Article

Extract Descriptors for Point Cloud Registration by Graph Clustering Attention Network

Yapeng Ren ¹, Wenjie Luo ^{1,2,*}, Xuedong Tian ^{1,2} and Qingxuan Shi ^{1,2}

¹ School of Cyber Security and Computer, Hebei University, Baoding 071000, China; lwj12111@hbu.edu.cn (Y.R.); xdtian@hbu.edu.cn (X.T.); shiqingxuan@hbu.edu.cn (Q.S.)

² Laboratory of Intelligence Image and Text, Hebei University, Baoding 071000, China

* Correspondence: luowenjie@hbu.edu.cn; Tel.: +86-188-3029-9830

Abstract: Extracting geometric descriptors in 3D vision is the first step. It plays an important role in 3D registration, 3D reconstruction, and other applications. The success of many 3D tasks is closely related to whether the geometric descriptor has accurate characteristics. Today, the main methods are divided into manual production and neural network learning. The applicability of descriptors is limited to a low-level point, corner, edge, and fixed neighborhood features. For this, we use the class attention of the point cloud. In order to extract class attention, the graph clustering approach is utilized. It can collect points with similar structures and divide regions dynamically. While maintaining rotation invariance, features can enhance their fit to the original data. Point attention and edge attention are used to describe the structural characteristics of point clouds. We combine the three attentions indicated before to improve the features obtained by the PointNet decoder. This feature can dynamically reflect the structure of the point cloud, which includes both soft shape information and rich detail information. Finally, the 3D descriptors are extracted with the FoldingNet decoder. Our method is validated on both indoor and outdoor datasets. The accuracy of the final result is improved by two percentage points.

Keywords: 3D machine vision; point cloud registration; graph cluster; feature fusion

Citation: Ren, Y.; Luo, W.; Tian, X.; Shi, Q. Extract Descriptors for Point Cloud Registration by Graph Clustering Attention Network. *Electronics* **2022**, *11*, 686. <https://doi.org/10.3390/electronics11050686>

Academic Editor: Byung-Gyu Kim

Received: 19 January 2022

Accepted: 22 February 2022

Published: 23 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Now, 3D machine vision has made substantial progress. Among these, effectively describing the characteristics of a 3D object has emerged as a critical challenge that must be handled in a number of jobs. The primary task of describing 3D objects has steadily become establishing realistic geometric relationships [1–9] for 3D objects, notably, point clouds. An accurate 3D descriptor can accurately reflect the features of 3D objects. Accurate and fast descriptors can be employed in other 3D applications to gain the most fundamental advantages.

Manual design descriptors and learning-based descriptors are two prevalent ways. The manual design descriptors use the reference coordinate system, as well as feature histogram and other methods. It extracts low-level point, corner, line, and neighborhood information to learn features. This method works well in certain situations and produces excellent results despite a number of limitations, such as low resolution, incompleteness, and the difficulty of matching geometric features. However, faced with various problems in the real world, such as more noise and complex objects, it cannot create suitable key descriptors for point clouds due to noise or inadequate information. There is still a lot of potential for improvement in the repeatability and distinguishability of key descriptors.

Learning-based approaches have received a lot of attention because of their robustness and higher efficacy. Learning-based methods rely on low-level geometric features. Angular deviation [1–3], point distribution [4,5], and volume distance function [9,10] are the three basic categories of learning approaches. They extract high-dimensional features from the

point cloud and map them to a low-dimensional feature space using MLPs (multilayer perceptrons) or 3D convolution. These methods are computationally expensive. It takes low-level features such as point pairs, angles, normals, etc., and focuses on the information within a point and its fixed neighborhood. For information with similar structures in 3D objects, there is no more subtle and accurate feature extraction.

The aim of the work is to find accurate 3D descriptors that can be used in a variety of situations. We use a PointNet decoder to extract features and enhance them with attention obtained from point, edge, and cluster information. Unlike previous research, this feature can dynamically acquire neighborhood information for similar structures. In the real world, the scales of different objects are different. The regular space mapped from the reference coordinate system does not well characterize objects in the scene. Compared with it, a cluster can produce irregular regional divisions. At the same time, different cluster features will be generated for different input data in each frame, and the captured data is more adaptable. The flexible clustering radius will be realistically altered by network parameters. It works well with 3D objects of various sizes. At the same time, features used in this paper do not destroy their rotational invariance.

Specifically, we employ a graph network with information from points, edges, and classes. It improves feature learning in communities with similar structures. In the task of key descriptors, most only focus on a low-level corner, edge, and line features. They are weak in generality in certain ways. We utilize graph clustering to group points with similar characteristics. In order to calculate similarity and validity, the degree matrix and adjacency matrix of the points are employed to record the self-learning weight of each category. The points with similar correlations are then grouped together. The goal of edge attention is to determine the edge weight between each pair of points. It uses the standard convolutional approach to collect local and global relationship properties. Class attention and edge attention have separate scales. We must bring them all into the same dimension in order to combine an attention matrix. We combine attention with PointNet [11] encoder features using graph convolution and then obtain descriptors using a FoldingNet [12] decoder.

The second chapter, related work, concentrates on the excellent work of manual design descriptors and learning-based descriptors, as well as their limits. It also summarizes graph clustering and graph convolution theory. The third chapter introduces the overall structure of the model. Sections 3.1–3.5 introduce the concept of attention, particularly class attention. Encoders with graph attention and decoders are discussed in Section 3.6. Section 3.7 presents the loss function. The data and evaluation metrics are presented in the fourth chapter. It includes experimental data as well as comparisons to other superior methods.

2. Related Works

2.1. Key Descriptors

2.1.1. Manual Design Key Descriptors

Registration, reconstruction, and other 3D activities are all based on feature descriptors. The manual design produces descriptors quickly.

