Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 10, No. 3, 2024

Research on Robotic Arm Grasping Algorithm Based on an Enhanced Edge Network

Hongyi Li, Jianjun Zhu

School of information and Control Engineering, Jilin Institute of Chemical Technology, Jilin, Jilin 132022, China

Abstract: To address the issue of low grasp detection accuracy in robotic arms, an improved point cloud-based grasp detection model, TES-Net (TransformEdgeSAGE Network), is proposed, which builds upon the Edge Grasp Network architecture. In this model, PointTransformerConv layers are first employed to extract local features from the point cloud, integrating a self-attention mechanism to capture the complex relationships inherent within the point cloud data. Subsequently, the SAGEConv graph neural network module is utilized to perform feature aggregation on the adjacency graph, thereby mitigating the issue of neglecting the inter-point relationships during the feature extraction phase and enhancing the overall network performance. Experimental results demonstrate that, in comparison to existing state-of-the-art point cloud grasp detection methods, the proposed model exhibits superior generalization capability, improved robustness and stability, as well as higher accuracy.

Keywords: Robotic Arm Grasp Detection; Edge Grasp Detection Network; Self-Attention Mechanism; Graph Neural Network Module.

1. Introduction

In recent years, robotic arm grasping has gradually transitioned from repetitive tasks to more intelligent operations, necessitating higher demands on the system, which must be capable of computing multiple potential grasp poses. At present, mainstream methods for robotic arm grasp pose detection predominantly rely on 2D images. However, these methods are limited by their ability to only provide projection information of objects onto a 2D plane, lacking depth and stereoscopic perception. Consequently, they are less effective in handling complex scenes and object poses. In contrast, point cloud data directly captures the position and shape of objects in three-dimensional space, offering rich depth information. This enables more accurate determination of an object's location, orientation, and pose. As a result, point cloud data has garnered increasing attention and application in research that requires precise grasp localization in recent years.

In the field of robotics, grasp detection is crucial. Robots need to identify a set of feasible grasping poses by analyzing images, voxels, or point clouds to achieve stable object manipulation. Typically, these methods are categorized into two groups: SE(2) (the special Euclidean group in two dimensions) and SE(3) (the special Euclidean group in three dimensions)[1]. The SE(2) methods rely on top-view images for inference, while the SE(3) approaches utilize point clouds or voxel grids for their analysis. Compared to SE(2) methods, SE(3) techniques offer significant advantages as they can more readily identify side grasping poses of objects. By processing point cloud data, SE(3) methods comprehensively account for an object's orientation and position within three-dimensional space, thereby providing more accurate predictions of grasping poses.

Due to the significant advantages of the SE(3) method, this paper presents an improved design based on SE(3), resulting in a point cloud-based grasp detection network structure referred to as TES-Net (TransformEdgeSAGE Network). This network represents unique six-degree-of-freedom grasp poses through a pair of vertices within a graph. To enhance the model's accuracy and robustness, TES-Net constructs a K-nearest neighbor graph (KNN) and selects the top k point clouds for building an adjacency matrix by calculating cosine distances. This approach ensures that the network effectively captures spatial relationships among point clouds. Subsequently, PointTransformerConv is employed for feature extraction, while the extracted point cloud features are sampled and aggregated using the graph neural module SAGEConv to enrich and enhance these features. Finally, by thoroughly mining potential information within the point cloud data, we aim to improve predictions of grasping poses more effectively.

2. Research on Robotic Arm Grasping

2.1. 6-DoF Grasping Methods

Six Degrees of Freedom (6-DoF) grasping methods are generally categorized into two main approaches[2]. The first category consists of sample-based methods, such as GPD, PointNetGPD, and GraspNet. These methods typically utilize grasp sampling and evaluation modules to individually represent and assess each grasp pose[3]. Consequently, these approaches tend to require significant computational time during both training and execution. To improve computational efficiency, the present study adopts a strategy of representing different grasp poses through shared features, thereby significantly enhancing processing speed and efficiency.

The second category encompasses element prediction-based methods, which include both point-based and voxel-based approaches. Point-based methods estimate the grasp quality for all relevant points or voxels via a single forward pass. For example, S4G employs PointNet++ to predict the features and grasp quality for each individual point[5], while REGNet predicts the grasp pose by regressing the geometry surrounding sampled points[7]. Although classification-based methods are capable of addressing multiple grasp poses, they generally require a larger quantity of training data.

