Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9477008, 15 pages
https://doi.org/10.1155/2022/9477008

Research Article

Rotation Estimation Based on Serial Network and Application in Cave Buddha Statues

Zixuan Guo, Qing Xie, Song Liu, and Xiaoyao Xie

Key Laboratory of Information and Computing Science Guizhou Province, Guizhou Normal University, Guiyang 550001, China

Correspondence should be addressed to Xiaoyao Xie; xyx@gznu.edu.cn

Received 8 July 2022; Revised 15 September 2022; Accepted 6 October 2022; Published 26 October 2022

Academic Editor: Jian Su

Rotation estimation is one of the crucial elements of pose estimation. Accurate pose estimation facilitates the observation of objects in the metaverse and is also the most fundamental part of automated archaeological mapping. Many studies have been conducted on 6D pose estimation, but no systematic study has only been conducted on 3D rotation estimation. We systematically study the rotation estimation problem for 3D point cloud models. Firstly, we discuss the representation of rotation and how it affects the training of the neural network. After that, the dataset is created. A new method of generating labels is also proposed, in which two points are used as labels, with the two points being able to recover the initial orientation. Finally, a new serial network is proposed, which can be used to improve the performance of rotational estimation systems. The research described in this paper has been successfully applied to the restoration of the Huangze Temple Grotto Buddha statue in Guangyuan, Sichuan Province, providing a reliable scheme for solving the rotation estimation problem with faster convergence and higher accuracy.

1. Introduction

In recent years, the metaverse has been an area of great interest to researchers. The integration of the virtual world with the natural world is a trend. Access to the metaverse requires immersive hardware devices such as virtual reality and augmented reality, as well as technical support related to immersive experiences. Furthermore, 3D objects need to present a realistic and natural, accurate, and stable posture in the metaverse, which involves the problem of posture estimation in computer vision and is an inevitable topic in the future regarding immersive experiences.

The field of technological archaeology has developed in recent years. Using 3D scanners, we can get more point cloud data of the cave statues. The archaeological reports' images are two-dimensional projections of the three-dimensional model in a fixed view. There are two ways of acquiring images, one is to acquire images manually, and the other is to acquire images automatically. The manual way to obtain an image is to manually rotate the statue to a certain angle in a professional point cloud processing software. The automatic method is to estimate the rotation angle of the point cloud and then automatically rotate the point cloud to a fixed perspective to generate a projection image. Generally, a front view is obtained first and based on the front view, two-dimensional views such as side, top, and profile views can also be obtained automatically. Images can be acquired in two ways: one is to acquire images manually, and the other is to acquire images automatically. Rotating the Buddha to the front is the most fundamental aspect of automatic archaeological mapping, and images of these views are an essential part of the archaeological report.

In archaeological reports, the size of a three-dimensional object in an image is determined by the scale and is not directly reflected in the image. The detail in the image is determined by the orientation of the three-dimensional

object, which affects the archaeologist's observation and analysis. The Buddha is a complex and irregular object in three-dimensional space. If data is missing and the Buddha needs to be readjusted to a frontal view, it can only be rotated to an approximate range by human intuition, which can lead to errors in the resulting side view and profile view, thus affecting the archaeologist's judgement.

Pose estimation is a significant subject in computer vision, which generally includes the estimation of translations and rotations, and is a six-degree-of-freedom estimation problem. Rotation estimation is one of the essential elements of pose estimation, which belongs to the three-degree-of-freedom estimation problem. Some researchers have studied the issue of continuity of rotation representations. However, there is a lack of research on the stability of various representations during training, and stability is one of the factors affecting the selection of rotation representations. On the other hand, in previous studies of rotation estimation, the concept of "Coarse to Fine" (C2F) has not been used to enhance the accuracy and convergence speed further.

In this paper, we propose a systematic scheme to estimate the amount of rotation of the point cloud to be measured to a fixed viewpoint using an end-to-end deep neural network. To summarize the contributions of this paper, there are three main points.

(i) We analyze the continuity and stability of the rotation representations

(ii) No dedicated dataset is available, so new datasets and labels must be generated

(iii) A serial network is proposed, and the model is trained using an online training method. The serial network can accelerate the convergence speed and increase the results' accuracy

2. Related Work

Rotation estimation is an important part of 6D pose estimation. Studies on pose estimation are numerous and cover many fields, such as human pose estimation [1–3], object pose estimation [4], SLAM [5], 3D reconstruction [6], and point cloud alignment [7, 8]. First, we elaborate on the work related to 6D pose estimation. We then explore work related to the representation of rotation in rotation estimation. Finally, we elaborate on the concept of iteration for the model used in this paper.

In this section, we review these three works and compare them with our approach with the aim of obtaining a class-level rotation estimation system with greater accuracy and convergence speed.

2.1. 6D Pose Estimation. The pose estimation problem is classified into an instance-level pose estimation problem and a category-level pose estimation problem, depending on whether an accurate 3D model of the object is known.

Instance-level pose estimation requires an accurate target point cloud against which the object to be estimated can be compared to obtain a relative pose. Instance-level pose estimation methods are mainly classified into two categories: traditional methods and deep learning-based methods. The traditional method is mainly based on a template approach, using images from different viewpoints to construct a discrete pose space about the object, and the images from unknown viewpoints are compared with the template to find the most similar state in the pose space, thus obtaining the pose of the current point cloud. [9–12] construct a template space by extracting feature points from a 2D image and then searching through the templates of numerous object poses to find the closest pose. The other category is based on deep learning methods. Deep learning methods have powerful representational capabilities, while eliminating the tedious manual step of design features, achieving better results than traditional methods. [13] is the first method that uses CNN to process the input RGB image to obtain the object pose. [14–16] fuse the color and depth information to extract features for more accurate results. In instance-level pose estimation, the difficulty of pose estimation is reduced because it is compared to an accurate model, but when the geometry, color, texture, and lighting change slightly, the point cloud to be estimated does not match well with the exact target model, resulting in poor accuracy.

Category-level pose estimation does not have an accurate 3D model of the target. Currently, the commonly used solution is to generate a continuous template space at the category level which is an abstract representation of the category-level pose information. The representation of the space can be of a concrete shape and size, as in the case of the NOCS map space [17], or it can be an abstract implicit space, as in the case of the CASS space [18] constructed using VAE [19]. Eventually, the pose of the object is estimated by mapping the instances to that space. Category-level pose estimation requires the construction of an intermediate template space, which can lead to complex neural network models. As it does not require the provision of an accurate target model, this greatly expands the application scenarios for pose estimation.

The problem we are addressing is that of category-level rotation estimation. In contrast to [17, 18], the method in this paper does not generate an intermediate template space, but maps the point cloud directly to the rotation representation space via PointNet [20] and MLP. We have also demonstrated the feasibility of the method experimentally.