To locate the proper axis, spin-images [4] use the matching of distinct surface patches. It is used to figure out a 3D shape. Spin-images have been improved by TriSI [13]. Principal component analysis (PCA) minimizes dimensionality and fuses point local information. With dimensionality reduction, the algorithms above project a 3D point cloud into 2D space.

Three-dimensional SC descriptors are more common than spin-images descriptors. The normal vector of a point is taken as the north pole direction of its spherical neighborhood in the 3D shape context [14]. The radius, longitude, and latitude were used to partition the region. It counts the weighted total of each region for each area. SHOT (signatures of histograms of orientations) [6] divides zones along radial and dimensions using spherical neighborhoods. It calculates the normal vector angle and cosine value distribution. It uses two-line interpolation to learn histogram properties. On the z-axis, ISS (Intrinsic Shape Signatures) [15] uses a separate shape descriptor from the surface normal line. It iterated the best descriptor in the spherical neighborhood using the k-Nearest Neighbor (KNN).

Feature histograms are the alternative method. By projecting the distribution map on three planes, RoPS (Rotational Projection Statistics) [16] creates a histogram. It obtains anti-noise and anti-interference characteristics by rotating the center distance. The programs PFH (point feature histogram) [7] and FPFH (fast point feature histogram) [8] compute histograms using a normal vector. It describes the geometric information in the vicinity of a point. To pick the optimal descriptors, 3DhoPD [17] uses nearest neighbor search to compute the geometric distance among point clouds in 3D space.

2.1.2. Learning-Based Key Descriptors

Learning to produce 3D descriptors has become popular due to the adaptability of neural networks.

The Siamese network with share weight is used by many methods. To learn 3D descriptors, Zeng et al. [9] employ a Siamese convolutional network. It established the local data's corresponding association. To extract the features of outside images, Yew and Lee [18] employed a three-branch Siamese convolutional network.

Except for these, the PointNet network is commonly utilized. PointNet is efficient and quick for practically any point cloud. The multilayer perceptron is used by Khoury et al. [5] to transfer the 3D histogram from high-dimensional space to low-dimensional space. Deng et al. [2,3] train shape features using PointNet and FoldingNet. This network only analyzes point relationships and is devoid of coordinate and normal vector properties.

Choy et al. [19] extract features using a fully convolutional network. It employs the ResUNet [20] structure to deepen the network's learning and produce a denser output.

2.2. Graph Clustering and Graph Convolution

The graph approach includes graph clustering and graph convolution. Point cloud is a non-Euclidean geometric form that is well suited to the use of a graph. We look at the characteristics of a point cloud and establish how to learn features using a graph. The feature vectors of the sample's Laplacian matrix are analyzed to perform graph clustering.

The goal of graph clustering is to group points together that have a similar structure. Dense edges and tighter connections characterize vertices in the same cluster. The approach of graph clustering has been extensively researched as a fundamental topic [21–26]. Initially, graph convolution relied on modular [25–27] and density-based methods ([28]). These methods simply categorize the members of the graph without examining the various influences of the member points. Structured graph clustering was proposed by SCAN++ () [24]. It is the first to employ structural similarity to depict the vertices' proximity relationship. The number of points in the public domain between two vertices is represented by structural similarity $\delta(u, v)$. It used its degrees to standardize the average. The larger, the closer u and v are. By sampling the picture edges, LinkSCAN (Linkspace Structural Clustering Algorithm for Networks) [29] presented an approximation technique that reduces the calculation of structural similarity and improves scanning efficiency.

3. Network

The graph clustering point cloud is used to acquire fine and exact features. Non-Euclidean structures are well-suited to graph theory, which excels at dealing with point-point and edge-edge relationships. The following are the steps that make up our network:

1. Up-sampling for the point cloud and graph structure generation;
2. Analysis cluster on the generated graph;
3. Fusion of class and edge attention;
4. The hierarchical PointNet with deep features and attention;
5. Generating accurate and efficient 3D key descriptors.

3.1. Establish Graph

The coordinate information and other properties of the point cloud (color, normal, etc.) are used as input. The coordinate feature size is $N \times 3$. To sample the points, we employ

FPS (farthest point sampling). The final size is $M \times 3$. We establish graph $G = (V, E)$. V is the set of points. E is the set of edges. We calculate the Euclidean Distance for every point in the set. As shown in Formula (1), Euclidean Distance is used for adjacency matrix A .

$$\rho = \sqrt[2]{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (1)$$

3.2. Class Attention

Graph clustering attention and increased learning attention make up class attention. First, we will go through how to gain class attention using the graph clustering method. The main process is as Algorithm 1. We initialize the core points in step 1 of Algorithm 1. The number of core points is k . To build *cornset* and *noncornset*, all points are simply grouped in steps 2 and 3. The core point set contains k core-point, while the non-core set contains the remaining points. The goal of grouping is to make the clustering data more visible and to make the calculation of similarity and validity easier. In the point set, we set the edge information to empty, and we calculate it only when it needs to be calculated. Next, as steps 4, 5, and 6, we initialize the similarity and validity of each point. The similarity is set to zero, and the validity is determined using the radius of the neighborhood R . The following two chapters will cover the next steps, which are coarse and fine clusters. Following the fine cluster, we re-analyze each class to reselect the core point. It is necessary to iterate Algorithm 1 numerous times in order to acquire a satisfactory local optimal outcome.

