171169	0.002772	1.561003	0.001385
DeepGMR	**0.000104**	**0.003671**	**0.240103**	**0.000024**	**0.115103**	**0.000013**
RGM	< **0.000001**	**0.001464**	**0.057414**	**0.000785**	**0.023415**	**0.000399**
BTreeNet	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988
IBTreeNet	**0.010008**	**0.059417**	**2.346349**	**0.007623**	**1.119935**	**0.003796**

Finally, we ues the mean isotropic rotation and translation errors (MIE) proposed in RPM [12] and the mean absolute errors (MAE) of rotation and translation proposed in DCP [11], as defined in Equation 10.

$$\begin{aligned} &MIE(\mathbf{R}) = \angle(\mathbf{R}_{gt}^{-1}\mathbf{R}_{est}); \quad MIE(\mathbf{t}) = \|\mathbf{t}_{gt} - \mathbf{t}_{est}\|_2 \\ &MAE(\mathbf{R}) = \frac{1}{m}\sum_{i=1}^{m}\left|\angle(\mathbf{R}_{gt} - \mathbf{R}_{est})\right|; \quad MAE(\mathbf{t}) = \frac{1}{m}\sum_{i=1}^{m}|\mathbf{t}_{gt} - \mathbf{t}_{est}| \end{aligned} \tag{10}$$

where $\mathbf{R}_{gt}$ and $\mathbf{t}_{gt}$ denote the ground-truth rotation and translation, and $\mathbf{R}_{est}$ and $\mathbf{t}_{est}$ represent the estimated rotation and translation. $\angle(\cdot)$ denotes the angle of rotation matrix in degrees.

4.3. ModelNet40 Clean Test

In this experiment, we evaluate the registration performance on *ModelNet40 Clean* testing dataset that contains 2,468 clean point cloud pairs. *ModelNet40 Clean* testing dataset is generated from the official test split in ModelNet40 [46]

Table 2: Quantitative comparison of our approach against previous works on parameters, number of iterations and computational time. OOM means a 32GB memory with an Intel i7-9700 3.00GHz CPU or a 12GB NVIDIA RTX 2080Ti GPU out of memory with a forward pass on a single instance.

Methods	Supervision	Network Parameters	Iterations	Computational Time on CPU(s) (1,024 points)	(20,480 points)
ICP	-	-	20	1.449999	1.436067
NDT	-	-	30	0.229999	1.539240
CPD	-	-	20	0.699999	259.711729
PNLK	Yes	151,686	10	0.797498	2.894987
DCP	Yes	5,568,905	1	6.414267	22.199531
RPM	Yes	905,154	5	4.881233	264.245900
FMR	No	3,440,006	10	0.168407	9.192588
DeepGMR	Yes	1,527,440	1	1.531708	2.281754
RGM	Yes	25,000,836	2	2.685069	OOM
BTreeNet	No	1,483,463	1	0.141319	0.806401
IBTreeNet	No	1,483,463	4	1.114456	4.225035

that includes 2,468 clean point clouds. We randomly sample 1,024 points and normalize points into a unit sphere. To obtain target point clouds during the testing, the random rotations are in the range of $[0, 45]$ degrees and translations in the range of $[-0.5, 0.5]$ on each axis are applied to the official test split without noise and missing data with exact point-to-point correspondences. All learning-based network models (PointNetLK [10] DCP [11], RPM [12], FMR [13], Deep-GMR [14], RGM [15] and ours) are trained on the same ModelNet40 Clean Training Data.

Figure 6 illustrates the iteration process of our IBTreeNet, which shows that our IBTreeNet can iteratively improve the registration accuracy between two 3D point clouds. Note that the first iteration (b) in Figure 6 is the result of BTreeNet, and other iterations (c), (d) and (e) are the results of IBTreeNet.

The average registration errors of BTreeNet and IBTreeNet against three traditional methods [7, 8, 9] and six state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] are shown in Table 1 with qualitative results in Figure 5. Lower average errors indicate lower registration errors. Although our networks do not achieve the best registration performance in *ModelNet40 Clean* testing set, BTreeNet and IBTreeNet still outperform three traditional methods [7, 8, 9] and three learning-based network models [10, 11, 13]. DeepGMR [14], RGM [15] and RPM [12] achieve top measures on clean data. However, they perform poorly on partial and noisy 3D point clouds without training them in such scenarios, and detailed information is described in Sections 4.4 and 4.5.

Furthermore, the number of parameters, supervision method, iterations and computational time are also essential for each network. Since the traditional algorithms [7, 8, 9] do not provide the GPU acceleration, we use a 32GB memory card with an Intel i7-9700 3.00GHz CPU for a fair comparison of computational time. The computational time is collected on two testing datasets with different number of points, including *ModelNet40 Clean* (1,024 points) and *Unseen KITTI* (20,480 points) (Section 4.6.1) testing datasets. As shown in Table 2, PointNetLK [10] DCP [11], RPM [12], DeepGMR [14] and RGM [15] are fully-supervised, which needs ground-truth rotation and translation as the supervision signal while FMR [13] and our methods are unsupervised, thus, can largely reduce the training cost.

RMG [15] contains the most significant number of network parameters and has the out of memory issue on processing dense 3D point clouds, which requires a high computational cost and relies on powerful commodity GPU or CPU processors. DCP-v2 [11] contains the second largest number of network parameters, which requires a high computational cost and has the out of memory issue on dense 3D point clouds. Thus, we use DCP-v1 [11] to evaluate dense point clouds with 20,480 points and DCP-v2 [11] on sparse point clouds with 1,024 points. Traditional methods [7, 8, 9] take maximum 20 or 30 iterations for 3D point cloud registration whereas learning-based methods PointNetLK [10] and FMR [13] require maximum 10 iterations. RPM [12] and RGM [15] need 5 and 2 iterations during the testing. DCP [11], DeepGMR [14] and our BTreeNet

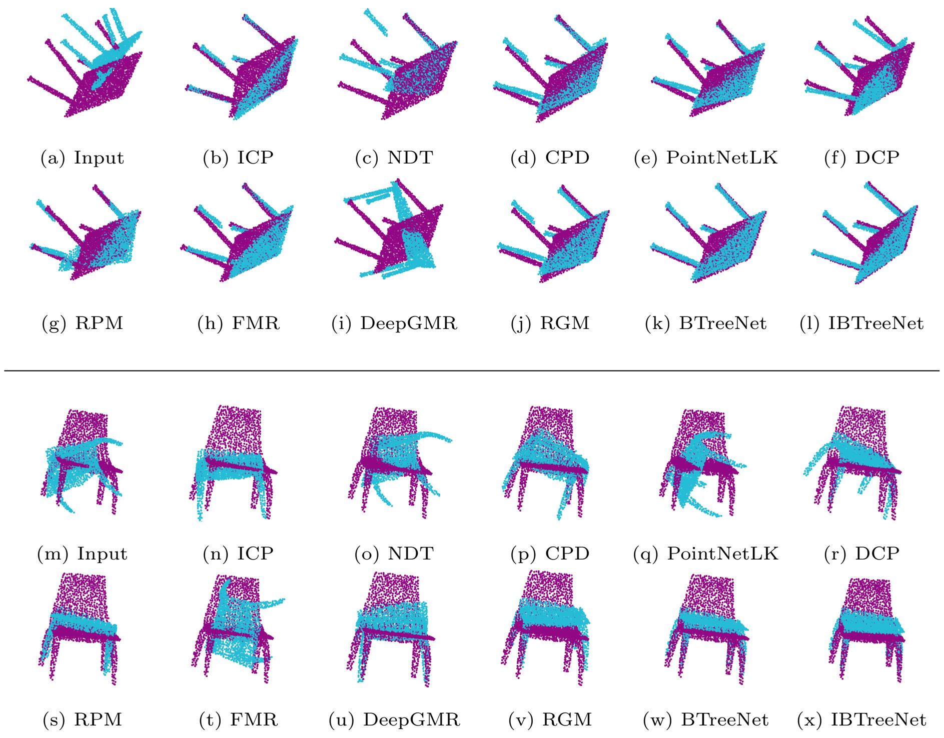


Figure 7: Registration results on partial 3D point clouds. Each 3D point cloud contains 1,024 points.

only require 1 iteration (see Table 2).

For sparse 3D point clouds with 1,024 points, DCP [11], RPM [12] and RGM [15] require a longer computational time on a single instance, respectively. Our BTreeNet only takes around 0.1 seconds and ranks first in computational time. For dense 3D point clouds with 20,480 points, CPD [9] and RPM [12] require the longest computational time on dense point clouds and takes approximately 4.5 minutes for a single instance. Our BTreeNet requires less than 1 second on dense 3D point clouds with 20,480 points, ranking first in computational time.

4.4. Partial Visibility

To generate partial point clouds that do not fully overlap in extent, we follow the protocol in RPM [12] and RGM [15] and generate the *ModelNet40 Partial*

Table 3: Evaluations on ModelNet40 partial 3D point clouds with 30% missing data. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top performing measures.