2.2. Rotation Representations. The rotation has three degrees of freedom and various representations. The rotation representations can be divided into two categories according to whether they are continuous or not. One category is discontinuous rotation representations such as quaternions [21, 22], axis angle [23], Euler angles [24, 25], 6D vectors with GS constraints [26], and 9D vectors with SVD constraints [27]. The paper [9] demonstrates that for 3D rotations, all representations are discontinuous in the real Euclidean spaces of four or fewer dimensions and refer to some negative effects on the learning of neural networks. The study of the properties of rotation representations can contribute

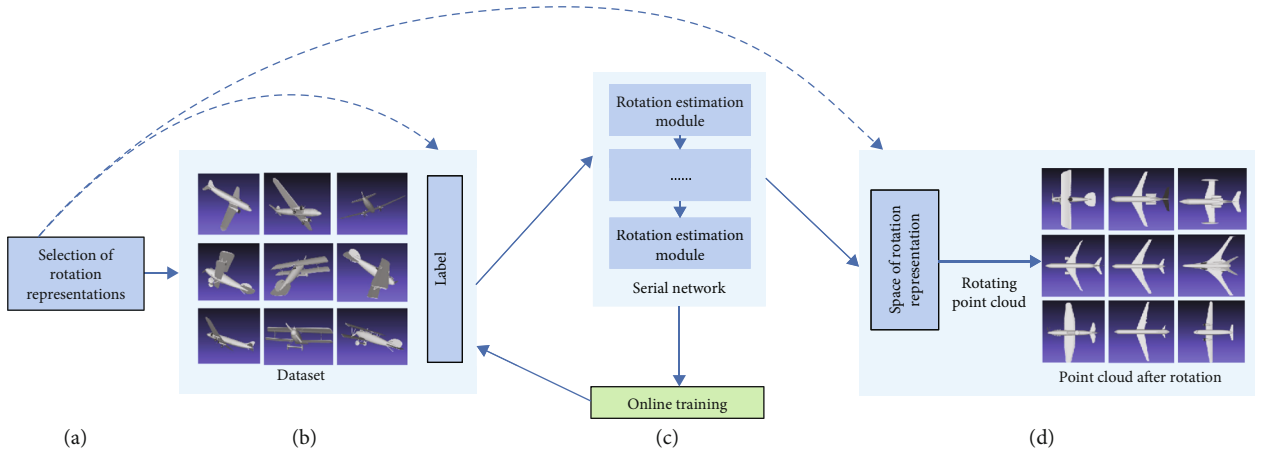


FIGURE 1: Pipeline of rotation estimation systems. The system is divided into four parts in total. First, a suitable rotation representation is selected for the actual situation (a). Then, the dataset and labels are generated (b) and the tandem network is trained using online training (c). Finally, the rotation angle that will rotate the model to be estimated to a fixed viewpoint is obtained (d).

to the selection of rotation representations in neural networks. Some of these properties have not been studied further, such as the lack of research and experiments related to training stability. Training stability is a factor that influences the selection of representations; the more stable the training, the less susceptible the training can be to noise in the dataset.

2.3. C2F. C2F (Coarse to Fine) is an idea of hierarchical iteration commonly used in neural network training, which is widely used. [28] applies the method to the field of image segmentation, [29] applies it to the classification of diabetic retinas, [30] applies it to the localisation of feature points on faces, and so on. The refine module used in DenseFusion [31] is also based on the C2F idea, which iteratively optimizes the 6D pose. The inputs to DenseFusion are RGB images and RGBD images, and the position and pose are optimized using two modules, coarse and fine, in a cyclic and iterative manner. The C2F iteratively utilizes the output of the model, adding parameters and calculations to the neural network, but this process allows the accuracy of the results to be further improved.

The serial network we propose is also formed based on the idea of Coarse to Fine. In contrast to DenseFusion, our input is point cloud data, and we use the same modules in series. The number of modules can be greater than two and only iteratively optimize 3D rotations without considering translations. Furthermore, we elaborate on the effectiveness of serial networks in terms of data distribution.

3. Overview of the Rotation Estimation System

We propose an end-to-end network that enables the rotation of the original point cloud to a fixed view without the target point cloud as a reference. The pipeline for this system is shown in Figure 1. The system consists of four main components.

(i) The first step is to select the appropriate rotation representations for the specific dataset. The different rotation representations generate different labels and map the point cloud to different rotation representation spaces

(ii) The dataset and labels are then generated. We use an online training approach, generating new datasets during the training process

(iii) In the next step, the rotation estimation network was used for training. The input to this network is a 3D point cloud coordinate, and the output is a rotation representation. The series network consists of several rotation estimation modules connected in series. Each rotation estimation module contains two parts, the feature extraction module and the rotation representation estimation module, which use PointNet and MLP, respectively, for the extraction of features

(iv) Finally, after mapping the point cloud into the rotation representation space, the point cloud can be rotated to a fixed viewpoint using the values of the resulting rotation representation

4. Rotation Estimation

In this section, three main components of the 3D rotation estimation system are discussed. In Section 4.1, we consider the issue of continuity and stability in rotational representations. The method for generating datasets is described in Section 4.2. In Section 4.3, a serial network and an online training method that we propose are presented.

4.1. Rotation Representation

4.1.1. Accuracy of Training Results. In a neural network, a particular constraint is applied to make the output strictly satisfy the following equation, and the output under this constraint is called the constrained representation, as

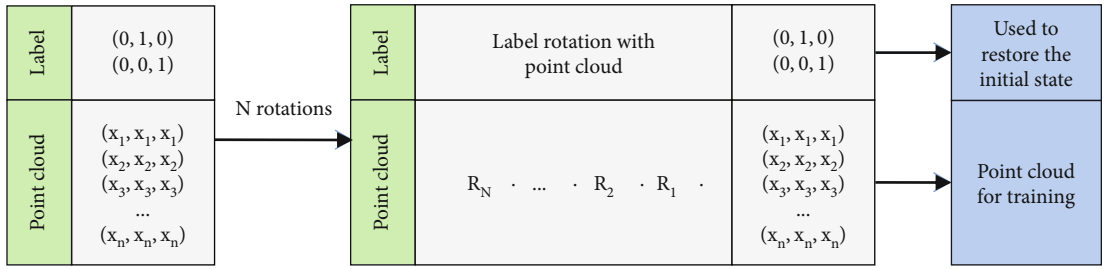


Figure 2: Storage structure and computation process of labels and point clouds.

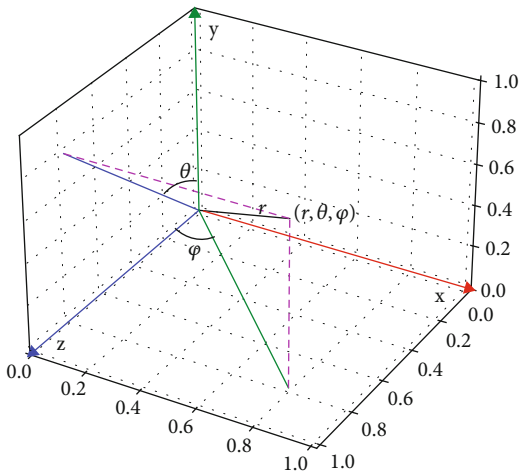


Figure 3: The rotation of the point around the "*xy*" axis.

Data: point cloud
Result: a rotation representation;
1 initialization;
2 set N as the number of re-generation datasets;
3 set S as the number of dataset size;
4 set e as the number of epochs;
5 $k \longleftarrow 0, m \longleftarrow 0$;
6 **while** $k < N$ or not converged **do**
7 generate a smaller dataset, the size of which is S;
8 **while** $m < e$ **do**
9 perform one epoch of training;
10 $m \longleftarrow m - 1$
11 **end**
12 empty the dataset;
13 $k \longleftarrow k - 1$
14 **end**
15 training completed.

Algorithm 1: Online training algorithms.

defined in papers [9, 10].

$$\left\{M \middle| MM^T = M^T M = I, \det(M) = 1\right\}. \tag{1}$$

By observing the experimental results, we draw the following two conclusions:

(i) The convergence accuracy of the representations with constraints is higher than that of the discontinuous representations. This phenomenon is caused for two reasons. First, the paper [26] argues that it is mainly because the discontinuous representations can negatively affect the learning of neural networks. Second, a neural network can easily learn the hidden laws with the constraint

(ii) Quaternions perform the worst accuracy and stability in discontinuous representations, while Euler angles and axis angle are very close. We hypothesize that the output dimensions of the neural network and the orthogonality of each dimension belong to solid prior knowledge that can help the neural network be trained. The dimensions of Euler angles and axial angles are equal to the number of freedom degrees of 3D rotation, and therefore, they have an advantage over quaternions

4.1.2. Stability of Training. We define N as the number of iteration rounds on each dataset and define the error of each iteration round as e_i and $\bar{e}$ as the average of the errors under this dataset. We can know the training smoothness by the variance, which is calculated as follows:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(e_i - \bar{e})^2. \tag{2}$$

The training method in this paper generates several small datasets, each of which can be calculated to obtain a variance, and a broken line about the training stability can be obtained by connecting all the variances. The discontinuous representations gradually decrease errors during training, and the training process is smooth. Constrained representations have a large error at the beginning of the training phase, and then, the error decreases rapidly. The training process is relatively jittery.