Algorithm 1. Graph Cluster.

```

input: graph :  $G(V, E)$ , Initial radius of neighborhood:  $r$ ,
output: cluster sets of  $G$ :  $C_f$ 
1   Initialize corn – points;
2   Build cornset;
3   Build noncornset; //build cornset :  $G_c(V_c, E_c, W_c)$  and noncornset :  $G_n(V_n, E_n, W_n)$ 
4   For each vertex  $u \in V$  do
5      $sd(u) \leftarrow 0$ ; //initialize similar degree
6      $ed(u) \leftarrow d[u]$ ; // initialize effective degree
7      $C_c = \text{coarse cluster}()$ ; The same non-core point can have different categories.
8      $C_f = \text{fine cluster}()$ ; //fine cluster. Calculate structural similarity between nodes and cluster
      centers to achieve a non-core point having one center.
9     reset corn – points(); //reselect the core point.
10  Return  $C_f$ ;

```

We apply coarse clusters to all non-core points, as illustrated in Algorithm 2. Its procedure is straightforward. R is the original radius. We count all the non-core points in the spherical neighborhood of each core point, and they're all lumped together in one category. There will be a question following this approximate categorization. It is a fact that a non-core point might exist multiple times in a cluster of core points. As a result, deleting duplicate categories necessitates a fine cluster. It aspires to have only one most suitable cluster center for each non-core point. In the circumstance depicted in Figure 1, the nearest cluster core to a non-core point must be found. Coarse clustering can assist fine clustering in reducing calculation time. It also preferentially filters out an appropriate collection of points.

Algorithm 3 displays a fine cluster. The goal of fine clustering is to optimize clustering data. It can discover the most similar core point for each non-core point. The issue we have is that a non-core point is concerned with many core points at the same time. By examining each non-core point, we were able to choose a superior cluster center. To begin with, all core points in the neighborhood that contain non-core points are created as a set. Then, as indicated in Formula (2), we employ structural similarity to choose the best cluster center. Both μ and ν are distinct points. $N[\mu]$ represent the points in the neighborhood of μ with the radius r . $|N[\mu] \cap N[\nu]|$ means the common vertex in the neighborhood of the

point μ and the point v . $d(\mu)$ represents the number in the μ 's neighborhood. The range of structural similarity ss is $[0, 1]$.

Algorithm 2. Coarse Cluster.

Input: $cornset : G_c(V_c, E_c, W_c)$, $noncornset : G_n(V_n, E_n, W_n)$, radius of neighborhood: r
 output: cluster set C_C

```

1  For  $m$  in  $V_c$ : // Traverse all points in  $V_c$ 
2  For  $n$  in  $V_n$ :
3    If  $distance(m, n) < r$ :
4       $C_m \leftarrow C_m \cup \{n\}$ 
5   $C_C \leftarrow C_C \cup C_m$ 
  
```

Figure 1. Core-point cluster. The color black is the core point. The current cluster range is indicated by the dashed line. The red point should be part of the current core point. The blue dot is not in the right category right now.

$$ss(\mu, v) = \frac{|N[\mu] \cap N[v]|}{\sqrt{d(\mu) \cdot d(v)}} \quad (2)$$

Algorithm 3. Fine Cluster.

Input : $cornset : G_c(V_c, E_c, W_c)$, $noncornset : G_n(V_n, E_n, W_n)$, cluster set C_C
 output: new cluster set C_f

```

1  For  $n$  in  $V_n$ : // Traverse all non-core points in the set of non-core points
2  For  $m$  in  $clu(n)$ : // Traverse the core points associated with each non-core point
3    if  $ss(m, n) > max$ :
4       $max = ss(m, n)$ 
      // the calculation of maximum structural similarity
5   $updatecluster(m, n)$ 
      // Update cluster information
  
```

A non-core point with a single class is determined by a fine cluster. The main points of each category, however, may change. We re-analyze each class in the cluster by calculating similarity and validity, as illustrated in Algorithm 4. The Formula (3) for calculating similarity sd is as follows.

$$sd(v) = num(ss_{2\varepsilon}[v]) \quad (3)$$

$ss_\varepsilon[v]$ represents all points whose structural similarity is more than ε in the neighborhood. $num()$ is a function that calculates the number. The formula for the effective degree equation is as follows (4).

$$ed(v) = num(ss_\varepsilon[v]) \quad (4)$$

The sd is the number of points with structural similarity more than 2ε in the neighborhood. The ed is the number of points with structural similarity more than ε in the point

neighborhood. For each point in the cluster, we calculate the similarity and validity. Set the current point's confidence degree (cd) (the formula is shown in Algorithm 4). We calculate the degree of confidence for each point in each class. The cluster center within the class, which is the core point, has the highest point value. Instead of cluster centers, we use core points. The point weight of each core point and its confidence degree, as given in Formula (5), determine the weight of the class attention.

$$W_{cls} = W_c \cdot kd \quad (5)$$

Algorithm 4. Reselect the Core Point.

Input : $cornset : G_c(V_c, E_c, W_c)$, $noncornset : G_n(V_n, E_n, W_n)$, cluster set C_f
 output: new cluster set C_f

- 1 For c in C_f : // Analyze each class in the cluster set
- 2 For v in c : // Calculate sd and ed for each point in the class
- 3 $cd(v) = \lambda sd(v) + ed(v)$ // Calculate the confidence of each point
- 4 $v = \max(cd)$ // The highest value of its cd is the core point
- 5 $V_c \leftarrow V_c \cup \{v\}$

3.3. Fixed and Variable Clustering Radius

It is necessary for iterating the algorithm multiple times and selecting the best core points and point sets. To choose the neighborhood radius, we devised two methods: fixed and variable value.

Every iteration, the fixed neighborhood radius utilizes a fixed value R . In these categories, fine clustering can locate suitable core and non-core points. The cluster center will also be upgraded at the same time. The constant radius indicates that each algorithm's attention range is constant, but it does not imply that the iteration is meaningless.