Voxel-based methods, on the other hand, represent spatial information through voxel grids. However, the memory

requirements for these methods increase cubically as the grid resolution is enhanced, imposing significant constraints on their applicability in high-resolution scenarios. As a result, while voxel-based methods provide structured spatial representations, they face challenges related to excessive memory consumption and reduced computational efficiency when processing high-resolution data, which in turn directly affects their real-time processing capabilities.

2.2. Grasp Pose Problem Description

The problem of 3D grasp pose detection involves identifying a combination of positions and orientations at the robotic arm's end-effector that can successfully grasp the object, based on the input point cloud data[8] . The point cloud is represented as shown in Equation (1), where p_i is a point in three-dimensional space, n is the number of points in the point cloud, and $\mathbb{R}^3$ represents the three-dimensional Euclidean space.

$$P = \{ p_i \in \mathbb{R}^3 \}_{i=1}^{n} \tag{1}$$

The pose of the robotic gripper can be represented as $\alpha = (C, R) \in SE(3)$, where $C \in \mathbb{R}^3$ denotes the position of the gripper's center in space, and $R \in SO(3)$ represents its orientation. The representation of the grasp pose is shown in Equation (2).

$$S: P \to \{ \alpha_i \in SE(3) \}_{i=1}^{m} \tag{2}$$

The key challenge in object grasping is to map the point cloud P to the m grasp poses detected within the scene. The grasp quality is represented by a function $\Phi: (P, \alpha) \mapsto [0, 1]$ that indicates the quality of the grasp pose α . When the same rotation and translation transformations are applied to both the object and the grasp pose, the predicted grasp quality should remain invariant. Therefore, this paper applies centering and random rotation augmentation to the point cloud data to improve data quality and enhance the model's generalization capability.

2.3. Definition of Robotic Arm Grasp Pose

In this paper, a grasp is represented as a pair of points in the point cloud: the approach point p_a and the contact point p_c . The approach point is the position where the gripper is close to the object, while the contact point is where the gripper makes contact with the object. The grasp direction is defined by estimating the normal at the approach point. The gripper's end-effector moves parallel to the vector n_c and approaches the object along the vector a_{ac} . The gripper's center is positioned at the ideal contact point at the midpoint of the fingers.

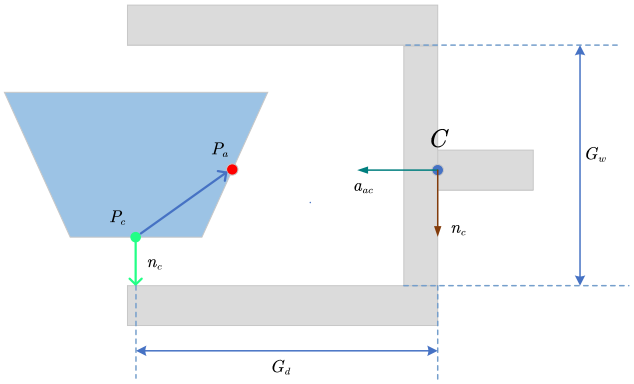

Fig 1. Diagram of Grasp Pose Representation

3. Network Model

3.1. Structure Design of the Grasp Detection Model

Feature Extraction Module: This module extracts useful geometric features from the input point cloud data. By using methods such as K-nearest neighbors (KNN), it constructs neighborhood relationships between points and aggregates local features through specific convolution operations, thus generating a local feature description for each point.

Graph Neural Network Module: Based on the local features generated by the Feature Extraction Module, this module further processes these features using graph convolutions. Graph convolutions effectively capture both the global and local structural information of the point cloud, enhancing the expressive power of the features.

Classification and Evaluation Module: This module performs the final classification on the processed features. Through a multi-layer perceptron (MLP) network, combined with the Sigmoid activation function, it evaluates each candidate grasp pose and outputs the probability of grasp success or the classification result, ultimately determining the best grasp pose.