In conclusion, we can be guided by the following principles when selecting a representation.

(i) If the training data is noisy, the discontinuous continuous representation can be selected because the training process is more stable in discontinuous representations, and the error is lower at the beginning of training

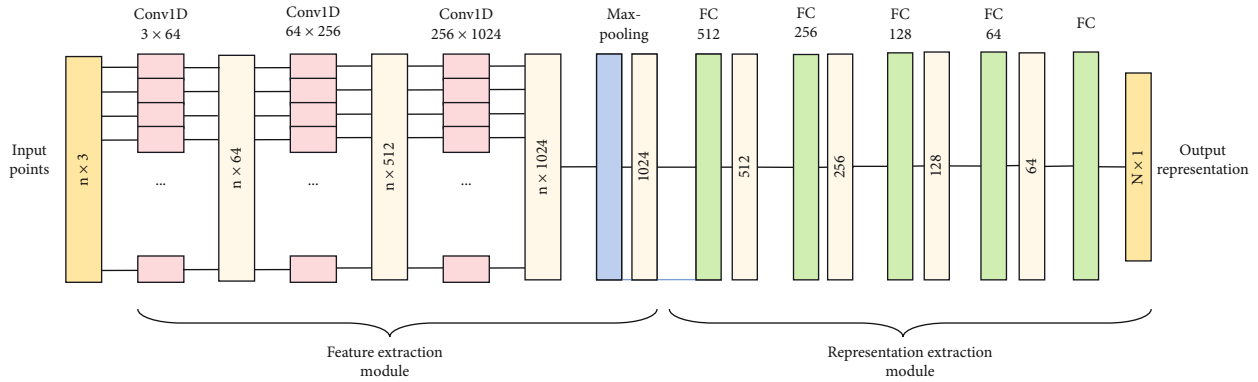


Figure 4: Neural network architecture for rotational estimation. n is the number of points in the point cloud, and N is the dimension of the representation.

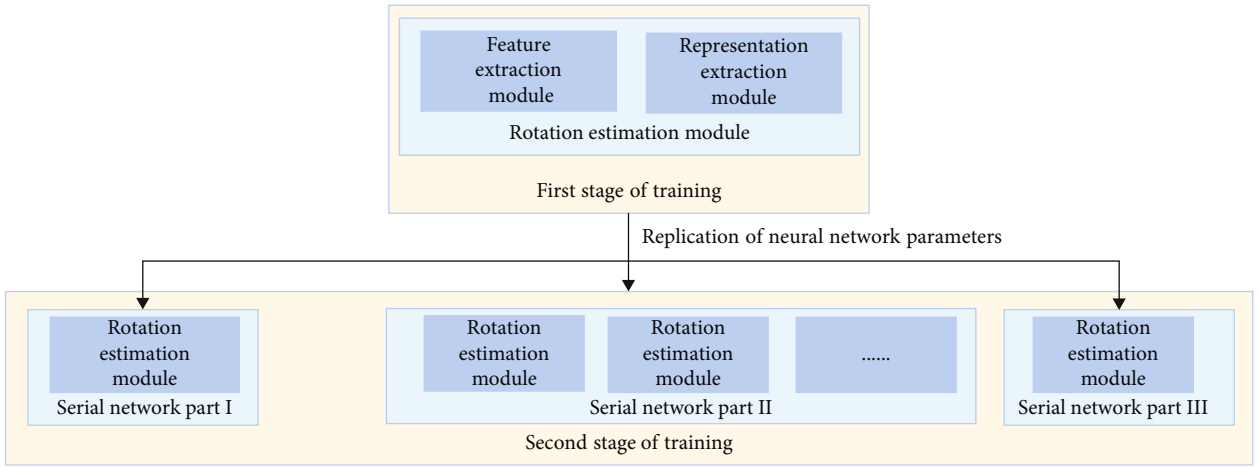


Figure 5: Two stages of training.

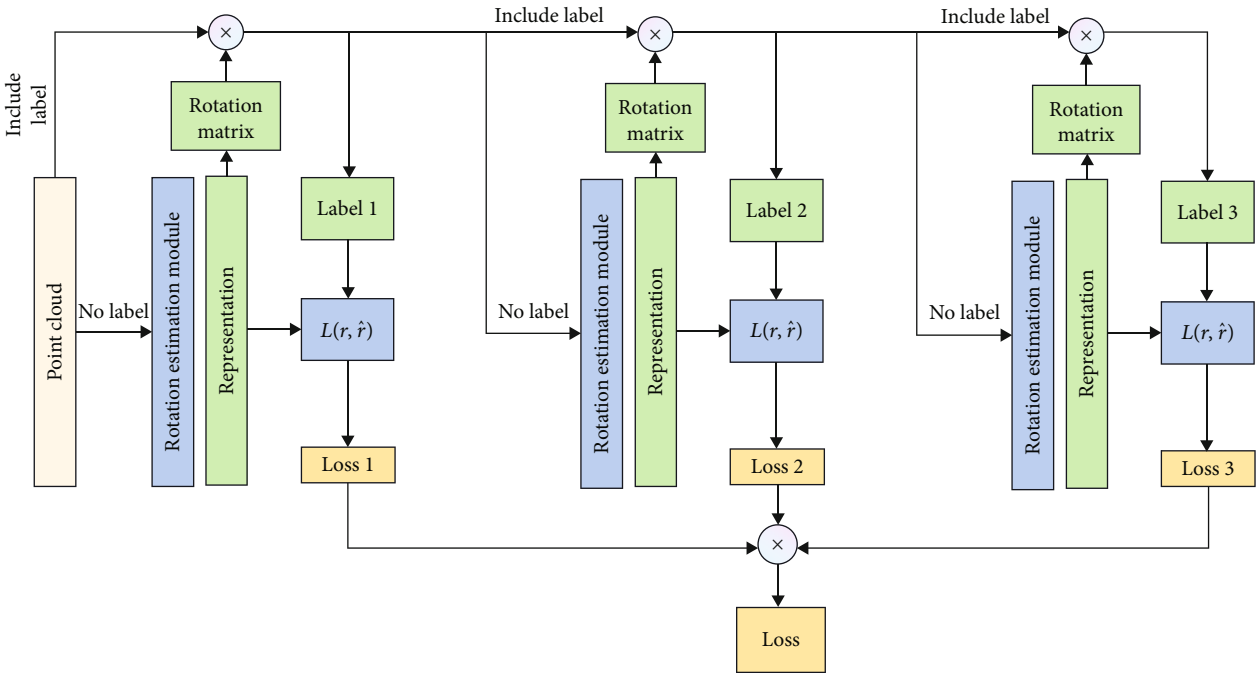


Figure 6: Three rotation estimation modules are used as an example to demonstrate the architecture of a serial network.

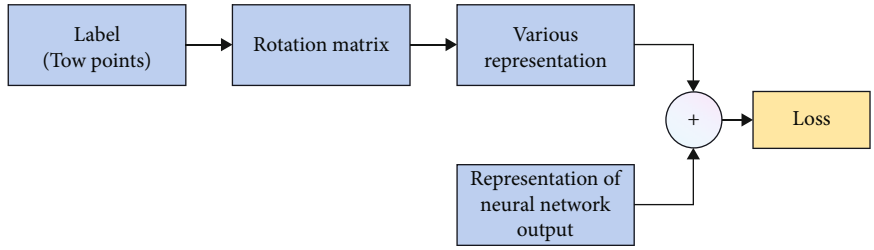


Figure 7: The process of calculating the loss function.

