

Received December 7, 2021, accepted January 14, 2022, date of publication January 25, 2022, date of current version February 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3146247

Self-Supervised Segmentation for Terracotta Warrior Point Cloud (EGG-Net)

YAO HU¹, GUOHUA GENG¹, KANG LI¹, BAO GUO¹, AND PENGBO ZHOU²

¹School of Information Science and Technology, Northwest University, Xi'an, Shaanxi 710127, China

²School of Arts and Communication, Beijing Normal University, Beijing 100875, China

Corresponding author: Guohua Geng (ghgeng@nwu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61731015, in part by the National Key Research and Development Program of China under Grant 2020YFC1523301, Grant 2019YFC1521103, and Grant 2020YFC1523303, in part by the Key Research and Development Program of Shaanxi Province under Grant 2019ZDLSF07-02 and Grant 2019ZDLGY10-01, and in part by the Major Research and Development Project of Qinghai under Grant 2020-SF-142.

ABSTRACT At present, our team focuses on cultural relics restoration and fragment splicing research. In the research process of terracotta warrior splicing, we find that the existing calibrated fragment data is relatively small, which is not enough for related research. Therefore, we need to calibrate and segment different parts of the intact terracotta warrior data and extract some data we need to use in the future. However, at present, we are short of human resources. If we want to carry out manual calibration, it will take much time, bringing trouble to our future work. Therefore, we hope to design a method to automatically calibrate the terracotta warrior dataset with a small amount of calibrated data. The existing 3D neural network research mainly focuses on supervised classification, segmentation, and unsupervised reconstruction. We cannot find enough schemes to refer to, and the existing methods do not perform well on our terracotta warrior dataset. Therefore, in this article, we propose EGG-Net to solve this problem. EGG-Net is an end-to-end self-supervised model, and it consists of two modules. The first module is an encoder based on dynamic graph and edge convolution. We can extract point cloud features with this module. The second module, called segmenter, is based on multi-layer perceptron, adding labels to points and segmenting the point cloud. After the neural network, we add point refinement operation to the pipeline. Point refinement can adjust the cluster label estimated by the neural network with superpoint, which can optimize the loss function and help us train the neural network. Our EGG-Net can back-propagate with the refinement operation. We evaluated EGG-Net on the terracotta warrior data and ShapeNet Part by measuring the accuracy and the latency. The experiment result shows that our EGG-Net outperforms the state-of-the-art methods.

INDEX TERMS Point cloud, self-supervised learning, convolution neural network, terracotta warrior.

I. LIST OF ABBREVIATIONS

Abbreviations	Full Form.
CNN	Convolutional Neural Network
KNN	K Nearest Neighbors.
STN	Spatial Transformer Networks.
MLP	Multi-layer perceptron.
ReLU	Rectified Linear Unit.
SRG	Seed Region Growing.
DG-CNN	Dynamic Graph Convolutional Neural Network.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Jiang¹.

II. INTRODUCTION

Nowadays, our team focuses on cultural relics restoration and fragment splicing research. In the terracotta warrior fragment study, the main problem is the lack of calibrated fragment data, and it is not easy to conduct further research without enough calibrated data. However, we are currently short of human resources. If we want to calibrate the terracotta warrior dataset manually, it will take much time, which will bring trouble to our future work. Therefore, we hope to design a method to achieve automatic calibration on the terracotta warrior dataset with a small amount of calibrated data.

In order to improve the efficiency of part labeling on terracotta warrior models, we propose a self-supervised method called EGG-Net for the terracotta warrior dataset, which is based on a convolutional neural network. Our end-to-end

model can automatically segment an intact terracotta warrior into different parts: hands, heads, feet, and others, with only a tiny amount of segmentation results. Our EGG-Net can significantly improve efficiency and accuracy compared with the traditional manual and state-of-the-art methods.

Our terracotta warrior dataset is saved in the form of OBJ, a format created by wavefront technologies. OBJ is an open data format, which other 3D graphics application providers widely use. OBJ is a simple data format, which only represents 3D geometry, such as the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, faces, and texture vertices. By default, vertices are stored in counter-clockwise order, so there is no need to declare face normals explicitly. OBJ coordinates have no units, but OBJ files can contain scale information in the form that humans can read.

As to which data format we choose to research, we compare different data formats. Recently, many pieces of research have focused on how to voxelize the point cloud to make them evenly distributed in regular 3D space and then implement 3D-CNN on them. However, voxelization brings high space and time complexity. Besides, there may be quantization errors in the process of voxelization, which would result in low accuracy. Compared with other data formats, the point cloud is a data structure suitable for the 3D scene calculation of terracotta warrior data. At last, we choose to segment terracotta warrior data in the form of point clouds (see samples in Fig. 1).

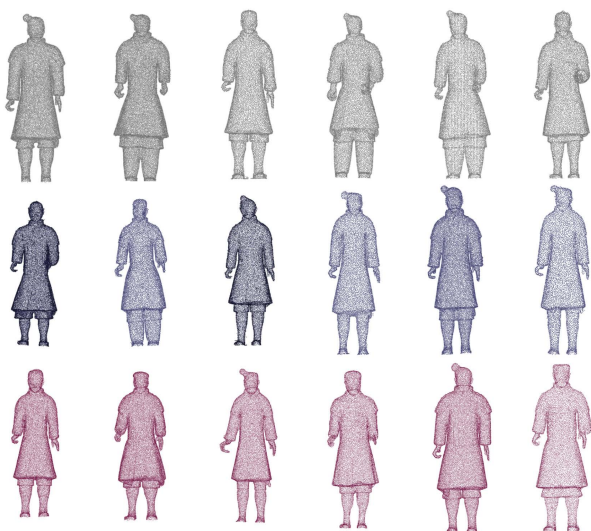


FIGURE 1. Simplified terracotta warrior point clouds.