We devised an algorithm to modify the neighborhood radius in order to adaptively adjust attention. It is determined after the core points have been updated. We identify the closest negative point (non-similar point) and the farthest positive point (similar point) for each core point. The Formula (6) for calculating radius distance is presented below.

$$D_c = \left| \frac{[d(p_p, p_{c_1})]^2}{r_{c_1}} - \frac{[d(p_n, p_{c_2})]^2}{r_{c_2}} \right| + d(p_p, p_{c_1}) \quad (6)$$

The farthest positive point is p_p . It belongs to c_1 class. The nearest negative point is p_n . It belongs to c_2 class. The weight difference between the nearest negative point and the farthest positive point is found in the first part of the formula. The distance between the non-core point and the core point determines the weight, which is proportional to the distance and radius. $d(p_p, p_{c_1})$ represents the distance between the farthest positive point and the core point p_{c_1} . $\frac{d(p_p, p_{c_1})}{r_{c_1}}$ means the weight of the farthest positive point. The nearest negative point is calculated in the same way as the farthest positive point. The last part of the formula is the distance between the farthest positive points. This distance $\leq D_c$ can promise the negative point not missing. The radius distance improves as the first half of the formula approaches zero.

3.4. Edge Attention

The distinctive relationship between edges is calculated through edge attention. We capture the shape information of the point cloud by learning the features between the edges.

In order to acquire features, this section uses graph convolution. $G = (V, E)$ is a graph that has been transformed from point cloud. For point clouds, graph convolution is an effective method to extract detail features, which can express rich edge-edge relationships. We begin by calculating the feature matrix. The graph G is linked, and each point has a degree of $n - 1$. As a result, the information of the Laplacian matrix L and the adjacency matrix A is nearly identical. The adjacency matrix is used to directly replace the Laplace

matrix. a_{ij} represents every element in the matrix. i is the destination node. j is another endpoint connected to i . Initial weight is the Euclidean distance of point i and point j . Figure 2 describes the feature extraction procedure. The input is adjacency matrix A . It has m feature layers, which means a relationship matrix of m points. The size of every layer matrix is $1 \times m$. The horizontal and vertical features of the adjacency matrix are the same because it is a symmetric matrix. The calculating process is illustrated in Formula (7) in our module, which uses two various approaches to learn features. To begin, we utilize a large number of 2D convolution layers to obtain the edge feature m_1 , which has a size of $1 \times m$. Second, utilizing average pooling and 2D convolutions, the edge feature m_2 size of $m \times 1$ is obtained.

$$\begin{cases} m_1 = \text{relu}(\text{bn}(\text{conv2d}(A))) \\ m_2 = \text{relu}(\text{bn}(\text{conv2d}(\text{avgpool}(A)))) \end{cases} \quad (7)$$

Figure 2. Edge attention generation network. Different color blocks represent different features. A is the adjacency matrix.

To achieve a new weight matrix with the size of $m \times m$, we apply matrix multiplication. As seen in Formula (8), we put the weight matrix and adjacency matrix into the network. Multiple 2D convolution layers will be used to learn the edge features of a single point in the adjacency matrix. We also achieve an edge attention matrix with a dimension of $m \times 1$.

$$X(i) = \frac{1}{N} \sum_{j \in N} F_i(A(i, j), \omega_i) + b_j \quad (8)$$

3.5. Graph Feature Fusion

At this time, class and edge attention can be achieved. We combine them to create a single and consistent attention matrix. The class attention A_{cls} has a size of $k \times 1$. Its weight is the multiplication of the weight of cluster centers and the validity of cluster centers. The size of A_{edge} 's edge attention is $m \times 1$. Every point's weight is equal to the total of its edge weights.

Two attention features are normalized, and class and edge attention are merged to the $m \times 1$ dimension. The edge attention does not need to be altered because its input has m points. The weight of each point is equivalent to the weight of the center in the class attention adjustment approach.

The feature fusion module we selected is shown in Figure 3a. The edge feature and the class feature have the same size. Choose X and Y to represent the edge and class features, respectively (X and Y are indistinguishable). We start by splicing X and Y to achieve an $m \times 2$ matrix. It is stored in the multi-scale fusion module (Figure 3b). The goal is to enhance features and generate a $m \times 2$ attention matrix. This attention matrix is retained for the following X and Y step. The goal is to increase the efficiency of X and Y . A matrix with the size of $m \times m$ is obtained by multiplying X and Y . Then we use the multilayer perceptron (MLP) to learn the features obtaining feature whose size is $m \times 1$. After combining these two attributes, we achieve aggregated features.

Figure 3. (a) is Feature fusion module. X and Y represent two input features, respectively. (b) is multi feature fusion (MFF). Z is the input data.

From two pathways, the MFF module deepens the point feature. The input is Z, and its size is $m \times 2$. The first is to aggregate the features of each point using average pooling to build a 1×2 feature matrix. The second method employs a multilayer perceptron (MLP) to enhance each point's characteristics and build an $m \times 1$ matrix. The output matrix Z of $m \times 2$ is obtained by multiplying the two matrices.

3.6. PointNet

This section covers how to extract original point cloud features as well as how to use the graph attention mechanism.

As the network's basic architecture, we adopt a three-encode-decode structure. PointNet is used in the encoder, and FoldingNet is used in the decoder. PointNet retains its inherent benefits in terms of speed and accuracy for extracting point cloud features as a basic point network structure. A simple network structure can also be used as a basic module for collaborating with other networks. Deformation allows the Folding to decode better surface shapes. It is a fantastic decoder for 3D point clouds.

Figure 4 depicts our network, with three tiers of encoder-decoder scales of 64, 256 and 512. The encoder we use is PointNet. We add the graph attention whose point feature increased by its attention weight after encoding. The folding decoder uses the resulting feature to learn a better surface through folding.