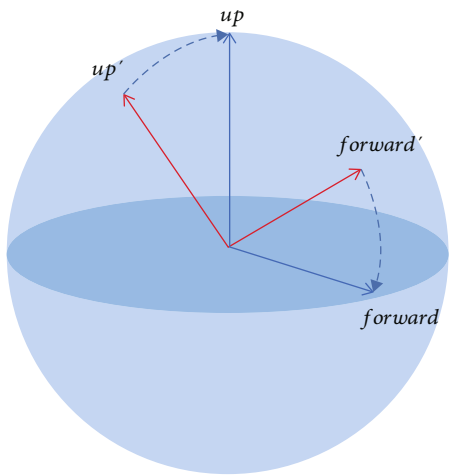


Figure 8: The geodetic error of orientation.

(ii) If the requirement for accuracy is very high and the data is relatively clean, the constrained continuous representation can be selected to have a higher accuracy of the final result

4.2. Label Generation and Dataset Creation

4.2.1. Label Generation. The label is a vector that restores the 3D model to its initial rotated state, and this vector can be an arbitrary rotation representation. However, keeping track of the label changes is tedious if a neural network is trained with multiple rotations on the data.

In this paper, we propose a method to recover the initial angle by two points, which combines the Euler angles with the unit sphere and records the rotation information of the point cloud with two points. Figure 2 shows the storage structure and calculation process of labels and point clouds. Note that the two points are rotated together with the point cloud but not as inputs to the neural network.

(1) The Connection between the Euler Angle and the Spherical Coordinate System.

(1) We assume that the point cloud is rotating in the direction of the "*yxy*" axis. First, select two points on the unit sphere as the orientation of the point cloud. The center of the sphere and the two points on the surface of the sphere form two vectors, and these two vectors are not colinear. For example, in this paper, we select up$(0, 1, 0)$ and forward$(0, 0, 1)$, two orthogonal vectors, to represent the model's orientation

(2) The two points rotate α degrees around the y axis, $\alpha \in [0, 2\pi]$. After rotation, the position of forward changes, and the position of up does not change

$$R_1(\alpha) = R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{3}$$

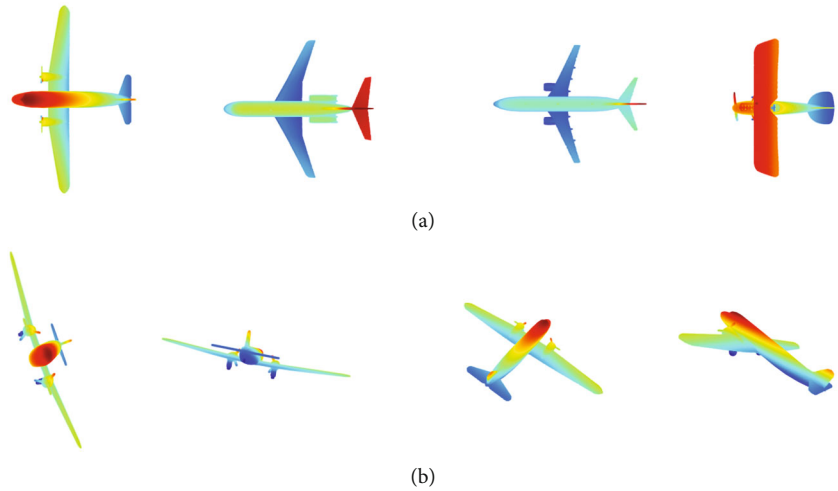
(3) The two points rotate about the x and y axes. After two rotations, the point up can be rotated to any position on the sphere. As shown in Figure 3, θ is the angle of rotation on the x axis, and ϕ is the angle of rotation on the y axis. A point $(0, r, 0)$ on the y axis is rotated to the point (r, θ, ϕ) under the representation of the spherical coordinate system. Then, the coordinate values of the points in the spherical coordinate system are converted to the coordinate values in the rectangular coordinate system. Given $r = 1$, $\theta \in [0, \pi]$, $\phi \in [0, 2\pi]$, the formula is as follows:

$$\begin{aligned} x &= \sin\theta\cos\varphi, \\ y &= \cos\theta, \\ z &= \sin\theta\sin\varphi \end{aligned} \tag{4}$$

The rotation of the point around the "xy" axis can be understood as the movement of the point on the sphere. The formula for the second rotation is as follows:

$$R_2(\theta, \varphi) = R_y(\varphi)R_x(\theta) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}. \tag{5}$$

(4) The final rotation matrix is shown as follows:



(a)

(b)

Figure 9: Visualization of airplane point cloud data. (a) Initial orientation of partial airplanes. (b) Partial sample of airplanes with different orientations.

$$R(\alpha,\theta,\varphi) = R_2(\theta,\varphi)R_1(\alpha) = \begin{bmatrix} \cos\alpha\cos\varphi - \sin\alpha\sin\theta\sin\varphi & \sin\theta\sin\varphi & \sin\alpha\sin\varphi + \sin\varphi\cos\alpha\cos\theta \\ \sin\alpha\sin\theta & \cos\theta & -\sin\theta\cos\alpha \\ -\sin\varphi\cos\alpha - \sin\alpha\cos\theta\cos\varphi & \sin\theta\cos\varphi & -\sin\alpha\sin\varphi + \cos\alpha\cos\theta\cos\varphi \end{bmatrix} \tag{6}$$

(2) Restoring the Initial Angle Method Using Two Points. Given up′ and forward′ are the points after rotation. The specific steps to restore the initial state are as follows:

(1) Use up × up′ to get the rotation axis, and then, use the inverse trigonometric function to get the angle between up and up′. Finally, using the axis-angle formula, rotate up′ to up. The formula is as follows:

$$\begin{cases} \vec{\omega}_1 = \text{up} \times \text{up}', \\ \theta_1 = \arccos\left(\dfrac{\text{up}\cdot\text{up}'}{\|\text{up}\|\cdot\|\text{up}'\|}\right), \\ R_1 = \theta_1\vec{\omega}_1 \end{cases} \tag{7}$$

(2) Forward′′ is rotated by R_1. The formula is as follows:

$$\text{forward}'' = R_1 \cdot \text{forward}' \tag{8}$$

The angle between forward and forward′′ can be obtained using the inverse trigonometric function. The y axis is the axis of rotation. The rotation matrix R_2 between forward and forward′′ can be obtained using the axis-angle formula.

$$\begin{cases} \vec{\omega}_2 = \text{up}\vec{\omega}_2 = \text{up}, \\ \theta_2 = \arccos\left(\dfrac{\text{forward}\cdot\text{forward}''}{\|\text{forward}\|\cdot\|\text{forward}''\|}\right), \\ R_2 = \theta_2\vec{\omega}_2 \end{cases} \tag{9}$$