We think adding an annotation to different parts of terracotta warrior data contains two steps. The first step is to conduct segmentation on the 3D models. The second step is to add labels to different parts. So we can regard this problem as a segmentation problem. Our terracotta warriors data is in the form of $\{x_n \in R^p\}_{n=1}^N$, where R^p is the feature space, x_n means the features of one point, such as XYZ coordinates and

normal vector. N is the number of points in one terracotta warrior 3D object.

Our goal is to design a function $f : R^p \rightarrow L$, where L means the segmentation mapping labels and $\{c_n \in L\}_{n=1}^N$, so c_n is label of each point after the segmentation. Different from our previous work SRG-Net, only few $\{c_n\}$ is fixed and the mapping function f is trainable and the other $\{c_n\}$ changes according to f .

In order to solve the self-supervised segmentation problem, we can split the problem into two parts. Firstly, we want to design an algorithm to predict optimal L we need to design a network to extract the features and use the features to segment the point cloud. Secondly, we need to design an appropriate loss calculating model to evaluate the predicted segmentation results and train the network.

In designing neural network, we find that auto-encoder can help extract the global features of the point cloud from the neural network. In addition, we also find that the dynamic graph can learn the local features well. Therefore, Inspired by these two methods, our design of EGG-Net can learn the features of terracotta warrior data better. In Section IV, we will describe the design of the network in detail.

In designing loss, we propose a method to evaluate the segmentation results. We think that a good point cloud segmentation should work like human beings. Firstly, the points with similar semantic features are more likely to be classified as the same kind of points. In 2D images, points with similar color and texture are generally considered spatially continuous; In 3D space, we think those points with similar normal vectors, color, and texture will be considered space continuous. In addition, the Euclidean distance between the points with the same label should not be very long. To sum up, we think that an excellent segmentation result of $\{C_N\}$ has the following two characteristics:

- Points with similar spatial features are desired to be given the same label.
- The Euclidean distance between spatially continuous points should not be quite long.

Inspired by [1], we combine the segmentation result predicted by EGG-Net with the superpoint in the refining process, in which the superpoint meets the above two requirements. Then we calculate the loss by combining it with the prediction segmentation result predicted by the neural network. Please refer to Section IV-B for detail.

In Section V, we compare EGG-Net with other methods and show the superiority of our method in visualization and quantification.

To sum up, the critical contributions of our work are summarized as follows:

- Inspired by dynamic graph and auto-encoder structure, we propose our EGG-Net to learn local and global features with lower latency and higher accuracy.
- We propose a new loss model suitable for the 3D point cloud self-supervised segmentation to obtain more accurate results.

- Our end-to-end model can not only achieve good results on terracotta warrior data. We also evaluate our model on the ShapeNet Part dataset and achieve quite good results.

III. RELATED WORK

Segmentation is typical in the 2D image and 3D point cloud processing. In image processing, segmentation completes a task assigning labels to all pixels in an image and clustering them with their features. Similarly, point cloud segmentation assigns labels to all points in the point cloud. The expected result is that points with similar characteristics are given the same label.

In image segmentation, K-means is a classical segmentation method in 2D and 3D. It divides N observations into K clusters with the nearest mean, popular in data mining. The graph-based method is another popular method, such as prim and Kruskal [2], which realizes simple greedy decisions in segmentation. The methods above focus on global rather than local differential features, so they can not obtain satisfactory results in complex contexts. Among self-supervised deep learning methods, there are many learning features using the generative methods, such as [3]–[5]. They follow the model of neuroscience, where each neuron represents a specific semantic meaning. Meanwhile, CNN is widely used in supervised and unsupervised image segmentation. For example, in [1], Kanazaki combines the superpixel [6] method and CNN and employs superpixel for backpropagation to tune the unsupervised segmentation results. Besides, [7] uses a spatial continuity loss as an alternative to settle the limitation of the former work [6], whose method is also quite valuable for 3D point cloud feature learning. In addition, in the field of segmentation and clustering, [8] and [9] are worthy of attention.

In the field of 3D point cloud segmentation, the state-of-the-art 2D image method is not suitable for directly using point cloud. The 3D point cloud segmentation method needs to understand each point's global features and geometric details. We can classify 3D point cloud segmentation problems into semantic segmentation, instance segmentation, and object segmentation. Semantic segmentation focuses on scene-level segmentation instances. Instance segmentation emphasizes object-level segmentation, and object segmentation focuses on partial-level segmentation.

As to semantic segmentation, semantic segmentation aims at separating a point cloud into several parts with the semantic meaning of each point. There are four main semantic segmentation paradigms: projection-based methods, discretization-based methods, point-based methods, and hybrid methods. Projection-based methods always project a 3D point cloud to 2D images, such as multi-view [10], [11], spherical [12], [13]. Discretization-based methods usually project a point cloud into a discrete representation, such as volumetric [14] and sparse permutohedral lattices [15], [16]. Instead of learning a single feature on 3D scans, several methods are trying to learn different parts from 3D scans, such as [15]–[17].