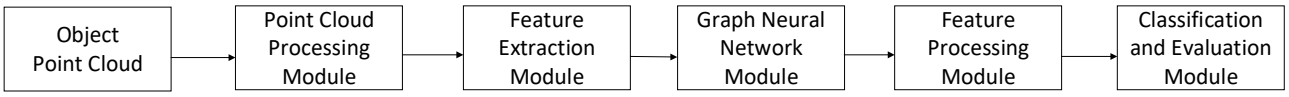

Fig 2. Edge Grasp Detection Network Model Framework Based on Point Cloud

3.2. Grasp Detection Network Based on TES-Net

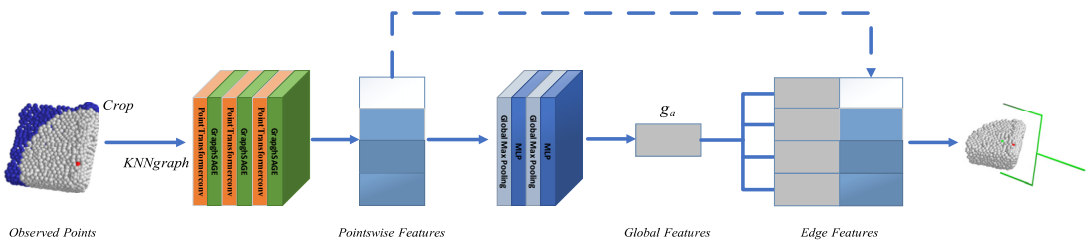

Fig 3. Grasp Detection Network Based on TES-Net

This paper mainly focuses on the Edge Grasp Detection Network (Edgegrasp-Net) and proposes an improved point cloud-based grasp detection network model, TES-Net. This method introduces PointTransformerConv and SAGEConv[9] into the edge grasp network to enhance its ability to capture global information, thereby improving the network's feature extraction and expressive capabilities. The network framework diagram is shown in Fig. 3.

As shown in Fig. 3, TES-Net first preprocesses the input point cloud data by constructing a K-nearest neighbor (KNN) graph and extracts local subsets related to the target grasp region. Then, features are extracted point-by-point using multiple PointTransformer [10]and GraphSAGE layers. The PointTransformer enhances the relationships between points through an attention mechanism, while GraphSAGE aggregates neighboring node information to further enrich the feature representation of each point. After the point-wise feature extraction, global max pooling (GlobalMaxPooling) is applied to aggregate all point features into a global feature vector that represents the entire point cloud. This global feature vector is then processed non-linearly through a multi-layer perceptron (MLP), serving as the global context information for subsequent grasp prediction. The global feature vector is combined with the point-wise features to generate edge features for each point, ensuring that each point's features contain both local geometric information and global structural information. Finally, the network uses these edge features to predict grasp positions, outputting the best grasp region and orientation, thereby providing precise guidance for the robotic arm's object detection and grasping localization.

3.3. PointTransformerConv Layer

Feature extraction is crucial in deep learning tasks for point cloud data. Traditional convolutional networks rely on regular grids, which reduces efficiency for unstructured point clouds. To address this, PointTransformerConv was developed, combining self-attention and global-local feature integration to significantly improve feature extraction and model representation.

The structure of PointTransformerConv is shown in Fig. 4. It takes the coordinates and positions of the input points, and maps them into a new feature space using two linear layers for the input points and their neighborhood points. Positional encoding is computed using a multi-layer perceptron (MLP) and added to the feature computation, helping the model capture spatial relationships between points. Self-attention scores are then calculated using an MLP, with the formula shown in Equation (3).

$$\gamma\left(\varphi\left(x_i\right)-\psi\left(x_j\right)+\delta\right) \tag{3}$$

The scores are used to measure the importance of each point relative to its neighboring points. Then, a linear transformation is applied to the features of each neighboring point, along with the addition of positional encoding. The formula for the feature linear transformation is shown in Equation (4).

$$\alpha\left(x_j\right)+\delta \tag{4}$$

Finally, the self-attention scores are element-wise multiplied with the transformed features, and the results for all points in the neighborhood are summed to obtain the final output features. The aggregation formula is shown in Equation (5).

$$y_i=\sum_{x_j \in N_i} \rho\left(\gamma\left(\varphi\left(x_i\right)-\psi\left(x_j\right)+\delta\right)\right) \odot\left(\alpha\left(x_j\right)+\delta\right) \tag{5}$$