Methods	CD ↓	F Norm ↓ ($\mathbf{T}$)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.068601	0.559043	21.683043	0.154110	11.064304	0.077339
NDT	0.078709	0.625850	25.139421	0.135835	12.272863	0.065737
CPD	0.064756	0.435410	15.968742	0.162647	8.067676	0.081858
PNLK	0.098572	0.919244	40.113300	0.181374	21.456369	0.091555
DCP	0.102747	1.043511	46.917641	0.001283	25.693749	0.000646
RPM	0.126092	1.284177	60.215692	0.207321	33.902779	0.103731
FMR	0.058666	0.413444	15.552787	0.139811	8.057808	0.069266
DeepGMR	0.149778	2.022872	104.408779	0.202822	60.829105	0.101588
RGM	0.061925	0.380452	12.045048	0.215026	6.335146	0.215026
BTreeNet	**0.027250**	**0.164416**	**6.679313**	**0.000063**	**3.392343**	**0.000029**
IBTreeNet	**0.025889**	**0.150333**	**6.105696**	**0.000049**	**3.037774**	**0.000024**

testing dataset from the *ModelNet40 Clean*, which is more realistic and closer to real-world applications. For each source point cloud in the *ModelNet40 Clean*, a random clipping plane passing through the origin is created and shifted to retain 70% of the points. In *ModelNet40 Partial*, the initial rotations and translations are the same as the *ModelNet40 Clean*, but exact point-to-point correspondences have been broken and do not exist.

Most importantly, to evaluate the robustness and generalization ability of each network, all networks are only trained on ModelNet40 Clean Training Data (Section 4.1) without missing points and directly tested on *ModelNet40 Partial*. For a fair comparison, we follow DCP [11], RPM [12] and RGM [15] and report the average registration errors on the group with 30% missing rate in Table 3 with qualitative results in Figure 7. Three traditional algorithms [7, 8, 9] outperform the four learning-based methods including PointNetLK [10], DCP [11],

Table 4: Evaluations on Gaussian noise with the standard deviation equals to 0.05. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes the top performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.049258	0.413435	16.154294	0.096340	8.241147	0.049071
NDT	0.058998	0.518387	21.541951	0.044982	10.476870	0.022424
CPD	0.016864	0.140918	5.753399	0.006682	2.750101	0.003356
PNLK	0.032994	0.332014	14.352293	0.037295	7.654664	0.018523
DCP	0.050837	0.463393	20.397403	0.001136	10.807378	0.000572
RPM	0.083747	1.079716	51.637915	0.003194	29.808305	0.001602
FMR	0.020769	0.151866	6.166866	0.022762	3.141542	0.011399
DeepGMR	0.067272	0.803346	37.094805	0.006741	22.525591	0.003408
RGM	0.050613	0.586429	29.838333	0.000880	15.885876	0.000443
BTreeNet	**0.013962**	**0.081754**	**3.317700**	**0.000070**	**1.700144**	**0.000034**
IBTreeNet	**0.012318**	**0.069757**	**2.829888**	**0.000060**	**1.389002**	**0.000031**

RPM [12] and DeepGMR [14]. FMR [13] outperforms the traditional algorithms [7, 8, 9] and other learning-based methods [10, 11, 12, 14, 15]. Our BTreeNet and IBTreeNet are significantly more accurate than both traditional and learning-based methods, which indicates remarkable generalization on partial 3D point clouds.

We analyse two possible reasons for the failure registration of the state-of-the-art learning-based methods [10, 11, 14, 12, 15] on partial 3D point clouds. These methods perform poorly on partial 3D pint clouds without training them in this scenario, indicating the poor generalization of each network on partial visibility with broken point-to-point correspondences. A trained network model should extract generalized features for different datasets and scenarios. Otherwise, it can only perform well on a specific dataset or a scenario and needs to be retrained under different conditions. Second, these networks apply the ground-

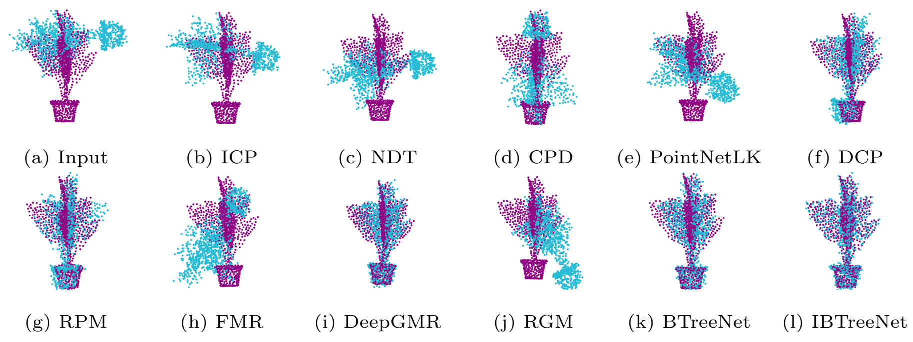


Figure 8: Registration results on 3D point clouds with Gaussian noise. Each 3D point cloud contains 1,024 points.

truth transformation labels (rotation and translation) as supervision during the training, which forces the network to output a certain transformation based on a certain pattern of input 3D point cloud pairs. Since the broken point-to-point correspondences have changed the pattern of input 3D point cloud pairs during the testing, the trained network models cannot recognize the changed pattern and generate wrong transformations.

FMR [13] and our methods are all trained in an unsupervised manner and find the global registration on input 3D point cloud pairs without ground-truth transformation, which avoids the limitation of the supervised manner and outperforms other learning-based methods on partial 3D point cloud registration.

4.5. Gaussian Noise

To evaluate the robustness and generalization ability of each method to noise, we generate the *ModelNet40 Noise* from the *ModelNet40 Clean*. Specifically, we sample noise from Gaussian distribution for the source 3D point cloud in *ModelNet40 Clean* with 0 mean and a standard deviation of 0.05. In the *ModelNet40 Noise*, the Gaussian noise is added to all source 3D point clouds in the *ModelNet40 Clean*, and these noises destroy the original point-to-point correspondences. The initial rotations and translations are the same as the *ModelNet40 Clean*.

Table 5: Evaluations on the unseen KITTI dataset that is not trained. All networks are trained on the Modelnet40 Clean Training Data without missing and noisy points. Bold denotes top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.030695	0.505911	16.626091	0.260118	8.847980	0.130960
NDT	0.030422	0.877822	36.667289	0.102342	16.307743	0.051711
CPD	**0.001883**	**0.086971**	**3.557581**	**0.009284**	**1.241874**	**0.004788**
PNLK	**0.002546**	**0.085499**	**3.672698**	**0.006742**	**2.088179**	**0.003364**
DCP	0.003720	0.181936	3.151685	0.143950	1.624810	0.073790
RPM	0.010139	0.360977	17.587084	0.029334	8.679858	0.014839
FMR	0.008252	0.245331	10.790774	0.031303	7.057602	0.015883
DeepGMR	**0.000039**	**0.000580**	**0.027025**	**0.000036**	**0.011429**	**0.000018**
RGM	0.005711	0.155436	6.258668	0.015307	3.305840	0.007752
BTreeNet	0.006533	0.256744	7.235349	0.144797	3.510368	0.074197
IBTreeNet	0.005452	0.235064	5.982500	0.144624	2.783603	0.074102

Most importantly, all networks are only trained on ModelNet40 Clean Training Data (Section 4.1) without noise and directly tested on *ModelNet40 Noise*. The average registration errors of each method to noise with the standard deviation equal to 0.05 are reported in Table 4 with qualitative results shown in Figure 8. Our methods are much more accurate than the state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] and traditional methods [7, 8, 9].

Similarly to the results on partial 3D point clouds, DeepGMR [14], RGM [15] and RPM [12] do not provide accurate registration on noisy point clouds if these networks are not trained in this scenario, which indicates the poor robustness and generalization ability of these networks with noise. Traditional method CPD [9] outperforms the state-of-the-art learning-based methods [10, 11, 12, 13, 14, 15] on noisy point clouds, which indicates that CPD [9] can tolerate large noise on point clouds.

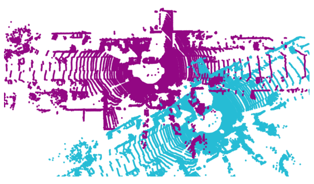
Unsupervised learning-based method FMR [13] and ours still outperform other supervised learning-based methods [10, 11, 12, 14, 15] on noisy 3D point clouds, which demonstrates the effectiveness of the unsupervised manner on the network generalization ability. The possible reasons for the failure registration of the state-of-the-art learning-based methods on noisy 3D point clouds are the same as that on partial 3D point clouds (see Section 4.4).

4.6. Generalization across Unseen Datasets

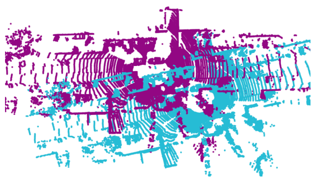
We extensively conduct several comparison experiments to evaluate the generalization and robustness capability of our method. All learned models including PointNetLK [10] DCP [11], RPM [12], FMR [13], DeepGMR [14], RGM [15] and ours are trained on the same ModelNet40 Clean Training Data (Section 4.1), and directly tested on the completely unseen datasets that are never trained. Note that the ModelNet40 Clean Training Data consists of sparse 3D point clouds (e.g. table, bookshelf and guitar), and the unseen datasets are the large and dense outdoor and indoor scenes captured from various sensors and totally different from the ModelNet40 Clean Training Data. This large domain gap between these datasets poses a significant challenge to the generalization of all learning-based methods.