The point-based network can directly learn features on a point cloud and separate them into several parts. Point clouds are irregular, unordered, and unstructured. PointNet [18] can directly learn features from the point cloud and retain the point cloud permutation invariance with a symmetric function like maximum function and summation function. PointNet can learn point-wise features with the combination of several MLP layers and a max-pooling layer. PointNet is a pioneer that directly learns on the point cloud. A series of point-based networks has been proposed based on PointNet. However, PointNet can only learn features on each point instead of the local structure. So PointNet++ is presented to get local structure from the neighborhood with a hierarchy network [19]. PointSIFT [20] is proposed to encode orientation and reach scale awareness. Instead of using K-means to cluster and KNN to generate neighborhoods like the grouping method PointNet++, PointWeb [21] is proposed to get the relations between all the points constructed in a local fully-connected web. As to convolution-based method. RS-CNN takes a local point cloud subset as its input and maps the low-level relation to the high-level relation to learn the feature better. PointConv [22] uses the existing algorithm, using a Monte Carlo estimation to define the convolution. PointCNN [23] uses $\chi - conv$ transformation to convert the point cloud into a latent and canonical order. As to point convolution methods, Parametric Continuous Convolutional Neural Network(PCCN) [17] is proposed based on parametric continuous convolution layers, whose kernel function is parameterized by MLPs and spans the continuous vector space. Graph-based methods can better learn the features like shapes and geometric structures in point clouds. Graph Attention Convolution(GAC) [24] can learn several relevant features from local neighborhoods by dynamically assigning attention weights to points in different neighborhoods and feature channels. Dynamic Graph CNN(DG-CNN) [25] constructs several dynamic graphs in the neighborhood and concatenates the local and global features to extract better features and update each graph after each layer of the network dynamically. FoldingNet uses the auto-encoder structure to encode the point cloud $N \times 3$ to 1×512 and decode it to $M \times 3$ with the aid of chamfer loss to construct the auto-encoder network.

Part segmentation is more complex than semantic and instance segmentation because there are significant geometric differences between points with the same label, and the number of parts with the same semantic meaning may differ. Wang and Lu [26] propose VoxSegNet to achieve promising part segmentation results on 3D voxelized data, which presents a Spatial Dense Extraction(SDE) module to extract multi-scale features from volumetric data. Synchronized Spectral CNN (SyncSpecCNN) [27] is proposed to achieve fine-grained part segmentation on irregularity and non-isomorphic shape graphs with convolution. Reference [28] is proposed to segment unorganized noisy point clouds automatically by extracting clusters of points on the Gaussian sphere. Reference [29] uses three shape indexes: the smoothness indicator,

shape index, and flatness index based on a fuzzy parameterization. [30] presents a segmentation method for conventional engineering objects based on local estimation of various geometric features. Branched AutoEncoder network (BAE-NET) [31] is proposed to perform unsupervised and weakly-supervised 3D shape co-segmentation. Each branch of the network can learn features from a specific part shape for a particular part shape with representation based on the auto-encoder structure.

In conclusion, our work is mainly inspired by the five papers in the table 1.

TABLE 1. Significant referred papers for our work.

Significant Referred Papers
1. FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation
2. Dynamic Graph CNN for Learning on Point Clouds
3. Unsupervised Segmentation for Terracotta Warrior with Seed-Region-Growing CNN (SRG-Net)
4. Unsupervised Image Segmentation by Backpropagation
5. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

IV. PROPOSED METHOD

In this paper, our input data for the terracotta warrior is in the form of 3D point clouds (see samples in Fig. 1). Point cloud data is represented in the form of 3D points $\{P_i | i = 1, 2, 3 \dots n\}$, where each point is a vector R^n containing coordinates x, y, z and other features like normal, color. Our method contains three steps:

- 1) If the point cloud only has three-dimensional coordinates x, y, z data, we need to estimate the normal vector value with xyz .
- 2) We use our pointwise CNN called EGG-Net to perform self-supervised segmentation of point clouds.
- 3) We design a refinement process to calculate loss and use it for back-propagation.

There are many effective normal vector estimation methods, such as [32] using integral images for efficient boundary and covariance estimation, [33]–[36] Use neural network to estimate. In our method, we tend to use the simplest method [37] because this method has lower time complexity and good accuracy.

A. EGG-Net

Inspired by dynamic graph and auto-encoder, we propose our EGG-Net. Unlike the classical graph CNN, our graph layer is dynamic and auto-updated at every layer of the network. Compared with the methods that only focus on the relationship between points, we also propose an auto-encoder structure to re-express the features of the whole point cloud, aiming at learning the whole structure of the point cloud and learning from a small number of samples. The structure of our network is shown in Figure 2. It consists of two parts. The first part is an encoder that generates features from the dynamic graph and the whole point cloud, and the second

part is a decoder segmentation network. We can also call it segmenter.

Next, we will explain the symbols used in the paper. We denote the point cloud as S . We use lower-case letters to represent vectors, such as x , and the upper-case letter to represent matrix, such as A . We call a matrix $m \times n$ if it has m rows and n columns. In addition, the terracotta warrior point cloud data is N points with 6 features x, y, z, N_x, N_y, N_z (xyz coordinates and normal values). We also denote each point as x , so $X = \{x_1, x_2, x_3, \dots, x_n\} \subseteq R^6$.

1) ENCODER ARCHITECTURE

The EGG-Net encoder follows a similar design of [38], the structure of EGG-Net is shown in Fig. 2. Compared with [38], our encoder concatenate several multi-layer perceptrons (MLP) and several dynamic graph-based max-pooling layers. The dynamic graphs are constructed by applying KNN on point clouds. Different from our previous work [39], we removed the STN (spatial transformer networks) [40] module because we found this module improved the latency greatly but do not have a great impact on the accuracy of the experiment 2.