Figure 4. Encoder-decoder module.

3.7. Loss Function

The loss function's main goal is to zoom out the distance between similar points while zooming in the distance between dissimilar classes. It is directly applicable to the properties of the final aggregation.

In the loss function, we use contrastive loss and triplet loss. Chopra [30] propose contrastive loss, which maps rich relational information into low dimensional space. It is widely utilized in portraits of people's faces. The computation does not necessitate a lot of measurement data; instead, it relies on the nearby relationship. Triplet loss [31] is frequently used to distinguish between quite similar conditions.

3.7.1. Contrastive Loss

On the subject of contrastive loss, we have made some improvements. It is better suitable for the graph features. It was classified into two categories: loss in the same category and loss in a different category. The difference between each point and the cluster center is the same-category loss. The cluster center and the above points are both in the same category; the greater the influence, the lesser the value. It is worth noting that the point cloud registration task has two inputs. The goal is to create a transformation equation that will allow the two input features to be registered.

The loss calculation also has two inputs. P_i represents the point belonging to category m in the feature i . P_j is same as P_i . $|\overline{P_i}|$ means the opposite point of P_i , which be not in category n . As shown in Formula (9), L_{c_in} adjust points of the same category. As shown in Formula (10), L_{c_out} adjust points of the different categories. n represents each category, and C represents the set of clusters. $m(f_i)$ indicates the mean distance between each cluster center in feature i . $D(x, y)$ represents the distance between x and y . The latter $\sum_{k \in p_i} D(f_i, f_k)$ represents the sum of the distance between any point k and the cluster center. It should be noticed that neither the clustering center nor the point inside the class is contained within a feature vector during the calculating procedure. The clustering center of F_i is calculated with the same category point of F_j . The main reason why the point loss in F_i is not calculated is that we have performed some work in graph cluster. As a result, the distance inside the same feature is ignored in the next calculation.

$$L_{c\_in} = \sum_n \sum_{i \in P_i} \left(\frac{\sum_{k \in p_j} D(f_i, f_k)}{|P_j|} - m(f_i) \right) + \sum_n \sum_{i \in P_j} \left(\frac{\sum_{k \in p_i} D(f_j, f_k)}{|P_i|} - m(f_j) \right) \quad (9)$$

The distance between the present cluster center and other points in various categories is described by the different category losses. m_n is a maximum threshold calculated by all cluster centers. $\sum_{k \in \overline{P_j}} D(f_i, f_k)$ represents distances of all non-homogeneous points.

$$L_{c\_out} = \sum_n \left(\frac{\left[m_n - \sum_{k \in \overline{P_j}} D(f_i, f_k) \right]_+^2}{|P_j|} + \frac{\left[m_n - \sum_{k \in \overline{P_i}} D(f_j, f_k) \right]_+^2}{|P_i|} \right) \quad (10)$$

3.7.2. Triplet Loss

We improve the triplet loss to fit our strategy, aiming at the unique clustering methodology. Triad loss is defined as a decrease in the distance between positive connections while an increase in the distance between negative connections. It is an improvement to the contrastive loss. A set triple (anchor, positive, negative) is used as the input to triple loss. It will automatically generate embeddings. The aim is also to distinguish between positive and negative features in the distance. The problem we face is that positive features are farther away than negative ones. The issue is that positive characteristics are more distant than negative ones. We chose hard link triplet loss as our method. The average distance difference between positive and negative embeddings is calculated. Most of the negative

connections are still biased to the far end. Because the distance is so great, the parameter m is employed as a backup. The positive feature considers edge pairs that belong to the same class, while the negative feature considers point pairs that belong to a different class.

$$L_{t_i} = \max \sum_n^C \left(\frac{\sum_{k \in P_i} D(f_j, f_k)}{|P_i|} - \frac{\sum_{k \in \bar{P}_i} D(f_i, f_k)}{|\bar{P}_i|} + m, 0 \right) \quad (11)$$

$$L_{t_j} = \max \sum_n^C \left(\frac{\sum_{k \in P_j} D(f_i, f_k)}{|P_j|} - \frac{\sum_{k \in \bar{P}_j} D(f_j, f_k)}{|\bar{P}_j|} + m, 0 \right) \quad (12)$$

As shown in Formulas (11) and (12), $\sum_{k \in P_i} D(f_j, f_k)$ represents the point-to-point distance between the point in feature f_j and every homogeneous point in feature f_i . $\sum_{k \in \bar{P}_i} D(f_i, f_k)$ represents the point-to-point distance between the point in feature f_i and every non-homogeneous point in feature f_j . They are divided by the mean $|P_i|$ or $|P_j|$. The result is usually a negative number plus an equilibrium variable m to calculate the loss.

4. Experiment

Our network is verified on indoor data sets and outdoor data sets. The accuracy of result is better than many other approaches. Our method speeds up and increases the accuracy of calculations. We also update the cluster's contrastive loss and triple loss.

4.1. Datasets and Parameter for Learning

We use three datasets. The first is the indoor dataset 3D Match [9]. The second is outdoor dataset KITTI [32]. The third is dataset ETH [33]. The 3D Match dataset is reconstructed from RGB-D [3]. To train local patch descriptors, it extracted local 3D blocks. It learns the 3D data registration relationship. There are 62 groups in the registration scene, which is a 3D point cloud of the surface. There are 54 training sets and 8 validation sets. The data set in 3D Match is directly used for registration training. KITTI is a dataset that contains real-world traffic data. It plays an important role in the field of autonomous driving. We use the visual odometry dataset. It consists of 22 sequences, 11 of which have true trajectories and 11 of which do not. Only 11 sequences with genuine trajectories are used. There are three types of sets: training, validation, and test. To establish the relevant 3D data pair for registration, we employ FCGF (Fully Convolutional Geometric Features) [19], updating KITTI. In the real data pair, we also remove a lot of noise pairs. Finally, 1358 pairs of training sets, 180 pairs of validation sets, and 555 pairs of test sets are used in this study. ETH is a dataset to test point cloud registration algorithms in specific environments and conditions. We use the stairs sets structured with large variations in scanned volumes.