4.6.1. Generalization on Unseen KITTI Scenes

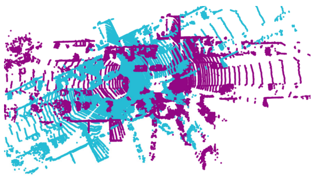
To evaluate the generalization of each network on unseen datasets that are not trained, we generate *Unseen KITTI* testing dataset, including 1,146 KITTI odometry [20] LiDAR 3D point cloud pairs with realistic sensor noise for the evaluation. KITTI odometry [20] is an outdoor dense point cloud dataset acquired by Velodyne-64 3D LiDAR scanners. We select every six frames in the sequence of outdoor scans to ensure that the selected 1,460 point clouds can fully represent the scenario and difficulty in this dataset. We sample 20,480 points from each selected dense 3D point cloud and normalize it into a unit sphere. Note that, the number of the down-sampled KITTI 3D point cloud is still 20 times larger than that in the *ModelNet40 Clean* testing dataset. The



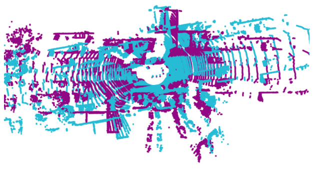
(a) Input unseen outdoor scenes



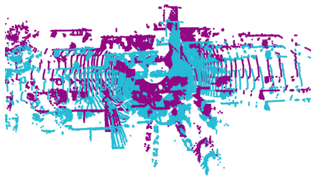
(b) ICP



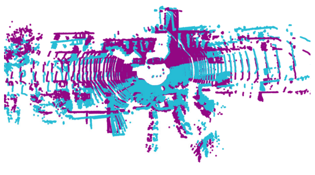
(c) NDT



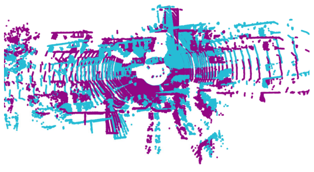
(d) CPD



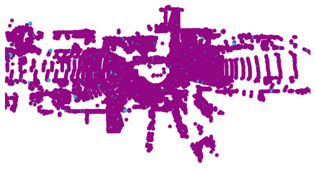
(e) PointNetLK



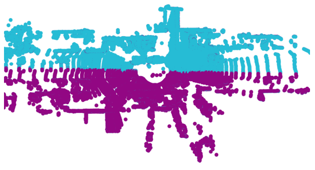
(f) DCP



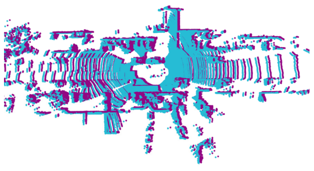
(g) RPM



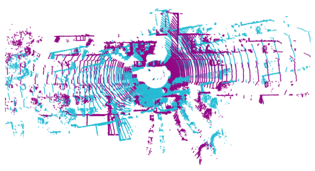
(h) FMR



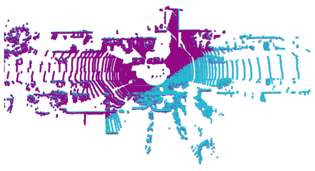
(i) DeepGMR



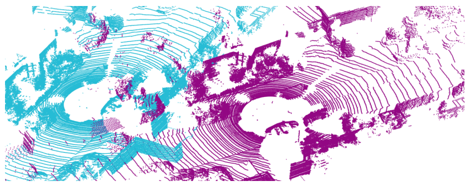
(j) RGM



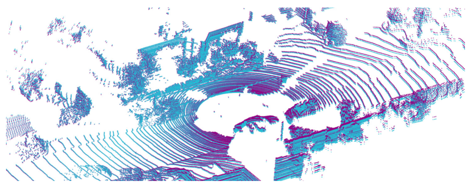
(k) BTreeNet



(l) IBTreeNet



(m) Input details



(n) Ours (iter.8)

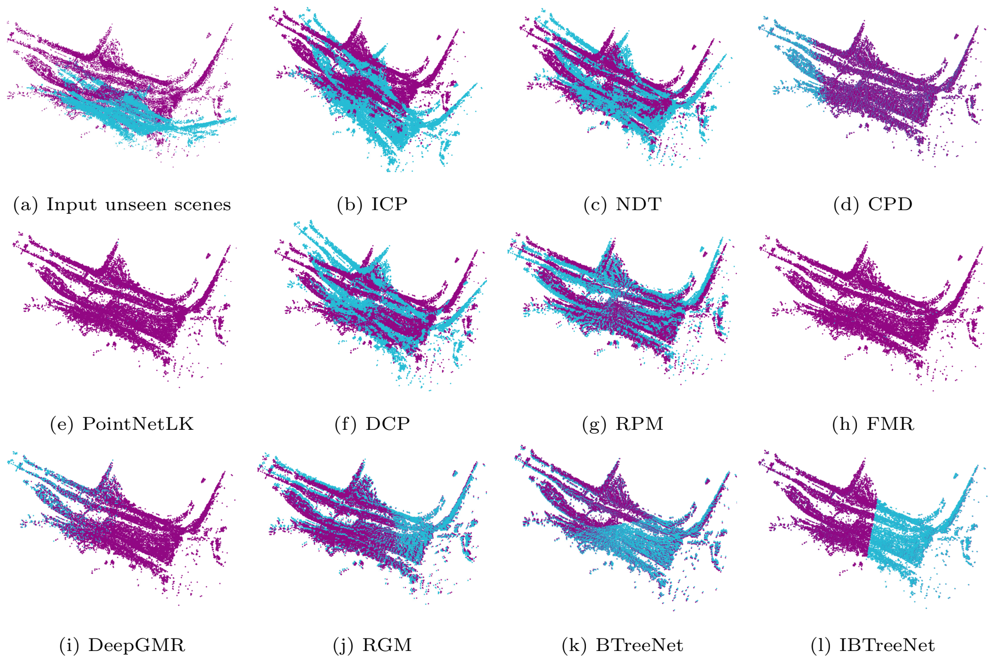
Figure 9: Qualitative registration results on unseen large and dense KITTI LiDAR 3D point clouds. Each 3D point cloud contains 121,210 points for display. (a) Input 3D point clouds. (b) - (d) Registration results of non-learning based methods ICP, NDT and CPD. (e) - (h) Registration results of learning-based methods PointNetLK, DCP, RPM and RGM. (i) Registration results of our IBTreeNet. (j) and (k) Details of input and registration results of IBTreeNet.

Table 6: Evaluations on the *Unseen Whu-TLS* datasets that are not trained. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes top three performing measures.

Methods	CD ↓	F Norm ↓	MIE ↓	MIE ↓	MAE ↓	MAE ↓
		(**T**)	(Rot.)	(Trans.)	(Rot.)	(Trans.)
ICP	0.030002	0.870613	36.191176	0.337518	19.398201	0.170002
NDT	0.048445	0.702454	27.452705	0.165263	13.076433	0.082384
CPD	**0.003510**	**0.057525**	**2.313392**	**0.007674**	**0.888502**	**0.004049**
PNLK	0.009444	0.456522	13.951705	0.278294	7.738035	0.129433
DCP	0.009758	0.351936	9.775516	0.344886	5.153150	0.162643
RPM	0.024656	0.466777	21.402695	0.069478	11.701789	0.034369
FMR	0.022515	0.390340	18.571252	0.062369	11.474063	0.030415
DeepGMR	**0.000128**	**0.022431**	**0.679253**	**0.014806**	**0.399354**	**0.007705**
RGM	0.008434	0.175065	6.893064	0.026624	3.693086	0.013407
BTreeNet	0.012251	0.378394	9.169036	0.277799	4.697491	0.139201
IBTreeNet	**0.008388**	**0.142164**	**6.251942**	**0.277579**	**2.882725**	**0.139073**

settings of initial rotations and translations are the same as the *ModelNet40 Clean* testing dataset. Since the target point clouds are rotated and translated from the source point clouds, thus exact point-to-point correspondences exist in *Unseen KITTI*.