For the input point cloud, we compute three dynamic graphs and get graph features, respectively. In graph feature extracting process, we adopt the edge convolution in [4] to compute the graph feature of each layer, which uses an asymmetric edge function in Eq. (1):

$$f_{ij} = h(x_i, x_i - x_j) \quad (1)$$

where it combines the coordinates of neighborhood center x_i with the subtraction of neighborhood point and the center point coordinates $x_i - x_j$ to get local and global information of neighborhood. Then we define our operation in Eq. (2):

$$g_{ij} = \Theta(\mu \cdot (x_i - x_j) + \omega \cdot x_i) \quad (2)$$

where μ and ω are parameters and Θ is a ReLU function. Eq. (2) is implemented as a shared MLP with leaky ReLU. Then we define our max-pooling operation in Eq. (3):

$$x_i = \max_{j \in N(i)} g_{ij} \quad (3)$$

where $N(i)$ means neighborhood of point i .

The graph feature extraction layer computes the bottleneck. The structure is shown in Fig. 2. First, we compute the covariance 3×3 matrix for every point and vectorize it to 1×9 . Then the $n \times 3$ matrix of point coordinates is concatenated with the $n \times 9$ covariance matrix into a $n \times 12$ matrix. Then we put the matrix into a 3-layer perceptron. Then we feed the output of the perceptron to two subsequent graph layers. In each layer, max-pooling is added to the neighbor of each node. At last, we apply a 3-layer perceptron to the former output and get the final output. The whole process of the graph feature extraction layer is summarized in Eq. (4):

$$Y = I_{max}(X) K \quad (4)$$

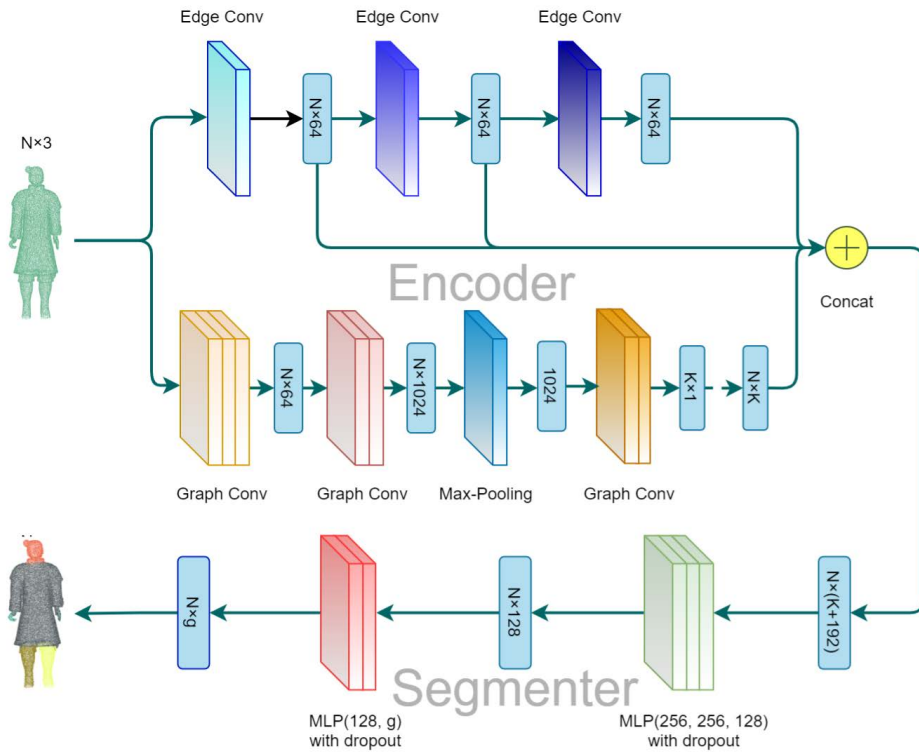


FIGURE 2. Network structure. The upper module in encoder represents the edge convolution, and the following module represents the graph convolution. The tensors of the upper and lower modules will be concatenated, and be input into the segmenter to get the final segmentation result.

In Eq. (4), X is the input matrix to the graph layer and K is a feature mapping matrix. $I_{max}(X)$ can be represented in Eq. 5:

$$(I_{max}(X))_{ij} = \Theta(\max_{k \in N(i)} x_{kj}) \quad (5)$$

where Θ is a ReLU function and $N(i)$ is the neighborhood of point i . The max-pooling operation in Eq. (5) can get local feature based on the graph structure. So the graph feature extraction layer can not only get local neighborhood features, but also global features. The advantages of combining the edge convolution and the graph convolution are shown as follows:

- 1) Edge conv can embed the relationship between points into high-dimensional feature space, which makes it easy for the following multi-layer perceptron to extract features.
- 2) Graph conv can imbed features from multi-graph constructed by KNN to extract the local and global features of point cloud.
- 3) Edge conv focuses on imbedding the features between points, and graph conv focuses on imbedding the features of the neighborhood of points.

To summarize, edge conv can embed the relationship between points into high-dimensional feature space, making it easy for the following multi-layer perceptron to extract features. Graph conv can imbed features from multi-graph

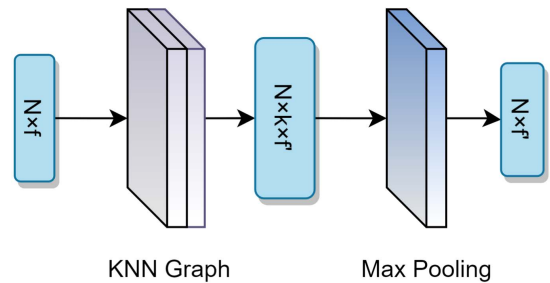


FIGURE 3. Edge convolution. The figure contains a KNN graph convolution mapping the point cloud to $N \times k \times f$, and then put into max-pooling to get the point cloud edge feature.

constructed by KNN to extract the local and global features of point cloud. Edge conv focuses on embedding the features between points, and graph conv focus on embedding the features of the neighborhood of points. We have compared different parts in the ablation study³ to show other factors impact on EGG-Net performance and accuracy results.