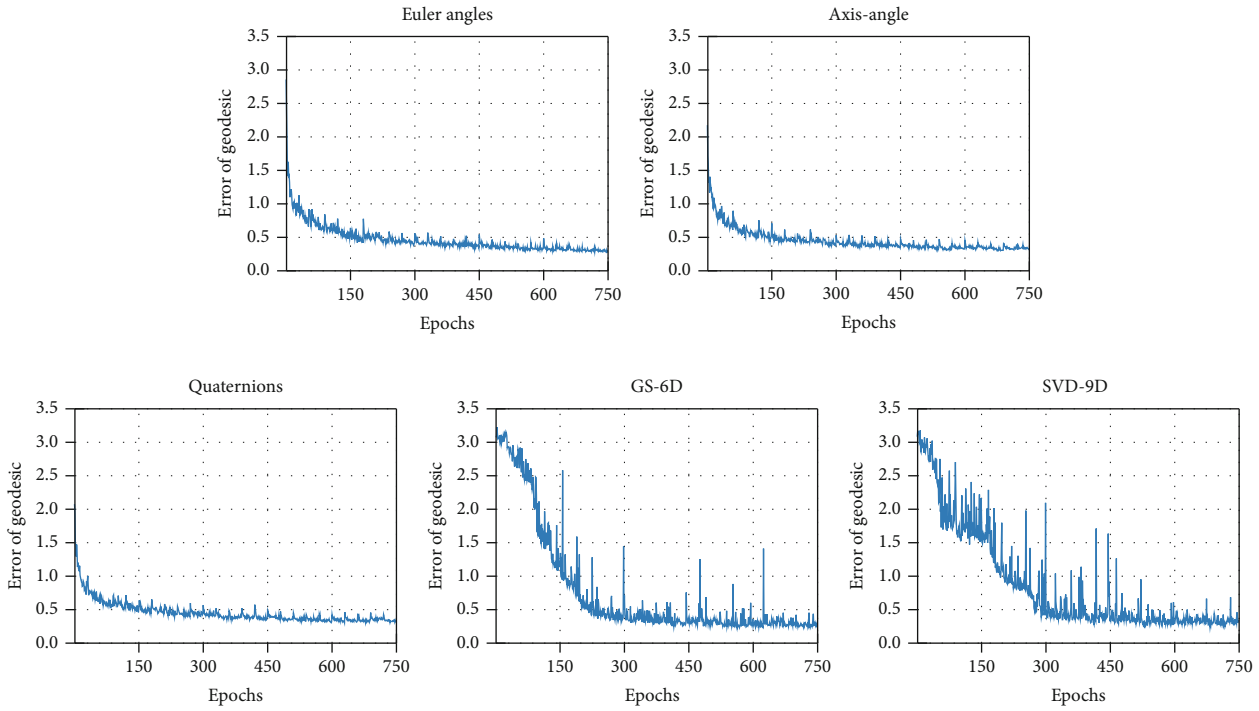
(3) The final matrix for recovering the initial angle is

$$R_{\text{recovery}} = R^{-1} = R_2 \cdot R_1 \tag{10}$$

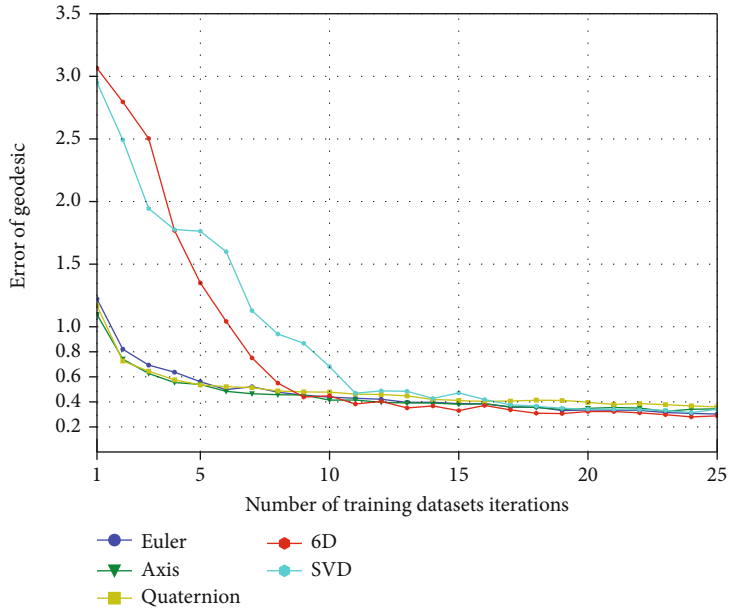
The approach described above is a two-point method to restore the initial orientation. When the label is rotated back to its original state, the points in the point cloud are correspondingly rotated back to their initial state.

4.2.2. Dataset Creation and Online Training Method. There is no specialized dataset for the rotation estimation problem, so samples with various orientations need to be generated based on the available dataset.

In general, uniform sample distribution and an adequate number of samples allow the neural network to achieve better results. It is necessary to ensure that the generated orientation samples are distributed uniformly. This means that the orientations cannot be clustered in a particular area. As the number of data increases, according to the law of large numbers, the orientation of the samples will naturally spread over all states, so there is no need to generate samples with different orientations deliberately. Furthermore, this paper adopts the training method of online learning, which can theoretically generate infinite data samples and thus ensure that the orientation will be uniformly distributed as the training continues to grow.

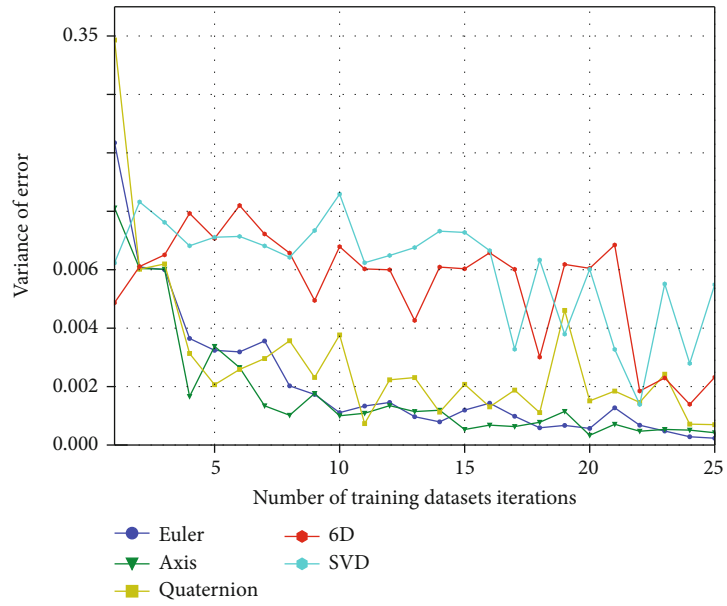


(a)



(b)

Figure 10: Continued.



(c)

Figure 10: Experimental results of the airplane. The dataset size is 5000, and 25 datasets are generated in total. Each dataset was trained for 30 iteration epochs. (a) The broken line plot generated by the epochs. (b) Broken line plot of the mean of the errors for all epochs within each dataset. (c) The variance of each dataset.

Table 1: Comparison of errors in different representations.

Representation	Minimum value (radian)	Error of the 25th round (radian)	Variance of the 25th round
Euler	0.27	0.3029	0.0002
Axis	0.3	0.3445	0.0004
Quaternion	0.303	0.3597	0.0007
GS-6D	0.227	0.2884	0.0023
SVD-9D	0.239	0.3408	0.0055

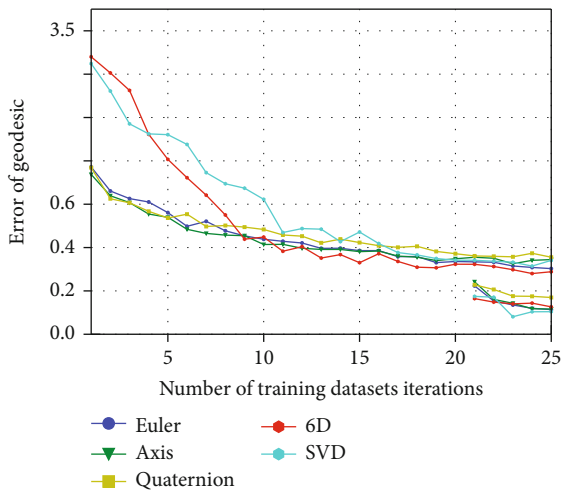


Figure 11: Comparison of the error of the single module with that of the serial network. The single module was trained for 25 rounds, and the tandem network was trained for 5 rounds.