In this experiment, all networks are trained on ModelNet40 Clean Training Data (Section 4.1) and directly tested on *Unseen KITTI*. RGM [15] requires significant memory and has the out of memory issue with a forward pass on a single instance tested on a 32GB memory with an Intel i7-9700 3.00GHz CPU and a 12GB NVIDIA RTX 2080Ti GPU. By this hardware setting, RGM [15] can only process approximately 3,078 points. Thus we down-sample all point cloud pairs in *Unseen KITTI* to 3,078 points only for RGM [15]. Once the transformation matrix has been obtained from RGM [15], we apply the estimated transformation matrix to the original source point cloud with 20,480


(a) Input unseen scenes (b) ICP (c) NDT (d) CPD

(e) PointNetLK (f) DCP (g) RPM (h) FMR

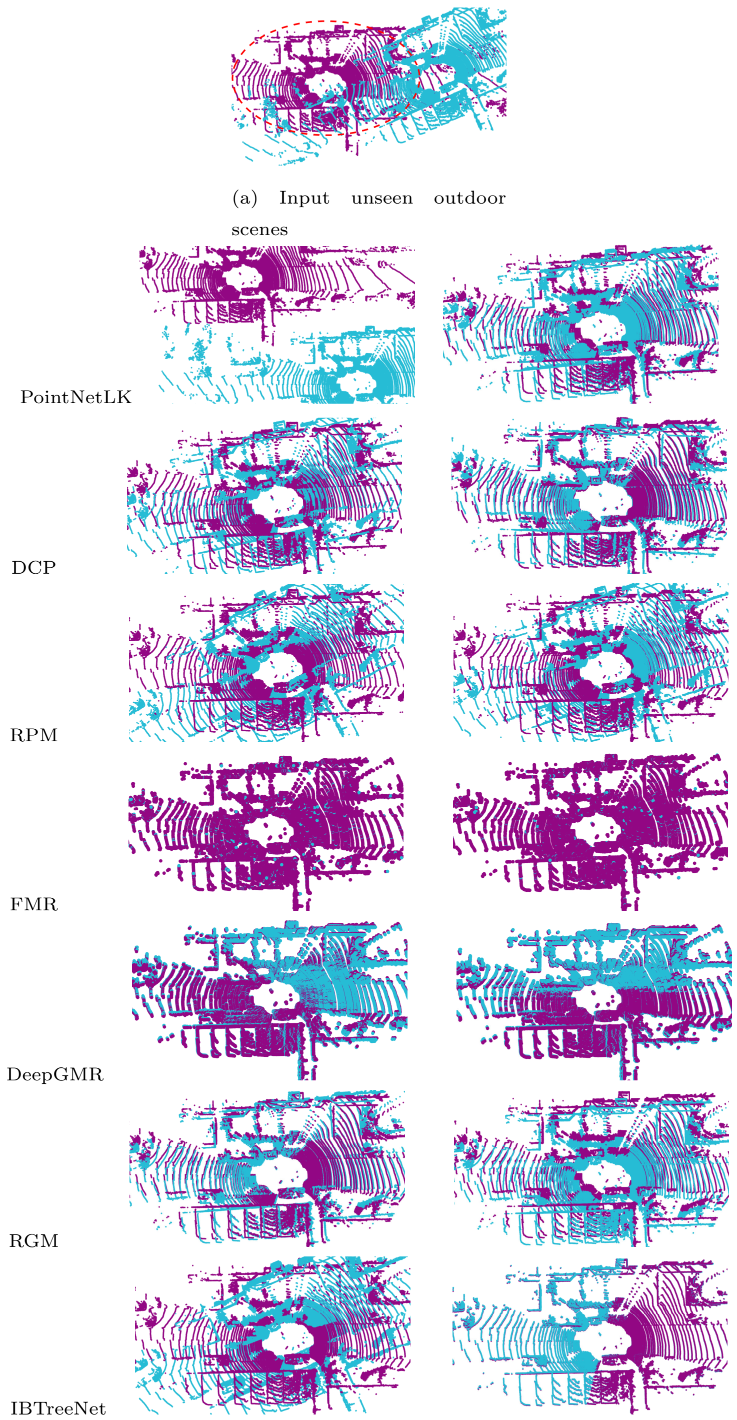
(i) DeepGMR (j) RGM (k) BTreeNet (l) IBTreeNet

Figure 10: Qualitative registration results on unseen large-scale 3D point clouds. Each 3D point cloud contains 20,480 points.

points.

The performance of each method on *Unseen KITTI* is reported in table 5. DeepGMR [14] ranks first in all performing measures and achieves the best registration performance on *Unseen KITTI*. Traditional method CPD [9] outperforms all learning-based methods except DeepGMR [14], which shows the strong generalization of the CPD algorithm. Our IBTreeNet outperforms PRM [12], FMR [13], RGM [15], ICP [7] and NDT [8].

Figure 9 shows the qualitative registration results of each methods on KITTI datasets. Figure 11 shows the iteration illustration of each learning-based methods. It is worth noting that, the iteration methods proposed in DCP [11], RPM [12] and RGM [15] are limited in improving the registration accuracy, whereas our IBTreeNet iteratively achieves the precise registration and shows great generalization and robustness to unseen KITTI LiDAR points.



(a) Input unseen outdoor scenes

(b) iter. 1

(c) iter. 8

Figure 11: Iteration illustration on unseen KITTI LiDAR 3D point clouds. Each 3D point cloud contains 111,989 points for display.

4.6.2. Generalization on Unseen Whu-TLS Scenes

To evaluate the generalization of each network on different unseen datasets that are not trained, we generate *Unseen Whu-TLS* testing dataset, including 1,000 Whu-TLS [43, 44, 45] 3D point cloud pairs with realistic sensor noise for the evaluation. Whu-TLS [43, 44, 45] dataset, captured by terrestrial laser scanners (TLS), is a large-scale 3D point cloud dataset, which includes 10 different 3D point cloud scenes (i.e., subway station, residence, riverbank, etc.) with variations in the point density, clutter and occlusion[1]. The Whu-TLS dataset is challenging because multiple laser scanner systems (i.e., VZ-400, ScanStationC5, Leica HDS6100, etc.) with differences in terms of the measurement range, accuracy and field of view are used to capture the 3D point clouds. For each scene in 10 different scenes, we generate 100 point cloud pairs with the same settings as the *Unseen KITTI* testing dataset.

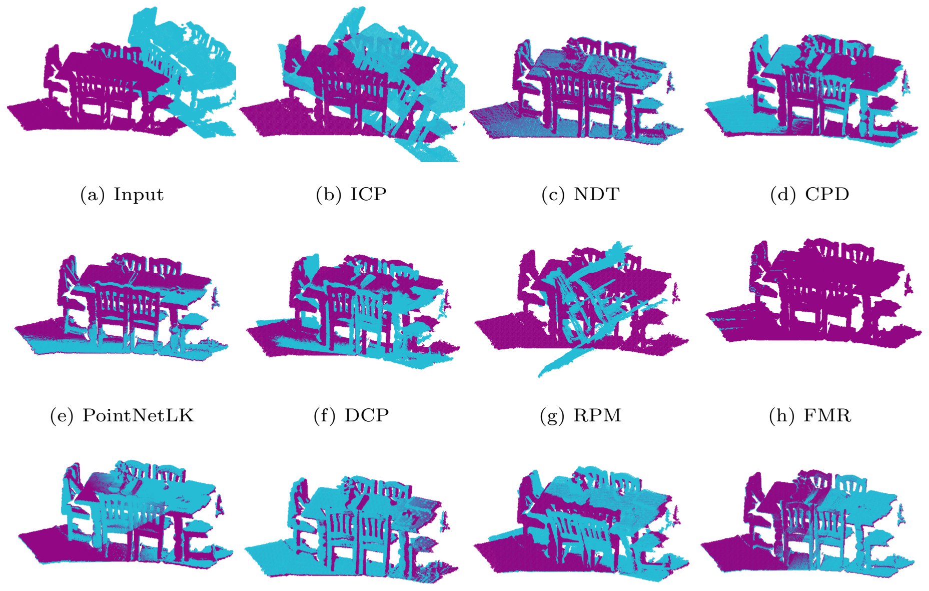
In this experiment, all networks are still trained on ModelNet40 Clean Training Data (Section 4.1) and directly tested on *Unseen Whu-TLS*. As illustrated in Section 4.6.1, we use the same settings for RGM [15].

The performance of each method on *Unseen Whu-TLS* is reported in table 6 with qualitative registration results of each method in Figure 10. Learning-based method DeepRGM [14] ranks first in all performing measures, and the traditional method CPD [9] outperforms all learning-based methods except DeepGMR [14], which shows the strong generalization of the DeepRGM [14] and CPD [9] on *Unseen Whu-TLS*. Our IBTreeNet outperforms RGM [15], PointNetLK [10], DCP [11], FMR [13], RPM [12], ICP [7] and NDT [8] based on the overall registration errors.

4.6.3. Generalization on Unseen Indoor Scenes

We then conduct the comparison experiments on unseen indoor scenes and generate *Unseen 3DMatch* testing dataset, including 433 3DMatch [21] 3D point cloud pairs with the same settings as the *Unseen KITTI* testing dataset.

[1]Whu-TLS generates 11 different scenes and 10 scenes are publicly available for researches.



(a) Input (b) ICP (c) NDT (d) CPD

(e) PointNetLK (f) DCP (g) RPM (h) FMR

(i) DeepGMR (j) RGM (k) BTreeNet (l) IBTreeNet

Figure 12: Registration results on unseen 3DMatch datasets.

3DMatch [21] is an RGB-D reconstruction dataset, which contains 433 reconstructed dense 3D point clouds from eight sets of 2D scene images created from the official testing split of the RGB-D reconstruction datasets [21]. The data distribution of the reconstructed 3D point cloud is different from the data captured by the LiDAR and laser scanners. In this experiment, all networks are still trained on ModelNet40 Clean Training Data (Section 4.1) and directly tested on *Unseen 3DMatch*. As illustrated in Section 4.6.1, we still use the same settings for RGM [15].