2) SEGMENTER ARCHITECTURE

Segmenter gets dynamic graph features and bottleneck as input and assign labels to each point to segment the whole point cloud. The structure of segmenter is shown in Fig. 2.

First, bottleneck is replicated N times in Eq. (6):

$$B' = \underbrace{B \cup B \cup \dots \cup B}_N \quad (6)$$

where N is the number of points in point cloud and B is the bottleneck. The output of replication is concatenated with dynamic features in Eq. (7):

$$C = B' \cup D_1 \cup D_2 \cup D_3 \quad (7)$$

where D_1, D_2, D_3 represent dynamic graph features.

At last, we feed the output of concatenation to a multi-layer-perceptron to segment the point cloud in Eq. (8).

$$D = \Theta(\Psi \cdot C + \Omega) \quad (8)$$

where Ψ and Ω represent parameters in the linear function, and Θ represents a ReLU function.

B. POINT REFINEMENT

In order to optimize the results of estimated by neural network and carry out back-propagation calculation, we designed point refinement to achieve better segmentation results.

In this section, we will describe how we train our network for self-supervised point segmentation. We can divide this problem into two sub-problems:

- 1) Estimate the cluster label using existing network parameters
- 2) Use the cluster label estimated by the neural network to train the neural network

As to the first sub-problem, we can use the auto-encoder network in our Section IV-A1 and Section IV-A2 to implement the forward process, and the second sub-problem is a backward process based on gradient descent. Next, we will describe the second sub-problem in detail.

We need to calculate the loss of network predicted labels and refine predicted labels in self-supervised segmentation. In the field of point cloud segmentation, we think that the points assigned the same label are spatially continuous (the clusters of image pixels should be spatially continuous in 2D images). Here, to better cluster the point cloud, we add additional restrictions on the points in the neighborhood. First, we use the region growing method to extract K' superpoints from the input point cloud. In this article, since our neural network can learn local and global features, we do not need to set K' very large in EGG-Net. In order to reduce the time complexity, we choose $K' = K$ to calculate the superpoint. The value of K is generally set to the number of segments of the few-shot samples. Then we set all the points in one superpoint with the same label. According to the cluster label estimated by the neural network, we select the most frequent cluster label c_{max} , where $|c_{max}|_{n \in S_k} \geq |c_n|_{n \in S_k}$ for all $c_n \in 1, \dots, q$. The cluster labels are replaced by c_{max} for $n \in S_k$, which are called refined predicted labels.

As to the seed-region-growing method used in superpoint calculating, we will describe in detail below:

Unlike 2D images, not all point cloud data has features such as color and normal. For example, our terracotta warrior

3D object does not have any color feature. Normal vectors can be calculated and predicted by point coordinates in IV. It is worth noting that there are many similarities and differences between the color feature in 2D and the normal feature in 3D. For the color feature in a 2D image, if pixels are semantically continuous, the color in the neighborhood generally does not change. For 3D point cloud normal features, compared with the color features in 2D images, the normal value of points in the neighborhood of the point cloud often differs. However, even though each point cloud neighborhood has different normal values, they usually do not change much unless they are not semantically continuous. We use seed-region-growing to cluster the point cloud to get the superpoints.

First, we implement KNN to the point cloud to get the nearest neighbors of each point. Then we initialize a random point as the start seed and add to the available points to start the algorithm. Then we choose the first seed from the available list to judge the points in its neighborhood. If the normal value and Euclidean distance are within the threshold we set, we think the two points are semantically continuous, and we can group two points into one cluster. The outline about the seed region growing method is given in Algorithm 1.

Algorithm 1: Seed Region Growing

Input: $P = \{p_n \in \mathbb{R}^3\}$ // x, y, z coordinates
 $N = \{n_n \in \mathbb{R}^3\}$ // normal value
 $E = \{e_n \in \mathbb{R}^m\}$ // euclidean distance
 $\Omega(\cdot)$ // nearest neighbour function
Output: $L = \{l_n \in \mathbb{R}^1\}$ // segmentation label of each point
Initialize: $S := \emptyset$ // set seed list to empty
 $A := \text{rand}\{1, 2, \dots, |P|\}$ // select random point to available point list as seed
for $t = 1$ to T **do**
 while $A \neq \emptyset$ **do**
 $a \leftarrow$ first item in A
 neighbours $\leftarrow \Omega(a)$ **for** neighbours $\neq \emptyset$ **do**
 neighbour \leftarrow rand(neighbours)
 if $E(a, \text{neighbour}) \leq e_{th} \wedge$
 $N(a, \text{neighbour}) \leq e_{th} \wedge \text{neighbour} \notin S$
 then
 append neighbour to S
 remove neighbour from neighbours
 else
 remove neighbour from neighbours
 add a to S

After obtaining the network predicted labels and refined predicted labels, we calculate their loss, and then we iterate this process T times to obtain the final prediction of cluster labels r_n .

Unlike general supervised learning, when the target labels are fixed, we need to perform batch normalization on each

dimension to get reasonable labels r_n . In parameter adjustment, we found that setting the learning rate to 0.1 and momentum to 0.9 can get the best results. For the comparison of different parameters, please check our Section V.