To train the network, we use the PyTorch frame. The learning rate is initially set to 0.1. The exponential learning rate (LR) is set to 0.9. The batch size in this experiment is set to two. The point cloud coordinate information and related attributes are used as input. This network can be used in a variety of ways, including just inputting point cloud coordinates or expanding features using color and normal lines. During training, we discovered that RGB color information helped to improve the training. The effect of normal lines is not substantial because graph clustering has performed geometric analysis on point clouds.

4.2. Evaluation

Different evaluation metrics are used by Dataset 3DMatch and KITTI. 3DMatch uses feature matching and registration matching to calculate the recall rate. KITTI makes use of relative translation errors and relative rotation errors.

4.2.1. Feature Match Recall

The fraction of point cloud fragments accurately restored by calculating the expected rotation transformation matrix is known as feature matching recall [2]. The Formula (13) for calculating feature match recall is presented below.

$$Re_F = \frac{1}{M} \sum_{s=1}^M \mathbb{I} \left(\left[\frac{1}{|P_s|} \sum_{i,j \in P_s} \mathbb{I}((X_i - TX_j) < \tau_1) \right] > \tau_2 \right) \quad (13)$$

M represents the collection of all fragments. P_s means the effective relation pair of the fragment s . X_i and X_j are coordinates in P_s . T is conversion equation. We calculate the coordinate difference in each relationship pair through the conversion equation. The value of τ_1 is 0.1 m, which is the threshold of internal distance. The value of τ_2 is 0.05, which is the threshold of internal percentage.

4.2.2. Registration Match Recall

The quality of the correctly recovered point cloud pieces is reflected in registration match recall [34]. The absolute distance between the coordinate difference and ground truth is used to calculate it. The Formula (14) for the computation is presented below.

$$Re_R = \sqrt{\frac{1}{P} \sum_{(x,y) \in P} \|T_{i,j}x - y\|^2} \quad (14)$$

P represents the set of ground truth registration pairs $\{i, j\}$. x and y are the coordinates from ground truth pairs. We calculated the difference between ground truth pairs through the transformation matrix T . It accurately evaluates the network.

4.2.3. Relative Translation Error and Relative Rotation Error

RANSAC (random sample consensus) is used to estimate the rigid transformation between point cloud pairs for outdoor datasets. It uses iterations to calculate suitable results by distinguishing between outsiders and insiders. Because it is possible to make mistakes when calculating the right probability with a single outsider and insider, numerous iterations are required.

The rigid transformation results of RANSAC are measured using relative translation error (RTE) and relative rotation error (RRE). As shown in Formula (15), $\hat{T}$ is the rigid transformation function after registration. As shown in Formula (16), $\hat{R}$ is the prediction rotation matrix, and R^* is the true rotation matrix. When both RTE and RRE are near to the set 2 m and 5° , the impact is optimal.

$$RTE = |\hat{T} - T^*| \quad (15)$$

$$RRE = \arccos \left(\left(\text{Tr}(\hat{R}^T R^*) - 1 \right) / 2 \right) \quad (16)$$

4.3. 3D Match Descriptor

We use the feature match recall and registration match recall on the 3D Match. Many of the most advanced approaches are compared to our method.

We compare six advanced methods, as given in Table 1. In six of the eight test categories, our technique achieves the present ideal state. Graph cluster greatly improves the quality of the input point cloud data and speeds up operation. On 3DMatch, our network had great average recall 2 percentages higher than FCGF. It has a more noticeable improvement than other methods. Figures 5 and 6 depict feature match recall performance under various internal parameters. The internal distance threshold τ_1 and the internal proportional threshold τ_2 are both controlled as independent variables.

Table 1. Comparison of feature matching recall rates between our model and other advanced models.

	SHOT	3DMatch	FPFH	PPF	PPF-FoldNet	FCGF	Ours
Red Kitchen	0.21	0.85	0.36	0.9	0.88	0.98	0.98
Home 1	0.37	0.78	0.56	0.58	0.59	0.92	0.93
Home 2	0.30	0.61	0.43	0.57	0.6	0.86	0.88
Hotel 1	0.28	0.79	0.29	0.75	0.77	0.94	0.93
Hotel 2	0.24	0.59	0.11	0.68	0.69	0.89	0.94
Hotel 3	0.42	0.58	0.12	0.88	0.89	0.96	0.97
Study Room	0.14	0.63	0.07	0.68	0.69	0.83	0.87
MIT Lab	0.06	0.51	0.09	0.62	0.63	0.87	0.86
Mean	0.18	0.67	0.40	0.71	0.73	0.90	0.92

Figure 5. The influence of different internal ratio thresholds on feature match recall. **Figure 6.** The influence of different internal distance thresholds on the feature match recall.

In graph cluster, we measure the effect of varied cluster radius on the feature match recall rate, as shown in Table 2. The cluster radius is separated into two categories: fixed and variable. Fixed radiuses of 20, 30, 40, and 50 have been set, while variable radiuses of 40 and 50 have been set. We do not contemplate utilizing a lower radius as a variable because the variable radius has been attenuated. When the radius is set to 40 from Table 2, the performance is better. The variable radius outperforms the fixed radius in terms of performance.