Table 2: Comparison experiment of data from single module and serial network.

Representation	Minimum error in the 20th round (radians)	Single module 25th round error (radian)	Series network 25th round error (radian)	Error reduction (radian)
Euler	0.304	0.3029	0.1137	0.1892
Axis	0.3257	0.3445	0.1164	0.2281
Quaternion	0.3232	0.3554	0.1696	0.1858
6D	0.2465	0.2884	0.1264	0.161
SVD	0.2703	0.3408	0.104	0.2368

The sample generation method is as follows. First, a rotation matrix is generated randomly. Then, the randomly oriented sample data is generated by multiplying the rotation matrix with the labels and point clouds.

The method often used to generate datasets is to generate all samples at once and store them on the hard disk. With too few samples in the dataset, overfitting problems can easily occur. However, generating a large number of samples means taking up a lot of storage space. Furthermore, since the dataset does not change, the fit is probably problematic when the sample distribution is unbalanced.

The online training method is used in this paper to generate a new dataset during the training process to overcome the overfitting problem and avoid an unbalanced sample distribution as shown in Algorithm 1.

The samples in the dataset generated by this training method are different each time, so it is not easy to have overfitting. Also, it is easier to analyze whether the data

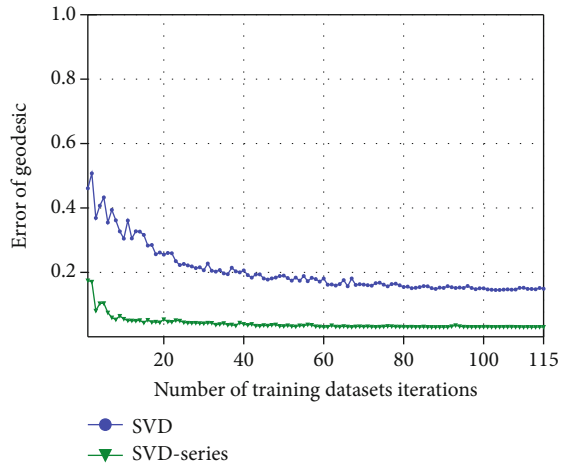


FIGURE 12: Both single module and serial network are trained using the SVD-9D representation, and the figure shows the experimental results after 115 rounds of training, which corresponds to 3450 epochs.

converges or not by observing the results at the scale of the dataset iteration.

4.3. Serial Network. The serial network is built on the basis of the rotation estimation module. Therefore, the rotation estimation module is first described in detail in Section 4.3.1, and then, the architecture of the tandem network is described in Section 4.3.2.

4.3.1. Architecture of the Rotation Estimation Module. The architecture of the rotation estimation module used for the experiments in this paper is shown in Figure 4 and is a complete neural network that can be trained independently. The model contains a feature extraction module and a rotation representation extraction module, which constitute the architecture of the rotation estimation model. The feature extraction is borrowed from PointNet by eliminating the T-Net from it because the T-Net can be considered as a rotation of the point cloud or feature, which will disturb the rotation feature of the original point cloud. The input of this module is (x, y, z), and the output is to obtain a 2048-dimensional feature vector. The MLP part uses five fully connected layers, which contain four hidden layers with outputs of 512, 256, 128, and 64 dimensional vectors, respectively, adding BN layers [32], while using the Leaky ReLU function, and the output layer is the dimension of the rotation representation.

4.3.2. Architecture of the Serial Network. When predicting the orientation of a model, a single-module rotation estimation network may not allow the model to learn effectively during the training phase due to the small dataset, unbalanced sample distribution, and symmetry of the model, which eventually leads to problems such as large errors in the regression results and unstable convergence results. It can be used to coarsely extract the rotation angle of a point cloud.

An approach to solving these problems is to call the model several times after completing the training. When the prediction results do not change much, we consider that the model has converged and then, the results estimated by each module are superimposed to obtain the final rotation angle.

However, the method mentioned above is not an end-to-end method, and the intermediate calculation results need to be saved, and additional hyperparameters are introduced, such as the number of iterations and angular errors. To avoid the above mentioned problems, we propose the serial network based on the idea of multiple iterations.

4.3.3. The Process of Serial Network Training. The process of training is shown in Figure 5. The training process has two stages:

(i) In the beginning, a single module is trained until the results converge or nearly converge

(ii) Then, the second stage of training is then performed. The parameters obtained from the training of the first-stage rotation estimation module are copied to the rotation estimation module in the serial model, and the parameters are fixed for the first and second parts of the serial network

Some details of the serial network are shown in Figure 6. In the serial network, the input data of each rotation estimation module is a point cloud and the output is a representation. After outputting a representation in the intermediate stage, it is necessary to convert it into a rotation matrix and apply it to the point cloud and labels to get the rotated data as the input data for the next module.

4.3.4. Loss Function and Error Evaluation Criteria. The flow of calculating the loss is shown in Figure 7. The loss function of the serial network is related to the number of rotation estimation modules. We define the representation obtained by the labels as r_{recover}, the representation output by each rotation estimation module as r_i, and the loss of the module as the $L2$ norm of r_i and r_{recover}. L is the total loss function, which is the sum of the loss functions of each module, with minimization of this loss function as the objective of optimization. Where D is the representation's dimensionality, N is the number of rotation modules, and the loss function is calculated as follows:

$$L(y, \hat{y}) = \sum_{N} \frac{1}{D} \sum_{D} \left\| y_D^N - \hat{y}_D^N \right\|. \tag{11}$$

Because each dimension in the different representations has a different physical meaning, the results of rotation estimation cannot be judged by L alone. In this study, the error is expressed as a geodesic distance on the unit sphere, which is used to quantify and compare the final error. The sphere's geodesic is the arc on the great circle intersected by the plane going through the sphere's center and surface. The radian value is the length of the arc on the unit circle. As previously described, two points can indicate orientation; then, the sum

Figure 13: Partial sample of Buddha statue.

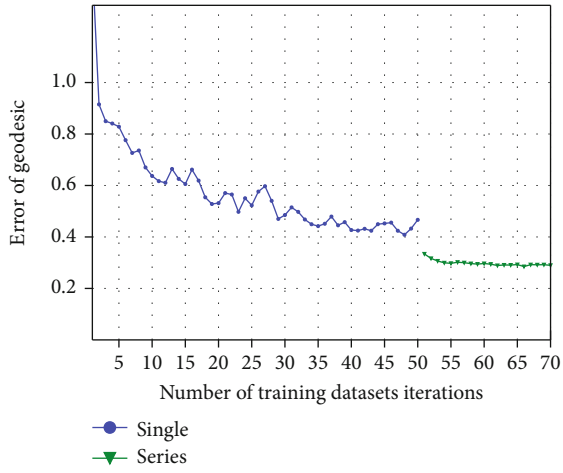


Figure 14: The figure shows the error of the Buddha data. After 50 rounds of training with the single module, 20 more rounds of training are performed using the serial network.

of the geodesic distances between the two points and the initial two points can indicate the difference in orientation. As shown in Figure 8, it is known that up′ and forward′ are the values of the current label, and up and forward are the initial points. Therefore, the calculation equation of the geodesic error is as follows:

$$\theta_{\text{error}} = \theta_{up\_\text{error}} + \theta_{\text{forward\_error}} = \arccos\left(\frac{\text{up}' \cdot \text{up}}{\|\text{up}'\| \cdot \|\text{up}\|}\right) + \arccos\left(\frac{\text{forward}' \cdot \text{forward}}{\|\text{forward}'\| \cdot \|\text{forward}\|}\right). \tag{12}$$

5. Experiment

In this section, we evaluate the convergence speed and stability of different rotation representations in experiments with the single rotation estimation module. To demonstrate the advantages of the serial network in terms of accuracy and convergence speed, we compare the serial network with the single module network in the experiment. Further, the neural network is trained over a long period of time to verify whether the online training method is subject to overfitting and the stability of the results. Finally, the serial network is applied to the Buddha statue dataset to verify its feasibility.