V. EXPERIMENTS

We do experiments on the terracotta warrior dataset and ShapeNet dataset. We implement the pipeline using PyTorch and Python3.7. All the results are based on experiments under RTX 2080 Ti and i9-9900K. The performances of each method in the experiment are evaluated by the accuracy (mIoU) and the latency.

A. EXPERIMENTS ON TERRACOTTA WARRIOR

We used Artec Eva [41] to collect 500 intact terracotta warrior models, and we took 400 of the 500 models as the training set and 100 as the validation set. Each model consists of about 2 million points, including xyz coordinates, vertical normals, triangle meshes, and RGB data. Before the experiment started, we eliminated the triangle meshes and RGB data of the original models and remained xyz coordinates and vertical normals. Moreover, we uniformly sample the above point clouds to 10,000 points thus as the experimental inputting. In reality, the terracotta warriors are generally unearthed in the form of limb fragments.

According to the description in [42], we divided the terracotta warrior 3D model into six parts: head, body, left hand, right hand, left leg, and right leg. In order to implement self-supervised learning better, we calibrated 5 ~ 10 terracotta warriors 3D models. Unlike the supervised problem, our self-supervised method solved two sub-problems: the existing network parameters to estimate the cluster label and the predicted cluster labels for the training network. The previous sub-problem was solved with section IV-A. The latter sub-problem was solved by section IV-B.

In order to show that EGG-Net could achieve better results in detail than other self-supervised methods, we used Pointnet, Pointnet2, DG-CNN, and PointHop++ to replace the neural network encoder in EGG-Net, and maintained the same structure as EGG-Net. In order to show that EGG-Net can achieve more correct results than unsupervised methods, we also compared EGG-Net with our previous unsupervised segmentation method SRG-Net, and the comparison methods of SRG-Net (SRG-DGCNN, SRG-PointNet2, and SRG-PointNet). We selected SGD as the optimizer and set lr to 0.005. We set the momentum parameter to 0.1 and set the number of iterations T to 500.

The visualization results are shown in Figure 2. We can find that EGG-Net can get more correct results than unsupervised methods (like SRG-Net, SRG-DGCNN, SRG-Pointnet2). In contrast, the unsupervised methods can not get correct results. Compared with self-supervised methods (PointNet-EGG, PointNet2-EGG, DG-EGG, and PointHop2-EGG), we can find that these methods can correctly segment the point cloud. However, in detail, EGG-Net has more accurate segmentation results. For example, EGG-Net can get a more

accurate result in the segmentation of the hands of the terracotta warrior 4.

The quantization results are shown in Table 2. Because the STN module is removed, compared with SRG-Net, EGG-Net reduces the latency by 27.3%, while accuracy improves about 8.2%. Compared with methods of the similar architecture (DG-EGG, PointNet2-EGG, PointHop2-EGG), our method also has quite good performance. Compared with DG-EGG, our solution reduces the latency by 40% and improves the accuracy by 9.8%. The accuracy of our method is also much better than PointNet2-EGG (17.1%) and PointHop2-EGG (12.7%), and the latency is about 65.8% and 63.1% of each method.

TABLE 2. Comparison of different methods on terracotta warrior dataset.

Method	Accuracy(%)	Latency(ms)
DG-EGG	81.34	31.19
PointHop2-EGG	79.01	25.26
Pointnet-EGG	75.44	13.36
Pointnet2-EGG	76.17	26.39
SRG-DGCNN	78.32	37.26
SRG-PointHop2	77.94	30.62
SRG-Pointnet	72.03	13.35
SRG-Pointnet2	70.55	24.10
EGG-Net	89.68	16.63

In summary, we can draw the following conclusions with the experiment results:

- 1) Compared with unsupervised methods (such as SRG-Net, SRG-DGCNN, SRG-PointNet, and SRG-PointHop2), our network can obtain more accurate results with less latency.
- 2) Compared with self-supervised methods with similar structures (such as PointNet-EGG, PointNet2-EGG, DG-EGG, and PointHop2-EGG), our EGG-Net can obtain more refined results.
- 3) In general, EGG-Net has obvious advantages in accuracy and latency on our terracotta warrior dataset.

B. EXPERIMENTS ON ShapeNet

In this section, we conducted experiments on the ShapeNet Part to evaluate the robustness of the EGG-Net method.

ShapeNet part is a consistent, large-scale 3D object dataset annotated with fine-grained, instance level, and hierarchical 3D part information. This dataset consists of 573585 part instances, including 26671 3D models of 24 object categories. The dataset acts as a catalyst for many tasks, such as shape analysis, dynamic 3D scene modeling, simulation, affordance analysis, etc. ShapeNet established three benchmark tasks for evaluating 3D part recognition: fine-grained semantic segmentation, hierarchical semantic segmentation, and instance segmentation. Among these tasks, ShapeNet Part is always used, for instance segmentation.

The quantitative results of our experiments are shown in Table 4. As shown in Table 4, EGG-Net outperformed all

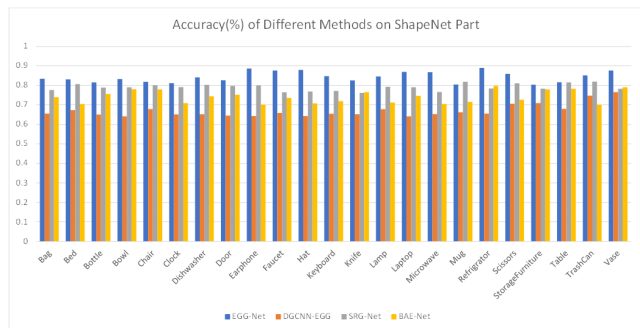


FIGURE 4. Per-category. Accuracy (%) of different methods on ShapeNet part. Each point represents mean IOU of different methods.

previous models. EGG-Net improved the overall accuracy of DGCNN-EGG by 8.2% and was even larger compared with PointNet2-EGG and PointNet-EGG. Significantly, our method outperformed DG-EGG on all categories, increasing 5% accuracy on the knife. Overall, our method achieved better accuracy on ShapeNet compared with other methods.