The average registration errors of BTreeNet and IBTreeNet against ICP [7], NDT [8], CPD [9], PointNetLK [10], DCP [11], RPM [12], FMR [13], DeepGMR [14] and RGM [15] on *Unseen 3DMatch* are shown in Table 7, with qualitative results shown in Figure 12.

Similarly to the results on *Unseen Whu-TLS*, the learning-based method DeepGMR [14] ranks first in all performing measures, and the traditional method

Table 7: Evaluations on the unseen 3DMatch datasets that are not trained. All networks are trained on modelnet40 clean data without missing and noisy points. Bold denotes top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.073672	0.480146	18.846508	0.128525	10.100460	0.064284
NDT	0.079254	0.521475	21.268932	0.109614	10.633771	0.056268
CPD	**0.018102**	**0.112831**	**4.536769**	**0.019583**	**2.186277**	**0.009842**
PNLK	0.029412	0.223548	9.839967	0.030038	5.330842	0.014840
DCP	0.057605	0.499294	14.018398	0.312191	7.619330	0.156427
RPM	0.051456	0.401897	12.399297	0.045833	6.329186	0.023072
FMR	0.031666	0.454459	10.464576	0.036622	5.778973	0.018512
DeepGMR	**0.005068**	**0.017258**	**0.678505**	**0.002990**	**0.360289**	**0.001487**
RGM	0.042781	0.439900	10.011215	0.041061	5.486697	0.020612
BTreeNet	0.039944	0.398068	8.667772	0.310266	4.306098	0.155422
IBTreeNet	**0.028588**	**0.221609**	**6.914699**	**0.310182**	**3.353310**	**0.155365**

CPD [9] outperforms all learning-based methods except DeepGMR [14]. Our IBTreeNet outperforms RGM [15], PointNetLK [10], DCP [11], FMR [13], RPM [12], ICP [7] and NDT [8] based on the overall registration errors. By comparison, our IBTreeNet also achieves remarkable generalization to unseen indoor scenes.

Note that for all learning-based methods and traditional methods except CPD [9], the registration performance decreases to some extent on *Unseen 3DMatch* compared to the evaluation on *Unseen KITTI* and *Unseen Whu-TLS*. We analyse one possible reason is that the data distribution and density of the reconstructed 3D point clouds from 2D images are different from the LiDAR and laser scanners.

Table 8: Evaluations on the *Unseen Shapes* testing datasets that are not trained. All networks are trained on Modelnet40 Clean Training Data without missing and noisy points. Bold denotes top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.064974	0.445631	17.599977	0.103102	9.721335	0.052357
NDT	0.050633	0.410507	16.808854	0.065777	8.472930	0.033677
CPD	**0.014910**	**0.078536**	**3.070854**	**0.017432**	**1.340475**	**0.008933**
PNLK	0.023981	0.232628	10.280659	0.029590	6.199633	0.014813
DCP	0.079755	0.663195	25.690459	0.196156	13.536360	0.095452
RPM	0.084928	0.653797	26.749477	0.073703	13.512754	0.035491
FMR	0.026686	0.253822	11.499332	0.045954	5.902958	0.023032
DeepGMR	**0.000121**	**0.000404**	**0.020372**	**0.000026**	**0.007894**	**0.000013**
RGM	0.049332	0.356896	14.399312	0.045966	7.821818	0.022589
BTreeNet	0.026233	0.267393	6.878716	0.196028	3.622638	0.095358
IBTreeNet	**0.019934**	**0.243259**	**5.151717**	**0.195974**	**2.528907**	**0.095340**

4.6.4. Generalization on Unseen Shapes

Additionally, we evaluate the performance of each method on unseen shapes including the Stanford Dragon [22], Bunny [22], Hand [47] and Children[2] that are not trained. We create an *Unseen Shapes* testing dataset, including 1,000 3D point cloud pairs with the same settings as the *Unseen KITTI* testing dataset. All networks are still trained on ModelNet40 Clean Training Data (Section 4.1) and directly tested on *Unseen Shapes.* As illustrated in Section 4.6.1, we still use the same settings for RGM [15].

The performance of each method on *Unseen Shapes* is reported in table 8. Similarly to the results on *Unseen Whu-TLS* and *Unseen 3DMatch*, Deep-RGM [14] ranks first in all performing measures and CPD [9] outperforms all

[2]`http://visionair.ge.imati.cnr/`

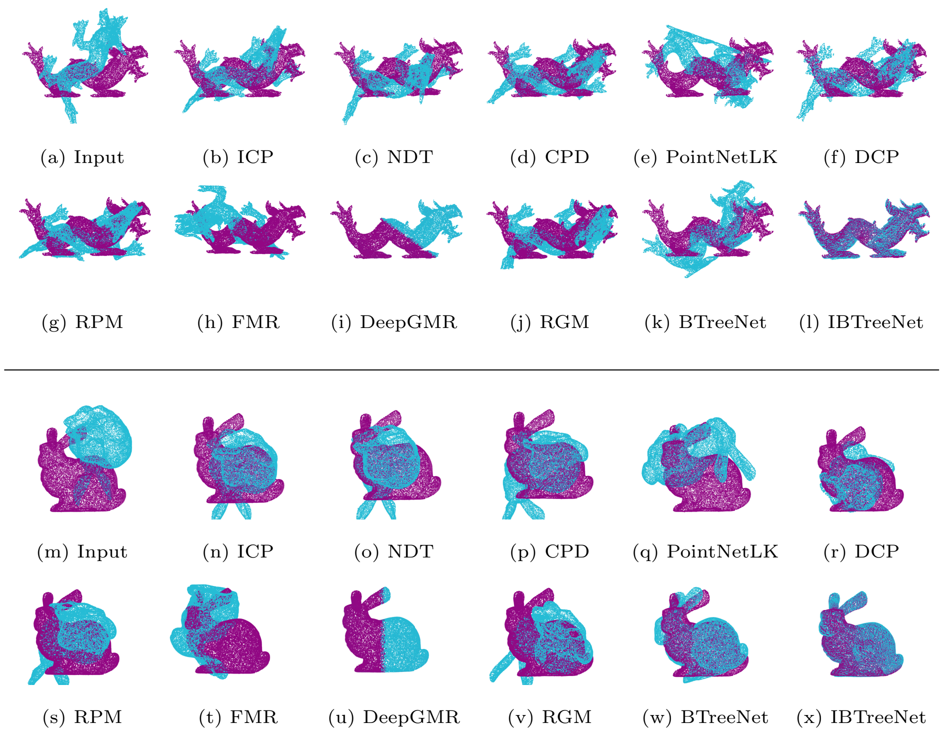


Figure 13: Registration results on unseen shapes datasets. Each 3D point cloud contains 20,480 points.

learning-based methods except DeepGMR [14] on *Unseen Shapes*. The learning-based method PointNetLK [10], FMR [13] and our BTreeNet achieve the approximately similar registration results based on the overall evaluation metrics. Our IBTreeNet outperforms all learning-based methods except DeepGMR [14]. The qualitative registration results are shown in Figure 13.

4.7. Registration with Large Rotations

Large rotation is a challenging task for 3D point cloud registration. In this experiment, we train all networks on ModelNet40 Clean Training Data (Section 4.1) with the random rotations from 0 to 180 degrees. We then test all trained models on *ModelNet40 Clean* testing dataset with the random rotations from 0 to 180 degrees. Note that, we only changed the random rotations from

Table 9: Evaluations on 3D point clouds with the initial rotation ranges from 0 to 180 degrees. Bold denotes top three performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.105312	2.463621	141.334273	0.162676	74.069710	0.079737
NDT	0.143004	2.472329	139.066895	0.113671	76.105349	0.056608
CPD	0.068746	2.386553	143.228976	0.053039	78.755450	0.026269
PNLK	0.126135	2.415207	137.081225	0.173611	79.806911	0.086528
DCP	0.137270	2.418509	134.438871	0.095526	74.231256	0.048041
RPM	**0.055095**	**1.012850**	**59.677372**	**0.010019**	**50.646270**	**0.005000**
FMR	0.090998	2.363750	137.574644	0.127406	77.094416	0.063657
DeepGMR	**0.000010**	**0.000050**	**0.015365**	**0.000002**	**0.001595**	**0.000001**
RGM	0.115444	2.642357	137.574520	0.737000	84.700130	0.365891
BTreeNet	0.067034	1.22952	64.790884	0.008188	57.402523	0.004077
IBTreeNet	**0.040702**	**0.920210**	**55.247189**	**0.007913**	**47.852112**	**0.003916**

$[0, 45°]$ to $[0, 180°]$ in ModelNet40 Clean Training Data (Section 4.1) and *ModelNet40 Clean* testing dataset, respectively. Other settings are not changed in these datasets.

As can be seen in Tables 9 and 1, even if all networks are trained on point clouds with large initial rotations, registration errors of all learning-based methods increase dramatically when the initial rotation ranges from 0 to 180 degrees, except DeepGMR [14]. DeepGMR [14] significantly outperforms other learning-based methods and traditional methods with large rotations and ranks first in all performing measures. DeepGMR [14] proposes a correspondence network and two differentiable computing blocks for solving the problem on large rotations. Although our IBTreeNet suffers on the large rotations, it still outperforms ICP [7], NDT [8], CPD [9], PointNetLK [10], DCP [11], FMR [13] and RGM [15]. The qualitative comparisons are shown in Figure 14.