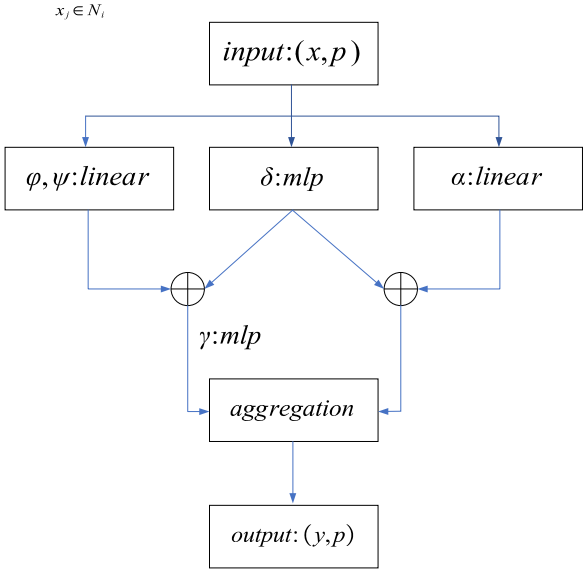

Fig 4. Structure Diagram of PointTransformerConv

3.4. Graph Neural Network Module (SAGE)

Graph Neural Networks (GNNs) have demonstrated strong expressive power and adaptability in handling graph-structured data. Among them, SAGE (GraphSAGE) is a classic graph neural network module that effectively improves feature learning and representation of graph data by aggregating information from neighboring nodes.

SAGE updates node feature representations through multi-layer aggregation of neighborhood node information. In this paper, the SAGE module is used for feature extraction from point cloud data, aggregating local features such as position and normal vectors, helping the model understand the local neighborhood structure of each point. This provides a foundation for subsequent global feature integration, enabling the model to more accurately assess the role of each point in the grasping task. Compared to traditional methods like PointNetConv, SAGE enhances the model's expressive power and task performance through multi-layer information propagation and feature aggregation.

The structure of SAGE is shown in Fig. 5, and its architecture involves sampling neighborhood nodes after the input node features. The sampling formula is shown in Equation (6).

$$N_v=sample\left(\mathcal{G},v\right) \tag{6}$$

Here, N_v represents the set of neighboring nodes of node. In this paper, the aggregation is performed by taking the mean of the features. The aggregation formula is shown in Equation (7).

$$h_v^{(k)}=Mean\left(\left\{h_u^{(k-1)} \mid u \in N_v\right\}\right) \tag{7}$$

After aggregation, the node features are updated through an MLP. The formula for updating the node features is shown in Equation (8).

$$h_v^{(k)}=MLP\left(h_v^{(k-1)} \| Aggregate\left(N_v\right)\right) \tag{8}$$

4. Dataset and Data Preprocessing

4.1. Dataset

In this paper, the YCB, BigBird, and KIT datasets were

chosen for model training experiments[11]. The training set contains 200 objects, while the test set contains 40 objects. To effectively utilize point cloud data for model training, key features such as point cloud coordinates and normal information were first extracted from the .npz files. The data was then processed using a custom dataset class, filtering out point clouds with fewer than 10 labels or those consisting entirely of positive samples, ensuring data diversity. During the preprocessing stage, the data underwent centering and random rotation augmentation. Finally, the data was subsampled, divided into training and test sets, and saved in PyTorch format for subsequent model training and evaluation.

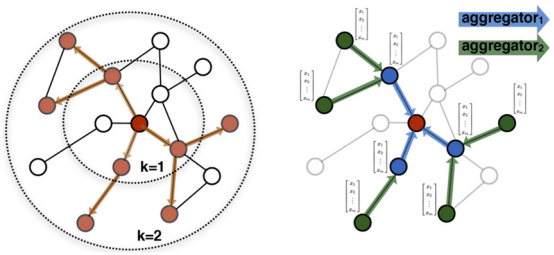

Fig 5. GraphSAGE Principle Diagram

4.2. Data Preprocessing

During the data preprocessing stage, this paper performed centering and random rotation augmentation on the point cloud data to improve data quality and enhance the model's generalization ability.

Centering: To ensure that the center of the point cloud is at the coordinate origin and reduce training bias caused by variations in the point cloud's position, the centroid of the point cloud data was first calculated, which is the average of the coordinates of all points. The centroid was then subtracted from the position of each point, achieving centering of the point cloud. The formula for calculating the centroid is shown in Equation (9).

$$\frac{1}{N}\sum_{i=1}^{N} P_i \tag{9}$$

Here, N is the number of points, and P_i is the position of the i-th point.