Some visualization results are shown in Fig. 5. As is shown in Fig. 5, EGG-Net achieves good results on bag, knife, motorbike, and achieves quite good results on airplane, earphone and laptop.



FIGURE 5. EGG-Net segmentation results on ShapeNet part.

C. ABLATION STUDY

In order to show the influence of different modules and epochs in our method, we conducted an ablation study on our

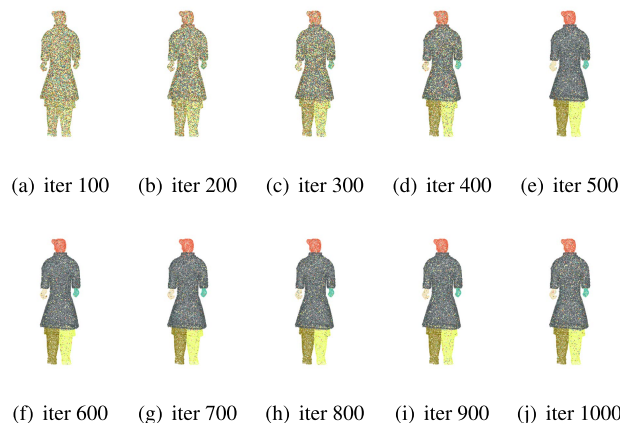


FIGURE 6. Results of different iterations in EGG-Net.

TABLE 3. Ablation study.

Method	mIoU(%)	OA(%)
without Graph Conv	65.32	68.11
without Edge Conv	73.24	74.83
without Refinement	71.74	75.67
EGG-Net	89.68	90.55

terracotta warrior dataset, which is described in Section IV-A and evaluated by overall accuracy and mIoU.





































1) THE INFLUENCES OF DIFFERENT MODULES

As shown in Table 3, the results of the pipeline without the graph convolution (row 1) show that the network is not working well in learning the topological features of the local neighborhood of the point cloud. The results of EGG-Net without edge convolution (row 2) demonstrate that our method without edge convolution will cause the network not to understand the relationship between points well. The third is a pipeline without refinement (row 3), and we calculated loss between labels of this epoch and the previous epoch. The results of the third method reveal that the pipeline cannot set tags reasonably based on point cloud content because the number of unique cluster labels should be adaptive to context. The results show that the refinement operation increases 15.6% on the accuracy of EGG-Net.

2) THE INFLUENCES OF DIFFERENT EPOCHS

To visualize the influence of different epochs, we set the number of epochs to 1000. The segmentation results of different iterations in one terracotta warrior model are shown in Fig. 6. We can find that when the number of iterations reaches 500, the visualization results do not change significantly, indicating that it is reasonable for us to set the number of iterations to 500.

TABLE 4. Results of different methods on terracotta warrior.

Method	005413	005420	005422	005423	005427	005455
Ground Truth						
Pointhop2-EGG						
Pointnet2-EGG						
Pointnet-EGG						
DG-EGG						
EGG-Net						

VI. DISCUSSION AND SIGNIFICANCE OF PROPOSED WORK

At present, our team focuses on the repair, fragment splicing, and other related work of terracotta warriors. In research, we found that we need enough data on part structures of terracotta warriors. However, the amount of existing data of part structures of terracotta warriors is tiny, which is not enough for further research and analysis. Therefore, we need to segment the existing data of intact terracotta warriors and extract part structure data used in the future. At present, the most commonly used method is to invite experts in archaeology to segment part data manually. However, relying only on handwork will consume much time and hinder future work.

Therefore, we hope to design a method that automatically extracts part structure data based on a small amount of calibrated data. The existing neural network research on the 3D point cloud mainly focuses on supervised classification, segmentation, and unsupervised reconstruction. There are few solutions for unsupervised and self-supervised segmentation. In order to solve the above problems, we propose a self-supervised method to segment the terracotta warrior point cloud automatically. The main contributions of our work are as follows:

- 1) Inspired by graph neural network and auto-encoder structure, we propose EGG-Net to learn local and global features with less latency and more accuracy.

- 2) We propose a novel loss model for self-supervised segmentation of 3D point cloud methods.
- 3) Our end-to-end model can not only achieve good results on terracotta warrior data. We also evaluated our model on ShapeNet Part dataset and achieved quite good results.

VII. CONCLUSION

This paper provides an end-to-end model called EGG-Net for self-supervised learning segmentation on terracotta point clouds. Our idea comes from the process of researching the terracotta warrior dataset. The existing calibrated data is insufficient for related research; however, we are currently short of human resources. It will cost much time to perform manual calibration, which would hinder future terracotta warrior restoration work. Therefore, we hope there will be a method that can achieve automatic calibration on many terracotta warrior 3D models with a small amount of calibrated data.

We designed EGG-Net, an end-to-end self-supervised model according to the existing problems. Our model contains three sub-modules. The first module is an encoder structure based on dynamic graphs and edge convolutions. We can extract features of our 3D point cloud with this structure. The second module is a segmenter based on a multi-layer perceptron. Finally, we designed a point refinement process. We calculate the superpoint with the seed region growing method and adjust the cluster labels calculated by the neural network to carry out back-propagation with this structure.