PointNetLK [10], FMR [13], DCP [11] and RGM [15] do not converge during

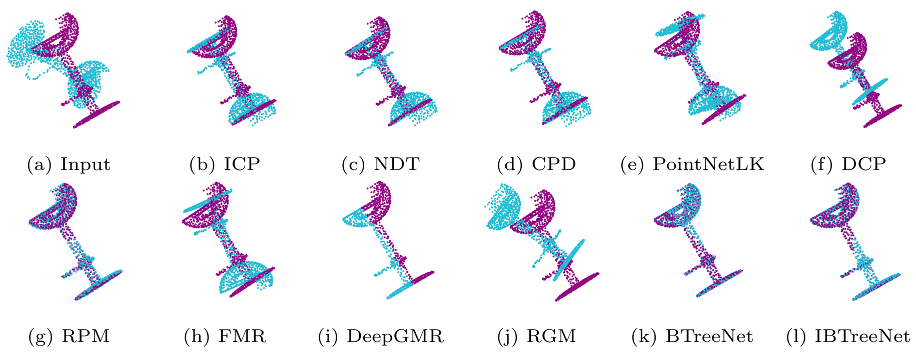


Figure 14: Registration results on 3D point clouds with large rotations.

Table 10: Ablation studies on our proposed binary tree and loss functions.

Methods	CD $\downarrow$	F Norm $\downarrow$ ($\mathbf{T}$)	MIE $\downarrow$ (Rot.)	MIE $\downarrow$ (Trans.)	MAE $\downarrow$ (Rot.)	MAE $\downarrow$ (Trans.)
without BT	0.021145	0.152280	7.028391	0.008171	4.090947	0.004103
BT-FC	0.016139	0.087589	3.511191	0.008165	1.771228	0.004099
BT-MLP	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988
BT-Supervision	0.019359	0.096562	3.885027	0.008143	1.984579	0.004091
CD	0.015577	0.085754	3.434221	0.636930	1.735888	0.004100
CD+EMD	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988

the training whereas RPM [12] and our methods converge to the local optima. One possible reason is that more point cloud pairs are needed to be trained under $[0, 180°]$ compared with that under $[0, 45°]$, which increases the variety and complexity of training point cloud pairs and is challenging to extract the generalized features for the increased varieties.

4.8. Ablation Studies

In this section, we present the results of the ablation studies to analyse the effectiveness of each component in our BTreeNet. In particular, we train all ablated models on the ModelNet40 Clean Training Data (Section 4.1) and test them on the *ModelNet40 Clean* testing datasets. The initial rotations in this

Table 11: Ablation studies on the weighted loss function. λ_1 and λ_2 denote the weights of CD and EMD, respectively.

Methods	CD ↓	F Norm ↓	MIE ↓	MIE ↓	MAE ↓	MAE ↓
		($\mathbf{T}$)	(Rot.)	(Trans.)	(Rot.)	(Trans.)
$\lambda_1 = 0.1; \lambda_2 = 0.9$	0.014545	0.078982	3.158734	0.008142	1.601008	0.004088
$\lambda_1 = 0.2; \lambda_2 = 0.8$	0.013478	0.074375	2.967007	0.008142	1.490887	0.004089
$\lambda_1 = 0.3; \lambda_2 = 0.7$	0.012810	0.071483	2.847388	0.008142	1.425907	0.004089
$\lambda_1 = 0.4; \lambda_2 = 0.6$	0.013562	0.075923	3.031684	0.008141	1.523975	0.004089
$\lambda_1 = 0.5; \lambda_2 = 0.5$	0.014778	0.080827	3.234577	0.008142	1.626481	0.004089
$\lambda_1 = 0.6; \lambda_2 = 0.4$	0.013858	0.076176	3.043757	0.008141	1.536414	0.004089
$\lambda_1 = 0.7; \lambda_2 = 0.3$	0.014435	0.079602	3.182555	0.008141	1.599607	0.004089
$\lambda_1 = 0.8; \lambda_2 = 0.2$	0.013205	0.073918	2.947582	0.008141	1.483749	0.004090
$\lambda_1 = 0.9; \lambda_2 = 0.1$	0.015058	0.084448	3.383622	0.008141	1.703698	0.004090
$\lambda_1 = 1.0; \lambda_2 = 1.0$	**0.012294**	**0.065951**	**2.618611**	**0.007907**	**1.379738**	**0.003988**

analysis are in the range of $[0, 45]$ degrees, and the initial translations are in the range of $[-0.5, 0.5]^3$.

We first analyse the effectiveness of our proposed binary tree-based forward propagation with the standard forward propagation without using a binary tree (without BT), as reported in Table 10. BT-FC and BT-MLP denote the fully connected layers and MLP models that are adopted in the binary tree, respectively. It is obvious that both BT-FC and BT-MLP outperform the standard forward propagation without using a binary tree, which shows the effectiveness of our proposed binary tree. Since BT-MLP outperforms BT-FC, the MLP models are adopted in the binary tree. The loss function of the CD shows the lower registration accuracy by comparing it with the combination of the CD and the EMD. We still evaluate our BTreeNet in a supervised manner following the supervised loss function in DCP [11], which minimizes the difference between the estimated transformation matrix and the ground-truth transformation matrix and is denoted as BT-Supervision. The combination of the CD and the

Table 12: Ablation studies on the feature extraction modules.

Methods	CD ↓	F Norm ↓	MIE ↓	MIE ↓	MAE ↓	MAE ↓
		(**T**)	(Rot.)	(Trans.)	(Rot.)	(Trans.)
BT-V1	0.012294	0.065951	2.618611	0.007907	1.379738	0.003988
BT-V2	0.066103	2.822350	174.840489	0.009611	100.515426	0.004796
BT-V3	0.045849	0.310504	12.621608	0.008175	6.370999	0.004105

Table 13: Ablation studies on the level of the binary tree.

Methods	MIE ↓	MIE ↓	MAE ↓	MAE ↓	F Norm ↓	CD ↓
	(Rot.)	(Trans.)	(Rot.)	(Trans.)	(**T**)	
BT-L2	3.624825	0.008165	1.824710	0.004099	0.090332	0.016149
BT-L3	3.068289	0.008164	1.536206	0.004099	0.076868	0.013486
BT-L4	2.830392	0.008165	1.416922	0.004099	0.071089	0.012618
BT-L5	**2.618611**	**0.007907**	**1.379738**	**0.003988**	**0.065951**	**0.012294**
BT-L6	13.380222	0.008176	7.443230	0.004106	0.286715	0.035227
BT-L7	19.369034	0.008166	8.403775	0.004099	0.406013	0.043825

EMD loss function also outperforms the supervised manner on our BTreeNet. Thus, we use the combination of the CD and the EMD as the final loss function. The weighted test of the final loss function is reported in Table 11. We select the best performance in all performing measures with $\lambda_1 = 1.0$ and $\lambda_2 = 1.0$ for CD and EMD, respectively.

We then analyse the effectiveness of three state-of-the-art feature extraction modules in PointNet [16], PointNet++ [29] and DGCNN [19] to select the most effective feature extraction module for our BTreeNet, and the resulting methods are denoted as BT-V1, BT-V2 and BT-V3. As reported in Table 12, the feature extraction module of MLP models in PointNet [16] outperforms others with the lowest average registration errors. PointNet++ [29] is a representative of the class of hierarchical feature extraction networks that aggregate local features before global pooling. PointNet [16] outperforms PointNet++ [29] in the en-

Table 14: Evaluations on 3D point clouds with the large rotations, Gaussian noise and partial visibility together. The initial rotation ranges from 0 to 180 degrees. The standard deviation of Gaussian noise is 0.01. The 70% of the points are retained in source 3D point cloud. All networks are trained on Modelnet40 Clean Training Data with the $[0, 180°]$ rotations without missing and noisy points. All networks are tested on *ModelNet40 Partial & Noise* with the random rotations from 0 to 180 degrees. Bold denotes the best performing measures.

Methods	CD ↓	F Norm ↓ (**T**)	MIE ↓ (Rot.)	MIE ↓ (Trans.)	MAE ↓ (Rot.)	MAE ↓ (Trans.)
ICP	0.127023	2.478114	140.039088	0.232349	73.982576	0.116232
NDT	0.163471	2.491908	138.536217	0.227821	76.550642	0.111283
CPD	0.097589	2.419856	140.816336	0.187100	80.316289	0.093111
PNLK	0.166495	2.438380	133.998653	0.264212	81.443547	0.131969
DCP	0.143335	2.418145	133.654160	0.096254	74.054247	0.048559
RPM	0.132424	1.384155	66.200785	0.205663	54.264680	0.102969
FMR	0.121134	2.395068	135.549886	0.215384	80.910621	0.106631
DeepGMR	0.140693	1.896580	97.199968	0.196313	72.883297	0.098153
RGM	0.446829	2.645025	136.359483	0.766860	84.548646	0.382923
BTreeNet	0.074901	1.310750	68.201930	0.008189	61.199467	0.004077
IBTreeNet	**0.058597**	**1.211751**	**60.334884**	**0.007321**	**50.298829**	**0.003508**

coder using the global pooling. We analyse that this is because local pooling is less stable than global pooling due to suboptimality in the selection of local neighbourhoods for the whole point cloud. Aggregating local features before global pooling results in unstable global features for our binary tree-based decoder to estimate the optimal rotation and translation. DGCNN [19] learns local geometric features via constructing the k points for each point and also aggregates local features using local pooling, which is also not suitable for our binary tree based decoder. In addition, PCN [41] and TopNet [40] achieve a similar conclusion for the problem of local pooling in PointNet++ [29] on the 3D point cloud completion task.