Random Rotation Augmentation: Random rotation augmentation is applied to the point cloud data to improve the model's robustness. First, a rotation matrix is generated by randomly selecting rotation angles in 3D space. The rotation matrix is composed of rotation matrices around the x, y, and z axes, as shown in Equations (10)-(12).

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{10}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{11}$$

$$R_z = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

The rotation matrix is the product of the rotation matrices mentioned above. Using this rotation matrix, the point cloud data can be rotated. This method allows the point cloud data to be rotated to different angles, thereby increasing the diversity of the training data and enhancing the model's robustness.

5. Model Training and Comparison Experiments

5.1. Model Training

The experimental simulation environment in this paper was run on an NVIDIA L20 GPU server, using the PyTorch neural network framework. Training data was created by simulating vertical and random placement environments, where each scene contains a random number of objects. Depth images captured from different camera viewpoints were corrupted with Gaussian noise, voxelized, and up to 2000 edge grasping candidates were generated for each scene. The grasping candidates were labeled through simulated grasping attempts. When generating 2000 candidates, 32 neighboring points were randomly sampled from the voxelized point cloud. Model training used the Adam optimizer with an initial learning rate of η\etaη, and the learning rate was halved after 6 epochs of stagnation in the test loss.

5.2. Comparison Experiments

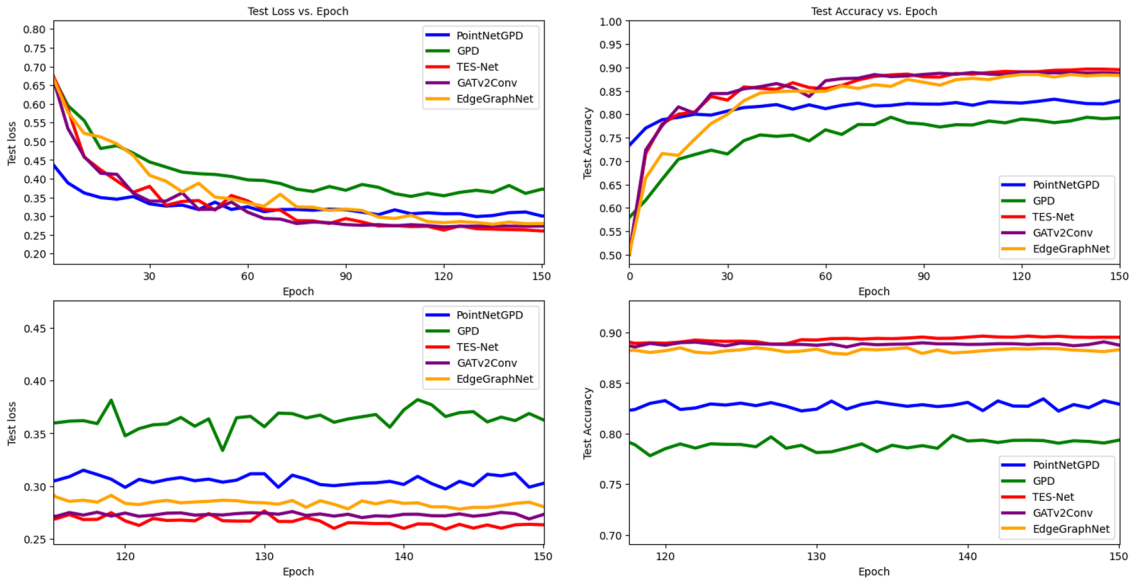

Fig 6. Algorithm Comparison Diagram

To objectively evaluate the performance of the improved network, this paper uses test accuracy (test_acc) and loss value as the evaluation metrics for the model. To verify the superiority of the proposed TES-Net model, comparison experiments were conducted with the current mainstream algorithms: PointNetGPD, GPD, the original Edgegrasp-Net algorithm, and a version of TES-Net with the Graph Neural Network module replaced by the GAT2V module. This experiment primarily compares the changes in classification accuracy and loss value across five different algorithms in the task of grasp detection.