In terms of registration match recall, Table 3 compares our method to other advanced methods. In 3DMatch, our technique has the highest recall rate in six of the eight categories. The remaining two categories are only somewhat worse than the top results in other methods.

Table 2. The recall of different fixed radius and gradient radius in the Home1 category of 3DMatch.

	Fixed Radius (20 cm)	Fixed Radius (30 cm)	Fixed Radius (40 cm)	Fixed Radius (50 cm)	Variable Radius (40 cm)	Variable Radius (50 cm)
Feature match recall	0.90	0.91	0.92	0.90	0.93	0.91
Registration match recall	0.90	0.91	0.91	0.90	0.93	0.92

Table 3. Comparison of registration recall rates between our model and other advanced models.

	FPFH	3DMatch	PPF	FCGF	Ours
Red Kitchen	0.36	0.85	0.90	0.93	0.96
Home 1	0.56	0.78	0.58	0.91	0.93
Home 2	0.43	0.61	0.57	0.71	0.81
Hotel 1	0.29	0.79	0.75	0.91	0.88
Hotel 2	0.36	0.59	0.68	0.87	0.89
Hotel 3	0.61	0.58	0.88	0.69	0.71
Study Room	0.31	0.63	0.68	0.75	0.79
MIT Lab	0.31	0.51	0.62	0.80	0.77
Mean	0.40	0.67	0.71	0.82	0.84

The descriptor we created, as illustrated in Figure 7, express the input point cloud information more fully. The semantic information in the input and output is similar.

Figure 7. The visualization of 3DMatch dataset. The input is raw data. The output is descriptor after learning. (1) is point cloud sequence in group jan_1. (2) belongs to group redkitchen. (3) belongs to group studyroom2. (4) belongs to group scan3.

The registration experiment on the 3DMatch dataset is shown in Figure 8. Our approach + PointDSC [35] are used to register point clouds. Facing incomplete object structure and few similar objects in point cloud pairs, the descriptor can still extract precise structural features and complete good registration tasks.

Figure 8. The result on dataset 3DMatch for point cloud registration. (1) is part one. (2) is part two. (3) is an unregistered pair. (4) is a registered pair.

4.4. KITTI Descriptor

The visualization result of KITTI and ETH are shown in Figure 9. They retain accurate and believable geometric features. On KITTI, we compare relative translation error and relative rotation error with numerous advanced approaches. To manually search for descriptors, both the FPFH and 3DMatch algorithms use the ISS approach. We also compare the network when different radiuses are used. It has been discovered that the lower the relative translation and rotation errors, the better the performance. When the sampling voxel is smaller, the network performs better. The reason for this is that the formulation is more specific, and the calculation will take longer. Table 4 shows the evaluation results of two parameters. And our method has good accuracy compared with others.

Figure 9. (a) The visualization of KITTI dataset. (b) The visualization of ETH dataset.

Table 4. The result of KITTI dataset.

Voxel Size (cm)	RTE (cm)	RRE (cm)	Accuracy (%)
ISS+FPFH	32.5	1.08	58.95
ISS+3DMatch	28.3	0.79	89.12
3DFeat	25.9	0.57	95.97
FCGF	8.03	0.27	98.92
Ours-20	4.56	0.17	98.02
Ours-25	5.88	0.18	98.91
Ours-30	6.52	0.20	99.15
Ours-35	6.77	0.23	99.23
Ours-40	7.83	0.25	99.23

5. Conclusions

Using point, edge, and class information, we offer a new approach for capturing structural features of 3D point clouds. It uses graph clusters to provide dynamic region partition. It has a wider application and feature capturing capability than the fixed neighborhood methods. Experiments also demonstrate that the cluster technique can increase the accuracy of results by speeding up feature learning. At the same time, this cluster approach can be used as an auxiliary tool for feature learning in other 3D point cloud jobs. The structure of our encoder and decoder is PointNet and FoldingNet.

This method can increase the accuracy of point cloud descriptors. Precise geometric descriptors can capture the evident geometric linkages in numerous point clouds' underlying work. It can assist with operations such as correspondence estimation, feature matching, registration, target recognition, and instance segmentation.

This method still has development space. Firstly, due to equipment and time constraints, we only use one indoor dataset and one outdoor dataset. Our method should be evaluated on bigger and more realistic scenes. Secondly, the actual point cloud registration work must be implemented in accordance with the actual application scenes. Our next task is to increase the model's applicability for unknown datasets and make necessary model modifications.

Author Contributions: Funding acquisition, Q.S.; Resources, Q.S.; Supervision, Q.S.; Writing—original draft, Y.R.; Writing—review & editing, W.L. and X.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 61375075), Funded by Science and Technology Project of Hebei Education Department (QN2018214).

Data Availability Statement: [3DMatch] Andy Zeng. 2016. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions; <https://3dmatch.cs.princeton.edu/>, accessed on 17 January 2022; [KITTI] Andreas Geiger.2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite; <http://www.cvlibs.net/datasets/kitti/index.php>, accessed on 17 January 2022; [ETH] F. Pomerleau.2012. Challenging data sets for point cloud registration algorithms; <https://projects.asl.ethz.ch/datasets/doku.php?id=laserregistration:laserregistration>, accessed on 17 January 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tombari, F.; Salti, S.; Stefano, L.D. Unique shape context for 3d data description. In Proceedings of the ACM Workshop on 3D Object Retrieval, Firenze, Italy, 25 October 2010.
2. Deng, H.; Birdal, T.; Ilic, S. PPFNet: Global context aware local features for robust 3D point matching. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 195–205.
3. Deng, H.; Birdal, T.; Ilic, S. PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *Computer Vision—ECCV 2018, Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 620–638.
4. Johnson, A.E.; Hebert, M. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1999**, *21*, 433–449. [\[CrossRef\]](#)