The number of binary tree level L in BTreeNet is examined by varying L

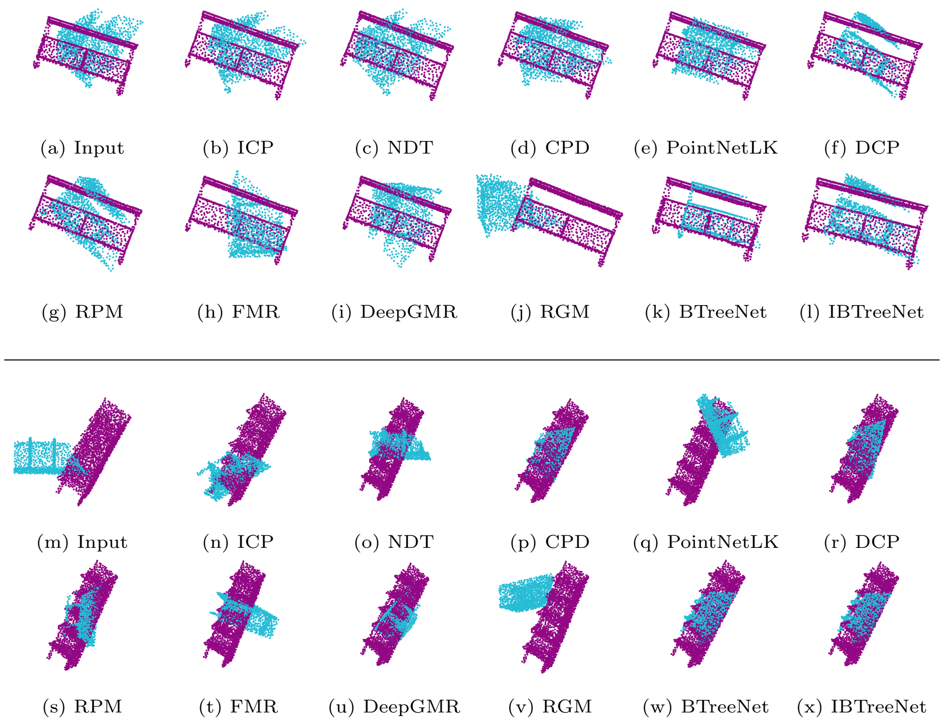

(a) Input (b) ICP (c) NDT (d) CPD (e) PointNetLK (f) DCP
(g) RPM (h) FMR (i) DeepGMR (j) RGM (k) BTreeNet (l) IBTreeNet
(m) Input (n) ICP (o) NDT (p) CPD (q) PointNetLK (r) DCP
(s) RPM (t) FMR (u) DeepGMR (v) RGM (w) BTreeNet (x) IBTreeNet

Figure 15: Failure cases on partial point clouds with 50% of missing data. The partial 3D point clouds should be transformed to the the same parts in target point clouds rather than the centre of the target point clouds.

in $2, 3, 4, 5, 6, 7$. The results are reported in Table 13, which shows 5 levels in BTreeNet has a good trade off between alignment accuracy and computational performance. As shown in the table, after a certain level the increase of the tree depth does not increase the accuracy of the registration. The reason is that 5 levels in BTreeNet already sufficiently include the majority of features in the alignment in this experiment.

5. Discussion and Limitation

We have identified two prominent failure cases for our BTreeNet and IBTreeNet. First, Our models fail to transform 3D point clouds with large missing data. As shown in Figure 15, 50% of the points are missing in the input source

3D point cloud. The partial point clouds should be transformed to the same parts in target point clouds rather than the centre of the target point clouds, as shown in Figure 15 (k), (l), (w) and (x). We analyse that the loss function of our combined CD and EMD finds the global registration between two shapes without considering the local to global registration. However, other supervised learning-based methods still meet this problem because they force the network to output a certain transformation based on a certain pattern of input 3D point cloud pairs by using the supervised manner and still do not consider the local to global registration in their loss function.

Second, our models fail to transform 3D point cloud with the combination of partial overlap, noise and large initial rotations. To evaluate the robustness of all algorithms on 3D point cloud pairs with the combination of these scenarios, we generate a *ModelNet40 Partial & Noise* testing dataset from *ModelNet40 Clean*. Specifically, for each source 3D point cloud in *ModelNet40 Clean*, we first sample noise from Gaussian distribution with a standard deviation of 0.01, and then create a random clipping plane passing through the origin and shift it to retain 70% of the points. All networks are trained on ModelNet40 Clean Training Data (Section 4.1) with random rotations from 0 to 180 degrees. We test all trained models on *ModelNet40 Partial & Noise* with random rotations from 0 to 180 degrees. The performance of each method on *ModelNet40 Partial & Noise* with the large rotation is reported in table 14. Both traditional and learning-based methods fail to achieve accurate registration, which indicates the poor robustness and generalization ability of each network under such challenging conditions.

DeepGMR [14] shows the best performance on 3D point clouds with point-to-point correspondences (Sections 4.3, 4.6 and 4.7), however, it shows poor robustness and generalization ability on partial visibility and noise with broken point-to-point correspondences, as reported in Tables 3 and 4. Thus, DeepGMR [14] fails under the combination of partial overlap, noise and large rotations. Although our BTreeNet and IBTreeNet still do not achieve accurate registration in this challenging condition, they achieve the best performing measures based on the overall registration errors. We analyse the possible reasons are: (i) Our methods can tolerate noise and partial overlap to some extent without training them in these scenarios (Sections 4.4 and 4.5); (ii) Although our methods suffer on the large rotations, they still outperform ICP [7], NDT [8], CPD [9], PointNetLK [10], DCP [11], FMR [13] and RGM [15] (Section 4.7).

6. Conclusion

A novel unsupervised deep learning network – BTreeNet is proposed for 3D point cloud registration. BTreeNet consists of a hierarchical binary tree-based forward propagation that learns features for the rotation separately from the translation and avoids the interference between the estimations of rotation and translation in one single matrix. Based on the BTreeNet, IBTreeNet is proposed to iteratively rotates and translates the source 3D point cloud to the target. With an identical network architecture to BTreeNet, the IBTreeNet is trained based on the registration result of a trained BTreeNet model. Once trained, the IBTreeNet model can be reused and iteratively improve the registration accuracy. Our method achieves precise registration and shows remarkable generalization and robustness to unseen outdoor and indoor scenes that are not trained. In future work, more accurate registration of partial 3D point clouds with large missing rates would be an interesting direction to pursue.

7. Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] F. Engelmann, K. Rematas, B. Leibe, V. Ferrari, From points to multi-object 3d reconstruction, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 4586–4595. `doi:https://doi.org/10.1109/CVPR46437.2021.00456`.

[2] C. Chen, D. Liu, C. Xu, T.-K. Truong, Genecgan: A conditional generative adversarial network based on genetic tree for point cloud reconstruction, Neurocomputing 462 (2021) 46–58. doi:https://doi.org/10.1016/j.neucom.2021.07.087.
URL https://www.sciencedirect.com/science/article/pii/S0925231221011693

[3] L. Xi, Y. Zhao, L. Chen, Q. H. Gao, W. Tang, T. R. Wan, T. Xue, Recovering dense 3d point clouds from single endoscopic image, Computer Methods and Programs in Biomedicine 205 (2021) 106077. doi:https://doi.org/10.1016/j.cmpb.2021.106077.

[4] X. Wang, M. Cai, F. Sohel, N. Sang, Z. Chang, Adversarial point cloud perturbations against 3d object detection in autonomous driving systems, Neurocomputing 466 (2021) 27–36. doi:https://doi.org/10.1016/j.neucom.2021.09.027.
URL https://www.sciencedirect.com/science/article/pii/S0925231221013850

[5] W. Lu, Y. Zhou, G. Wan, S. Hou, S. Song, L3-net: Towards learning based lidar localization for autonomous driving, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. doi:https://doi.org/10.1109/CVPR.2019.00655.

[6] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, S. Mita, Lidar scan feature for localization with highly precise 3-d map, in: Intelligent Vehicles Symposium, 2014. doi:https://doi.org/10.1109/IVS.2014.6856596.

[7] P. J. Besl, N. D. Mckay, A method for registration of 3-d shapes, Proceedings of Spie the International Society for Optical Engineering 14 (3) (1992) 239–256. doi:https://doi.org/10.1109/34.121791.

[8] P. Biber, W. Strasser, The normal distributions transform: a new approach to laser scan matching, in: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat.