Finally, we evaluated our method on the terracotta warrior dataset and compare it with the latest and classical methods. The quantitative and visual results show that our EGG-Net has higher accuracy and lower latency. In addition, we also carried out experiments on ShapeNet Part and achieved good results, which shows that our method is robust on the public dataset. We also conducted an ablation study for different modules to show the rationality of the EGG-Net network structure by using different encoders and different refinement methods. Finally, we researched the number of iterations, which shows the rationality of our chosen parameters.

Our work still has some limitations. For example, deploying our model and using it is not so convenient. We will try our best to solve this problem in the future. We hope our work can be helpful to the research of terracotta warriors in archaeology and other point cloud work of other researchers.

REFERENCES

- [1] A. Kanazaki, "Unsupervised image segmentation by backpropagation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 1543–1547.
- [2] J. B. Kruskal, Jr., "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math. Soc.*, vol. 7, no. 1, pp. 48–50, Feb. 1956.
- [3] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 22, 2009, pp. 1096–1104.
- [4] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8595–8598.
- [5] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 609–616.
- [6] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2011.
- [7] W. Kim, A. Kanazaki, and M. Tanaka, "Unsupervised learning of image segmentation based on differentiable feature clustering," *IEEE Trans. Image Process.*, vol. 29, pp. 8055–8068, 2020.
- [8] C. L. Chowdhary and D. P. Acharjya, "Clustering algorithm in possibilistic exponential fuzzy C-mean segmenting medical images," *J. Biomimetics, Biomater. Biomed. Eng.*, vol. 30, pp. 12–23, Jan. 2017.
- [9] C. Chowdhary and D. Acharjya, *Segmentation Mammograms Using a Novel Intuitionistic Possibilistic Fuzzy C-Mean Clustering Algorithm*, Jan. 2018, pp. 75–82.
- [10] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3D semantic segmentation," in *Proc. Int. Conf. Comput. Anal. Images Patterns*. Cham, Switzerland: Springer, 2017, pp. 95–107.
- [11] A. Boulch, B. Le Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," *3DOR Eurographics*, vol. 2, p. 7, Apr. 2017.
- [12] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1887–1893.
- [13] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet ++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 4213–4220.
- [14] B. Graham, M. Engelcke, and L. V. D. Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9224–9232.
- [15] A. Dai and M. Nießner, "3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 452–468.
- [16] M. Jaritz, J. Gu, and H. Su, "Multi-view PointNet for 3D scene understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3995–4003.
- [17] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2589–2597.
- [18] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.
- [19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.
- [20] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation," 2018, *arXiv:1807.00652*.
- [21] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5565–5573.
- [22] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9621–9630.
- [23] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on X-transformed points," 2018, *arXiv:1801.07791*.
- [24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
- [25] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019.
- [26] Z. Wang and F. Lu, "VoxSegNet: Volumetric CNNs for semantic part segmentation of 3D shapes," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 9, pp. 2919–2930, Sep. 2020.
- [27] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2282–2290.

- [28] Y. Liu and Y. Xiong, "Automatic segmentation of unorganized noisy point clouds based on the Gaussian map," *Comput.-Aided Des.*, vol. 40, no. 5, pp. 576–594, 2008.
- [29] L. Di Angelo and P. Di Stefano, "Geometric segmentation of 3D scanned surfaces," *Comput.-Aided Des.*, vol. 62, pp. 44–56, May 2015.
- [30] P. Benkő and T. Várady, "Segmentation methods for smooth point regions of conventional engineering objects," *Comput.-Aided Des.*, vol. 36, no. 6, pp. 511–523, May 2004.
- [31] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "BAE-NET: Branched autoencoder for shape co-segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8490–8499.
- [32] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 2684–2689.
- [33] D. OuYang and H.-Y. Feng, "On the normal vector estimation for point cloud data from smooth surfaces," *Comput.-Aided Des.*, vol. 37, no. 10, pp. 1071–1079, Sep. 2005.
- [34] J. Zhou, H. Huang, B. Liu, and X. Liu, "Normal estimation for 3D point clouds via local plane constraint and multi-scale selection," *Comput.-Aided Des.*, vol. 129, Dec. 2020, Art. no. 102916.
- [35] Y. Wang, H.-Y. Feng, F.-É. Delorme, and S. Engin, "An adaptive normal estimation method for scanned point clouds with sharp features," *Comput.-Aided Des.*, vol. 45, no. 11, pp. 1333–1348, Nov. 2013.
- [36] R. Sharma, T. Schwandt, C. Kunert, S. Urban, and W. Broll, "Point cloud upsampling and normal estimation using deep learning for robust surface reconstruction," 2021, *arXiv:2102.13391*.
- [37] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.
- [38] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.
- [39] Y. Hu, G. Geng, K. Li, W. Zhou, X. Hao, and X. Cao, "Unsupervised segmentation for terracotta warrior with seed-region-growing CNN(SRG-Net)," in *Proc. 5th Int. Conf. Comput. Sci. Appl. Eng.*, 2020, pp. 1–6.
- [40] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," 2015, *arXiv:1506.02025*.
- [41] A. Europe. *3D Object Scanner Artec Eva | Best Structured-Light 3D Scanning Device*. [Online]. Available: <https://www.artec3d.com/portable-3d-scanners/artec-eva>
- [42] J. Dang, "The features of the terracotta warriors in the museum of emperor Qin," *Art Educ.*, to be published.

• • •