5. Khoury, M.; Zhou, Q.Y.; Koltun, V. Learning compact geometric features. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
6. Salti, S.; Tombari, F.; Stefano, L.D. SHOT: Unique signatures of histograms for surface and texture description. *Comput. Vis. Image Underst.* **2014**, *125*, 251–264. [[CrossRef](#)]
7. Rusu, R.B.; Blodow, N.; Marton, Z.C.; Beetz, M. Aligning point cloud views using persistent feature histograms. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Acropolis Convention Center, Nice, France, 22–26 September 2008.
8. Rusu, R.B.; Blodow, N.; Beetz, M. Fast Point Feature Histograms (FPFH) for 3D registration. In Proceedings of the IEEE International Conference on Robotics & Automation, Kobe, Japan, 12–17 May 2009.
9. Zeng, A.; Song, S.; Niener, M.; Fisher, M.D.; Xiao, J. 3DMatch: Learning the matching of local 3D geometry in range scans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
10. Gojcic, Z.; Zhou, C.; Wegner, J.D.; Wieser, A. The perfect match: 3D point cloud matching with smoothed densities. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 5540–5549.
11. Charles, R.; Su, H.; Mo, K.; Guibas, L. PointNet: Deep learning on point sets for 3D classification and segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 77–85.
12. Yang, Y.; Feng, C.; Shen, Y.; Tian, D. FoldingNet: Point cloud auto-encoder via deep grid deformation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 206–215.
13. Guo, Y.; Sohel, F.; Bennamoun, M.; Lu, M.; Wan, J. TriSI: A distinctive local surface descriptor for 3D modeling and object recognition. In Proceedings of the 8th International Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), Barcelona, Spain, 21–24 February 2013.
14. Frome, A.; Huber, D.; Kolluri, R.; Bülow, T. Recognizing objects in range data using regional point descriptors. In Proceedings of the Computer Vision—ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, 11–14 May 2004; pp. 224–237.
15. Yu, Z. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Kyoto, Japan, 27 September–4 October 2009.
16. Guo, Y.; Sohel, F.; Bennamoun, M.; Lu, M.; Wan, J. Rotational projection statistics for 3D local surface description and object recognition. *Int. J. Comput. Vis.* **2013**, *105*, 63–86. [[CrossRef](#)]
17. Prakhya, S.M.; Lin, J.; Chandrasekhar, V.; Lin, W.; Liu, B. 3DHoPD: A fast low dimensional 3D descriptor. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1472–1479. [[CrossRef](#)]
18. Yew, Z.J.; Lee, G. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In Proceedings of the Computer Vision—ECCV 2018, 15th European Conference, Munich, Germany, 8–14 September 2018.
19. Choy, C.; Park, J.; Koltun, V. Fully convolutional geometric features. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 8957–8965.
20. Song, K.; Yang, G.; Zhang, G.; Wang, Q.; Xu, C.; Liu, J.; Liu, W.; Shi, C.; Wang, Y.; Yu, X.; et al. Deep learning prediction of incoming rainfalls: An operational service for the city of Beijing China. In Proceedings of the 2019 International Conference on Data Mining Workshops (ICDMW), Beijing, China, 8–11 November 2019; pp. 180–185.
21. Ding, C.H.Q.; He, X.; Zha, H.; Gu, M.; Simon, H.D. A min-max cut algorithm for graph partitioning and data clustering. In Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, CA, USA, 29 November–2 December 2001; pp. 107–114.
22. Xu, X.; Yuruk, N.; Feng, Z.; Schweiger, T. SCAN: A structural clustering algorithm for networks. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 12–15 August 2007; pp. 824–833.
23. Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 888–905. [[CrossRef](#)]
24. Shiokawa, H.; Fujiwara, Y.; Onizuka, M. SCAN++: Efficient algorithm for finding clusters, hubs and outliers on largescale graphs. *VLDB Endow.* **2015**, *8*, 1178–1189. [[CrossRef](#)]
25. Aaron, C.; Newman, M.E.J.; Cristopher, M. Finding community structure in very large networks. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **2004**, *70*, 264–277.
26. Guimerà, R.; Amaral, L. Functional cartography of complex metabolic networks. *Nature* **2005**, *23*, 22–231. [[CrossRef](#)] [[PubMed](#)]
27. Newman, M.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **2004**, *69*, 26113. [[CrossRef](#)] [[PubMed](#)]
28. Jiang, P.; Singh, M. SPICi: A fast clustering algorithm for large biological networks. *Bioinformatics* **2010**, *26*, 1105–1111. [[CrossRef](#)] [[PubMed](#)]
29. Lim, S.; Ryu, S.; Kwon, S.; Jung, K.; Lee, J.-G. LinkSCAN *: Overlapping community detection using the link-space transformation. In Proceedings of the 2014 IEEE 30th International Conference on Data Engineering (ICDE), Chicago, IL, USA, 31 March–4 April 2014; pp. 292–303.
30. Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 17–22 June 2006; pp. 1735–1742.

31. Schroff, F.; Kalenichenko, D.; Philbin, J. FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 815–823.
32. Geiger, A. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
33. Pomerleau, F.; Liu, M.; Colas, F.; Siegwart, R. Challenging data sets for point cloud registration algorithms. *Int. J. Robot. Res.* **2012**, *31*, 1705–1711. [[CrossRef](#)]
34. Choi, S.; Zhou, Q.-Y.; Koltun, V. Robust reconstruction of indoor scenes. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 5556–5565.
35. Bai, X.; Luo, Z.; Zhou, L.; Chen, H.; Li, L.; Hu, Z.; Fu, H.; Tai, C.-L. PointDSC: Robust point cloud registration using deep spatial consistency. In Proceedings of the 2021 Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021.