No.03CH37453), Vol. 3, 2003, pp. 2743–2748 vol.3. doi:https://doi.org/10.1109/IROS.2003.1249285.

[9] A. Myronenko, X. Song, Point set registration: Coherent point drift, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (12) (2010) 2262–2275. doi:https://doi.org/10.1109/TPAMI.2010.46.

[10] Y. Aoki, H. Goforth, R. A. Srivatsan, S. Lucey, Pointnetlk: Robust & efficient point cloud registration using pointnet, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. doi:https://doi.org/10.1109/CVPR.2019.00733.

[11] Y. Wang, J. Solomon, Deep closest point: Learning representations for point cloud registration, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019. doi:https://doi.org/10.1109/ICCV.2019.00362.

[12] Z. J. Yew, G. H. Lee, Rpm-net: Robust point matching using learned features, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 11821–11830. doi:https://doi.org/10.1109/CVPR42600.2020.01184.

[13] X. Huang, G. Mei, J. Zhang, Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 11363–11371. doi:https://doi.org/10.1109/CVPR42600.2020.01138.

[14] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, J. Kautz, Deepgmr: Learning latent gaussian mixture models for registration, in: European Conference on Computer Vision (ECCV), Springer, 2020, pp. 733–750. doi:https://doi.org/10.1007/978-3-030-58558-7_43.

[15] K. Fu, S. Liu, X. Luo, M. Wang, Robust point cloud registration framework based on deep graph matching, in: 2021 IEEE/CVF Conference on

Computer Vision and Pattern Recognition (CVPR), 2021, pp. 8889–8898. `doi:https://doi.org/10.1109/CVPR46437.2021.00878`.

[16] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017, pp. 652–660. `doi:https://doi.org/10.1109/CVPR.2017.16`.

[17] S. Baker, I. Matthews, Lucas-kanade 20 years on: A unifying framework, International Journal of Computer Vision 56 (3) (2004) 221–255. `doi:https://doi.org/10.1023/B:VISI.0000011205.11775.fd`.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems (NIPS), Vol. 30, Curran Associates, Inc., 2017, pp. 5998–6008.

[19] Y. Wang, Y. Sun, Z. Liu, S. Sarma, M. Bronstein, J. Solomon, Dynamic graph cnn for learning on point clouds, ACM Transactions on Graphics 38 (2019) 1–12. `doi:https://doi.org/10.1145/3326362`.

[20] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3354–3361. `doi:10.1109/CVPR.2012.6248074`.

[21] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, T. Funkhouser, 3dmatch: Learning local geometric descriptors from rgb-d reconstructions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 199–208. `doi:10.1109/CVPR.2017.29`.

[22] M. L. Greg Turk, The stanford 3d scanning repository, in: Stanford University Computer Graphics Laboratory, 2005.
URL `http://graphics.stanford.edu/data/3Dscanrep`

[23] H. Fan, H. Su, L. J. Guibas, A point set generation network for 3d object reconstruction from a single image, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017, pp. 605–613. doi:https://doi.org/10.1109/CVPR.2017.264.

[24] Y. Rubner, C. Tomasi, L. J. Guibas, The earth mover's distance as a metric for image retrieval, International journal of computer vision 40 (2) (2000) 99–121.

[25] S. Bouaziz, A. Tagliasacchi, M. Pauly, Sparse iterative closest point, Computer Graphics Forum 32 (5). doi:https://doi.org/10.1111/cgf.12178.

[26] A. W. Fitzgibbon, Robust registration of 2d and 3d point sets, Image & Vision Computing 21 (13–14) (2001) 1145–1153. doi:https://doi.org/10.1016/j.imavis.2003.09.004.

[27] J. Vongkulbhisal, F. De la Torre, J. P. Costeira, Discriminative optimization: Theory and applications to point cloud registration, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3975–3983. doi:https://doi.org/10.1109/CVPR.2017.423.

[28] Y. Zhao, W. Tang, J. Feng, T. Wan, L. Xi, Reweighted discriminative optimization for least-squares problems with point cloud registration, Neurocomputing 464 (2021) 48–71. doi:https://doi.org/10.1016/j.neucom.2021.08.080.
URL https://www.sciencedirect.com/science/article/pii/S0925231221012765

[29] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 5105–5114.

[30] S. A. Bello, C. Wang, N. M. Wambugu, J. M. Adam, Ffpointnet: Local and global fused feature for 3d point clouds analysis, Neurocomputing 461 (2021) 55–62. doi:https://doi.org/10.1016/j.neucom.2021.07.044.
URL https://www.sciencedirect.com/science/article/pii/S0925231221010882

[31] G. Zhu, Y. Zhou, J. Zhao, R. Yao, M. Zhang, Point cloud recognition based on lightweight embeddable attention module, Neurocomputing 472 (2022) 138–148. doi:https://doi.org/10.1016/j.neucom.2021.10.098.
URL https://www.sciencedirect.com/science/article/pii/S0925231221016477

[32] H. Liu, Y. Cong, C. Yang, Y. Tang, Efficient 3d object recognition via geometric information preservation, Pattern Recognition 92 (2019) 135–145. doi:https://doi.org/10.1016/j.patcog.2019.03.025.

[33] Q. Xie, Y.-K. Lai, J. Wu, Z. Wang, Y. Zhang, K. Xu, J. Wang, Vote-based 3d object detection with context modeling and sob-3dnms, International Journal of Computer Vision 129. doi:https://doi.org/10.1007/s11263-021-01456-w.

[34] X. Pan, Z. Xia, S. Song, L. E. Li, G. Huang, 3d object detection with pointformer, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 7459–7468. doi:https://doi.org/10.1109/CVPR46437.2021.00738.

[35] J. Li, B. M. Chen, G. Lee, So-net: Self-organizing network for point cloud analysis, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 9397–9406. doi:10.1109/CVPR.2018.00979.
URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00979

[36] F. Hao, R. Song, J. Li, K. Cao, Y. Li, Cascaded geometric feature modulation network for point cloud processing, Neurocomputing 492 (2022)

474–487. doi:https://doi.org/10.1016/j.neucom.2022.04.007.
URL https://www.sciencedirect.com/science/article/pii/S0925231222003757

[37] Y. Cui, X. Liu, H. Liu, J. Zhang, A. Zare, B. Fan, Geometric attentional dynamic graph convolutional neural networks for point cloud analysis, Neurocomputing 432 (2021) 300–310. doi:https://doi.org/10.1016/j.neucom.2020.12.067.
URL https://www.sciencedirect.com/science/article/pii/S0925231220319676

[38] H. Xie, H. Yao, S. Zhang, S. Zhou, W. Sun, Pix2vox++: Multi-scale context-aware 3d object reconstruction from single and multiple images, International Journal of Computer Vision 128. doi:https://doi.org/10.1007/s11263-020-01347-6.

[39] B. Yang, S. Wang, A. Markham, N. Trigoni, Robust attentional aggregation of deep feature sets for multi-view 3d reconstruction, International Journal of Computer Vision 128. doi:https://doi.org/10.1007/s11263-019-01217-w.

[40] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, S. Savarese, Topnet: Structural point cloud decoder, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 383–392. doi:https://doi.org/10.1109/CVPR.2019.00047.

[41] W. Yuan, T. Khot, D. Held, C. Mertz, M. Hebert, Pcn: Point completion network, in: 2018 International Conference on 3D Vision (3DV), 2018, pp. 728–737. doi:https://doi.org/10.1109/3DV.2018.00088.

[42] X. Bai, Z. Luo, L. Zhou, H. Chen, L. Li, Z. Hu, H. Fu, C. Tai, Pointdsc: Robust point cloud registration using deep spatial consistency, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp.

15854–15864. doi:10.1109/CVPR46437.2021.01560.
URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.01560

[43] Z. Dong, B. Yang, Y. Liu, F. Liang, B. Li, Y. Zang, A novel binary shape context for 3d local surface description, ISPRS Journal of Photogrammetry and Remote Sensing 130 (2017) 431–452. doi:https://doi.org/10.1016/j.isprsjprs.2017.06.012.
URL https://www.sciencedirect.com/science/article/pii/S0924271617300199

[44] Z. Dong, B. Yang, F. Liang, R. Huang, S. Scherer, Hierarchical registration of unordered tls point clouds based on binary shape context descriptor, ISPRS Journal of Photogrammetry and Remote Sensing 144 (2018) 61–79. doi:https://doi.org/10.1016/j.isprsjprs.2018.06.018.
URL https://www.sciencedirect.com/science/article/pii/S0924271618301813

[45] Z. Dong, F. Liang, B. Yang, Y. Xu, Y. Zang, J. Li, Y. Wang, W. Dai, H. Fan, J. Hyyppä, U. Stilla, Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark, ISPRS Journal of Photogrammetry and Remote Sensing 163 (2020) 327–342. doi:https://doi.org/10.1016/j.isprsjprs.2020.03.013.
URL https://www.sciencedirect.com/science/article/pii/S0924271620300836

[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1912–1920. doi:https://doi.org/10.1109/CVPR.2015.7298801.

[47] G. Turk, B. Mullins, Large geometric models archive, Georgia Inst. Technology.
URL https://www.cc.gatech.edu/projects/large_models/