All experiments were implemented using PyTorch 1.8 with an i7 6700 CPU and a gtx1080 GPU. The single-module network and the serial network in the experiment uniformly use the same architecture as the rotation estimation module. The architecture of the rotation estimation module can be found in Section 4.3.1. During online training, each generated dataset is of size 5000, and the number of iteration epochs for each dataset is 30. We define a training round as all training after generating a new dataset.

5.1. Dataset Generation. The Aligned ModelNet40 dataset contains 40 categories [33], of which the airplane data is used in this paper. As the base data for sample generation, 310 different airplane models are selected, and the initial orientation is shown in Figure 9(a). Firstly, add two points up $(0, 1, 0)$ and forward$(0, 0, 1)$ to the airplane data, which are the labels. Then, the point cloud is randomly rotated. The partially rotated point cloud is shown in Figure 9(b).

5.2. Representation Experiments. The experiment was designed to evaluate the speed of convergence and the stability of the results of different rotation representations during training. Five rotation representations were selected for comparison, which are Euler angles, axis angle, quaternions, GS-constrained 6D vector, and SVD-constrained 9D vector.

We use the dataset provided in Section 5.1. The architecture of the neural network is the single rotation estimation module in Section 4.1. The dataset was regenerated 25 times during the online training process, and each dataset was trained for 30 epochs, for a total of 750 epochs. We trained the neural network utilizing the Adam optimizer, with a learning rate of 10^{-3}, while the learning rate was decayed using an exponential decay strategy with a decay rate of 0.998. The results of the experiment are shown in Figure 10.

From the data in Table 1, we can draw the following conclusions:

(i) After 750 iteration epochs of training, the minimum values of the errors are listed in ascending order as GS-6D, SVD-9D, Euler angles, axis angles, and quaternions. The result of quaternions has the largest error, and the GS-6D has the smallest error. The maximum distance between the errors is 0.076 radians, which corresponds to 4.35

(ii) Figure 10(b) shows that after a period of training, the differences between the results of various representations will gradually become smaller, and eventually, all will converge to a more stable result. According to the results of the 25th round, the quaternions have the largest error, and GS-6D has the smallest error. The maximum distance between the errors is 0.056 radians, which corresponds to 3.21

(iii) As shown in Figure 10(c), the value of the variance of the discontinuous representations is smaller, while the variance of the continuous representations with constraints is large. Representations of GS-6D and SVD-9D were jittery and unstable during training. The training process for Euler angles, axis angle, and quaternions is relatively smooth

5.3. Serial Network Experiments. The experiment was designed to verify the effectiveness of the serial network in terms of acceleration of convergence and improvement of accuracy. Five rotation representations were also selected, and each rotation representation was tested using the single-module rotation estimation network and the serial network. The experimental results of the two architectures were then compared.

We use the dataset provided in Section 5.1. The dataset was regenerated 25 times during the online training process, and each dataset was trained for 30 epochs. The serial network utilizes the results of the single-module rotation estimation network from round 20. The parameters of the neural network are selected from the set of data with the lowest error in the 20th training round of the single module, and then, the training is continued for 5 rounds to compare the errors in rounds 20 to 25. We trained the neural network utilizing the Adam optimizer, with a learning rate of 10^{-3}, while the learning rate was decayed using an exponential decay strategy with a decay rate of 0.998.

As shown in Figure 11 and Table 2, the upper group of lines represents the error of a single module, and the lower group of lines represents the error of a serial network. The error of the serial network decreases rapidly between rounds 20 and 21. After 5 rounds of training, the minimum error in the serial network is SVD-9D, and the maximum error is quaternions. Compared to single-module networks, the reductions are all over 50%. In the 25th round, the minimum error was 0.104 radians for the SVD-9D. The error is reduced by 0.2368 radians compared to the single module, which is 30.54% of the single module.

5.4. Stability Experiment. We train the neural network sufficiently to verify the stability of the online training method and also to verify the advantages of the serial network in terms of accuracy by the final convergence results. The experiment was designed to test the stability of online training, so instead of selecting all rotation representations, only the rotation representation of SVD-9D was chosen as the output of the neural network. On the other hand, we experimentally compare the final convergence results using both architectures (Section 4.3.1 and Section 4.3.2) as a way to verify the advantages of serial networks in terms of accuracy. The errors after training are illustrated in Figure 12.

After 115 iterations of the dataset, there was no overfitting, and the results were very stable. The minimum value for the single module of SVD-9D is 0.1453 radians, and the minimum value for the series network is 0.0298 radians. The error is reduced by 0.1155 radians, which is only 20.51% of that of the single module. It is concluded that using a serial network can converge quickly and obtain a highly accurate and stable result.

5.5. Application in Buddha Statues

5.5.1. Buddha Statue Dataset. Samples in the Buddha statue dataset come from the Huangze Temple in Guangyuan, Sichuan (see Figure 13). The collected Buddha statue needs to be manually rotated to the front. Since the point cloud of the Buddha is different from the CAD data, the error is relatively large, and the number is relatively small, so it is more difficult to converge. In addition to verifying the algorithm's effectiveness in this paper, the experiment can also be conducted to test the algorithm's performance in an environment with few samples and large errors.

The dataset for the current experiment is generated from 49 Buddha statues, which are divided into two groups: 42 point clouds are used to generate the training set, and the other 7 point clouds are used to generate the testing set. The experiment was trained using the online training method. For the training set, five thousand samples with different orientations are generated. The point cloud of each Buddha statue in the testing set generates ten different orientations, and 70 samples of different orientations are generated in total.

5.5.2. Experimental Results. This experiment is designed to verify the performance of the serial network in a real dataset. Therefore, only the serial network is used for the training of the neural network. It is more difficult to converge due to the small number of Buddha statues and large errors. In this experiment, the axis angle with less variance is selected as the output of the neural network, and the number of iterations should be increased as a way to obtain a more stable training process and higher accuracy.

We use the dataset provided in Section 5.5.1. The architecture of the serial network in this experiment is the same as in the previous experiments. During online training, each generated dataset is of size 5000. The number of iteration epochs for each dataset is 30. In the first stage of training, 50 rounds of dataset generation were performed, with a total

data size of 25,000 and a total of 1500 iteration epochs of training. We trained the neural network utilizing the Adam optimizer, with a learning rate of 10^{-3}, while the learning rate was decayed using an exponential decay strategy with a decay rate of 0.998. The experimental results are shown in Figure 14.

A set of comparatively well-trained parameters with an error of 0.387 radians from the training results of the first stage is selected as the parameters of the serial network. After another 20 rounds of training, the model has converged. The error of the serial network decreases rapidly at the beginning of training and then basically stabilizes after five rounds of training. The minimum value of the error for the single module is 0.4085 radians, and the minimum value of the error for the series network is 0.2850 radians, which is a reduction of 0.1235 radians, or 30.11%.

The current amount of Buddha data is small, so the accuracy was not as high as that of the airplane dataset. However, the advantages shown by the network are consistent with the previous experiments, and it can be concluded that the accuracy is bound to improve as the scan data increases.

6. Discussion and Conclusion

6.1. Discussion

6.1.1. Significant Effectiveness of Serial Network. First, there are two constraints in the serial network. One is that the inputs and outputs between different modules are strictly controlled, and the other is that the training goal of each module is to try to recover the initial state. Both of these provide directions for the learning of neural networks.