Fig. 6 shows the variation curves of classification accuracy and loss value for five algorithms under a single view test. The blue and green curves represent the mainstream PointNetGPD and GPD algorithms, respectively. The red curve represents the proposed point cloud-based grasp detection network (TES-Net), while the purple curve shows the result after replacing the Graph Neural Network module in TES-Net with GATv2. The yellow curve represents the original EdgeGraspNet algorithm. The left plot shows the variation of loss value, and the right plot displays the trend of classification accuracy. The specific experimental data is shown in Table 1.

Table 1. Comparison Experiment Results

Method	Test_acc%	Loss/b
GPD	78.6	0.362
PointNetGPD	83.2	0.302
EdgeGraspNet	88.3	0.282
GATv2Conv	88.9	0.268
TES-Net	89.5	0.262

The experimental results indicate significant differences in classification accuracy and loss value among various grasp detection models. According to the data in Table 1, TES-Net performs the best in both classification accuracy and loss value, achieving a classification accuracy of 89.5% and a minimum loss value of 0.262. The GATv2Conv model follows closely behind, with a classification accuracy of 88.9% and a loss value of 0.268. In contrast, PointNetGPD and GPD perform relatively worse, with PointNetGPD achieving a classification accuracy of 83.2% and a loss value of 0.362, while GPD's classification accuracy is only 78.6%, with a loss value of 0.302. Combining the training curves from Fig. 6, it can be seen that TES-Net converges more quickly during training, with the loss value dropping rapidly. It maintains a low loss value and high classification accuracy throughout the entire training phase, demonstrating that TES-Net has high robustness and effectiveness in grasp detection tasks. Overall, TES-Net shows the best classification performance, outperforming existing mainstream grasp detection models, validating its ability to handle point cloud data in grasp detection tasks.

6. Simulation Validation

The simulation in this paper was conducted in the PyBullet simulation environment using the grasping simulator developed by Breyer et al. [13] , which includes the gripper of the Franka robot arm. The grasping simulator contains two different grasping environments: one is a vertically placed environment, and the other is a randomly placed environment. The simulation environment is shown in Fig. 7.

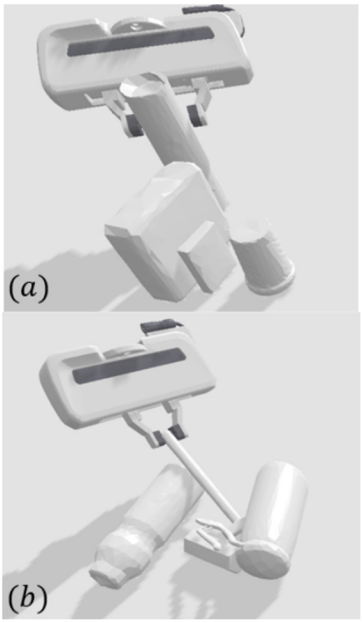

Fig 7. Left: Vertically Placed, Right: Randomly Placed

At the beginning of each test round, five random objects are placed in the workspace, as shown in Fig. 8. After the objects are loaded, grasping is performed sequentially. During the grasping process, the RGB data of the objects and the segmentation results can be observed from the workspace interface. Before each grasp attempt, a depth image of the scene is captured, Gaussian noise is added, and the point cloud or TSDF is extracted and input into the model. After the model outputs the grasp score, the grasping plan with the highest score is executed. Upon successful grasping, the object is removed. The test ends when all objects are cleared or when two consecutive grasp attempts fail. A total of 100 test rounds are performed.

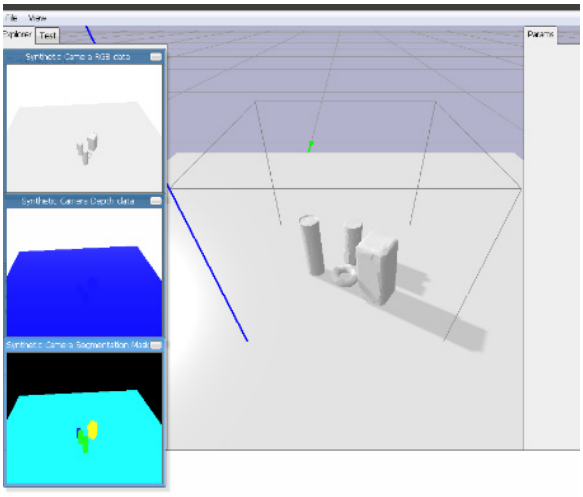

Fig 8. Workspace Environment

To objectively evaluate the performance of the improved network, this paper uses GSR (Grasp Success Rate) and DR (Clearance Rate) as simulation evaluation metrics, with the specific calculation formulas shown in Equation 13-14.

$$GSR = \frac{successful\ grasps}{total\ grasps} \tag{13}$$

$$DR = \frac{grasped\ objects}{total\ objects} \tag{14}$$

In the PyBullet simulation environment, this paper

conducted simulation comparison experiments with four algorithms: PointNetGPD, GPD, EdgeGrasp-Net, and TES-Net. The specific simulation grasping experiment data is shown in Table 2.