Second, the distribution of the input data of the last module changes considerably due to the multiple calls of the rotation estimation module. When the error of a single module is small, the input of the last module is almost close to the point cloud of the original orientation, which is very conducive to further convergence of the model.

6.1.2. Applicability of Online Training. Online training methods are well suited when the dataset needs to be generated manually, and the problem to be solved is the regression problem in supervised learning.

The prediction results of the regression problem are continuous values. Each time, the generated dataset can be considered the dataset with the same distribution as before but composed of different samples, which prevents the overfitting problem during training. In addition, the method uses less storage space, allowing convergence results to achieve high accuracy.

6.1.3. Summary of Rotation Representations. In this paper, the reasons for the relatively poor performance of quaternions in discontinuous representations are analyzed. It is concluded that the output dimensions of the neural network and its orthogonality are thought to be strong prior knowledge that can help with neural network training.

Continuous representations with constraints improve accuracy, and unconstrained discontinuous representations improve stability during training. We analyze the reasons for the higher accuracy of continuous representations with constraints. One reason is that the continuity of the representations makes it easier for the neural network to learn. Another reason is that constraints can influence the direction of parameter learning, allowing the neural network to learn hidden laws easily. Further research is needed as to the leading cause of the accuracy difference. All representations can be obtained with better results using the serial networks. When doing related projects, the appropriate representation can be selected as the neural network's output according to the requirements.

The use of PointNet and MLP in our experiments allows the point cloud to be mapped directly to the rotation representation space. This approach, which does not construct an intermediate template space, allows for a simpler model architecture and has demonstrated its effectiveness from experiments.

6.2. Conclusion. A systematic solution to the rotation estimation problem was provided through a series of experiments and summaries, and the research findings have been applied to the restoration work of the Huangze Temple Grottoes in Guangyuan, Sichuan, with expected results. First, we summarize the properties of the representations in the neural network and give recommendations for their selection. Second, the method of two-point recovery of the initial orientation is proposed and applied to the generation of the dataset. Finally, we use a serial network while using online training to improve the convergence speed and accuracy and make the training process more stable.

Data Availability

The data used to support the findings (Modelnet40 Aligned Objects) of this study have been deposited in the repository (10.1109/ICCV.2015.155). The data (Cave Buddha Statue) used to support the findings of this study have not been made available because the data also forms part of an ongoing study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been cosupported by the Key Laboratory of Information and Computing Science Guizhou Province of Guizhou Normal University and Sichuan Provincial Cultural Relics and Archeology Research Institute (grant number 11904-0621083).

References

[1] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, "Learning to reconstruct 3D human pose and shape via model-fitting in the loop," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2252–2261, Seoul, Korea (South), 2019.

[2] H. Zhang, Y. Tian, X. Zhou et al., "Pymaf: 3D human pose and shape regression with pyramidal mesh alignment feedback loop," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11446–11456, Montreal, QC, Canada, 2021.

[3] M. Kocabas, N. Athanasiou, and M. J. Black, "Vibe: video inference for human body pose and shape estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5253–5263, Seattle, WA, USA, 2020.

[4] X. Yu, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: a convolutional neural network for 6d object pose estimation in cluttered scenes," 2017, arXiv preprint arXiv: 1711.00199.

[5] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation*, pp. 1271–1278, Stockholm, Sweden, 2016.

[6] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113, Las Vegas, NV, USA, 2016.

[7] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2514–2523, Seattle, WA, USA, 2020.

[8] D. Bauer, T. Patten, and M. Vincze, "Reagent: point cloud registration using imitation and reinforcement learning," in *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14586–14594, Nashville, TN, USA, 2021.

[9] W. Changchang, F. Fraundorfer, J.-M. Frahm, and M. Pollefeys, "3D model search and pose estimation from single images using vip features," in *In 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, Anchorage, AK, USA, 2008.

[10] C. Steger, "Occlusion, clutter, and illumination invariant object recognition. International archives of photogrammetry remote sensing and spatial," *Information Sciences*, vol. 34, no. 3, pp. 345–350, 2002.

[11] E. Muñoz, Y. Konishi, V. Murino, and A. Del Bue, "Fast 6D pose estimation for texture-less objects from a single RGB image," in *In 2016 IEEE International Conference on Robotics and Automation*, pp. 5623–5630, Stockholm, Sweden, 2016.

[12] J. J. Lim, A. Khosla, and A. Torralba, "FPM: fine pose parts-based model with 3D CAD models," in *Computer Vision – ECCV 2014*, pp. 478–493, Springer, Cham, 2014.

[13] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," in *In Proceedings of the IEEE international conference on computer vision*, pp. 954–962, Santiago, Chile, 2015.

[14] W. Chen, X. Jia, H. J. Chang, J. Duan, and A. Leonardis, "G2l-net: global to local network for real-time 6D pose estimation with embedding vector features," in *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4233–4242, Seattle, WA, USA, 2020.

[15] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: a deep pointwise 3D keypoints voting network for 6D of pose estimation," in *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11632–11641, Seattle, WA, USA, 2020.

[16] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: consistent multi-view multi-object 6D pose estimation," in *European Conference on Computer Vision*, pp. 574–591, Springer, Cham, 2020.

[17] W. He, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2642–2651, Long Beach, CA, USA, 2019.

[18] D. Chen, J. Li, W. Zheng, and K. Xu, "Learning canonical shape space for category-level 6D object pose and size estimation," in *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11973–11982, Seattle, WA, USA, 2020.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, arXiv preprint arXiv:1312.6114.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, Honolulu, HI, USA, 2017.

[21] L. P. Cassinis, R. Fonod, E. Gill, I. Ahrns, and J. Gil-Fernández, "Evaluation of tightly- and loosely-coupled approaches in CNN-based pose estimation systems for uncooperative spacecraft," *Acta Astronautica*, vol. 182, pp. 189–202, 2021.

[22] N. T. Dantam, "Robust and efficient forward, differential, and inverse kinematics using dual quaternions," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1087–1105, 2021.

[23] Z. Lin, Y. Xiong, G. Cai et al., "Quantification of parkinsonian bradykinesia based on axis-angle representation and SVM multiclass classification method," *IEEE Access*, vol. 6, pp. 26895–26903, 2018.

[24] A. Perrusquía and Y. Wen, "Human-in-the-loop control using Euler angles," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 2, pp. 271–285, 2020.

[25] Z. Cao, Z. Chu, D. Liu, and Y. Chen, "A vector-based representation to enhance head pose estimation," in *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1188–1197, Waikoloa, HI, USA, 2021.

[26] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, Long Beach, CA, USA, 2019.

[27] J. Levinson, C. Esteves, K. Chen et al., "An analysis of SVD for deep rotation estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22554–22565, 2020.

[28] L. Jing, Y. Chen, and Y. Tian, "Coarse-to-fine semantic segmentation from image-level labels," *IEEE Transactions on Image Processing*, vol. 29, pp. 225–236, 2019.

[29] W. Zhan, G. Shi, Y. Chen et al., "Coarse-to-fine classification for diabetic retinopathy grading using convolutional neural network," *Artificial Intelligence in Medicine*, vol. 108, article 101936, 2020.

[30] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin, "Extensive facial landmark localization with coarse-to-fine convolutional network cascade," in *In Proceedings of the IEEE international conference on computer vision workshops*, pp. 386–391, Sydney, NSW, Australia, 2013.

[31] C. Wang, D. Xu, Y. Zhu et al., "Densefusion: 6D object pose estimation by iterative dense fusion," in *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3343–3352, Long Beach, CA, USA, 2019.

[32] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

[33] N. Sedaghat and T. Brox, "Unsupervised generation of a view point annotated car dataset from videos," in *2015 IEEE International Conference on Computer Vision*, pp. 1314–1322, Santiago, Chile, 2015.