Table 2. Grasp Success Rate in Different Environments

Method	Vertically		Randomly	
	GSR(%)	DR(%)	GSR(%)	DR(%)
GPD	78.5	81.3	75.6	77.2
PointNetGPD	81.1	85.4	77.9	79.8
Edgegrasp-Net	90.6	92.2	87.1	90.2
TES-Net	94.1	95.2	92.2	93.5

The results in Table 2 show that TES-Net excels in both Grasp Success Rate (GSR) and Detection Rate (DR). In the vertical environment, TES-Net achieves a GSR of 94.1% and a DR of 95.2%. In the random environment, the GSR and DR are 92.2% and 93.5%, respectively. These metrics significantly outperform other methods. This indicates that TES-Net provides higher grasp success and detection rates across different scenarios, demonstrating its superiority in handling grasping tasks.

7. Summary

To address the issue of low object grasping accuracy in robotic arm grasp detection, this paper proposes an improved point cloud-based grasp detection network, TES-Net, based on the Edge Grasp Detection Network. By incorporating the PointTransformerConv and SAGEConv graph neural modules, the network enhances the ability to capture global information, thereby improving feature extraction and representation capabilities. This approach solves the problem of neglecting the relationships between points during feature extraction and improves overall network performance. The model was trained and validated on public datasets such as YCB, BigBird, and KIT, and simulation grasping experiments were conducted in two different grasping environments in PyBullet. Experimental results show that TES-Net achieves a classification accuracy of 89.5%, a loss value of 0.262, a success rate of 94.1% in the vertical grasping environment, and a success rate of 92.2% in the random grasping environment. Compared to existing mainstream point cloud grasp detection methods such as PointNetGPD and GPD, TES-Net demonstrates superior performance in terms of model generalization ability, robustness, and stability.

Acknowledgments

Jilin province science and technology development item (YDZJ202201ZYTS555).

References

[1] Liu J, Sun W, Yang H, et al. Deep Learning-Based Object Pose Estimation: A Comprehensive Survey[J]. arXiv preprint arXiv: 2405. 07801, 2024.

[2] Ten Pas A, Gualtieri M, Saenko K, et al. Grasp pose detection in point clouds[J]. The International Journal of Robotics Research, 2017, 36(13-14): 1455-1473.

[3] Liang H, Ma X, Li S, et al. Pointnetgpd: Detecting grasp configurations from point sets[C]//2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019: 3629-3635.

[4] Mousavian A, Eppner C, Fox D. 6-dof graspnet: Variational grasp generation for object manipulation[C]//Proceedings of the IEEE/ CVF international conference on computer vision. 2019: 2901-2910.

[5] Qin Y, Chen R, Zhu H, et al. S4g: Amodal single-view single-shot se (3) grasp detection in cluttered scenes[C]//Conference on robot learning. PMLR, 2020: 53-65.

[6] Qi C R, Yi L, Su H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space[J]. Advances in neural information processing systems, 2017, 30.

[7] Xu J, Pan Y, Pan X, et al. RegNet: Self-regulated network for image classification[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 34(11): 9562-9567.

[8] Huang H, Wang D, Zhu X, et al. Edge grasp network: A graph-based se (3)-invariant approach to grasp detection[C]//2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023: 3882-3888.

[9] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.

[10] Xiang Y, Schmidt T, Narayanan V, et al. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes[J]. arXiv preprint arXiv:1711.00199, 2017.

[11] Singh A, Sha J, Narayan K S, et al. Bigbird:(big) berkeley instance recognition dataset[C]//2014 IEEE International Conference on Robotics and Automation. 2014: 509-516.

[12] Kasper A, Xue Z, Dillmann R. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics[J]. The International Journal of Robotics Research, 2012, 31(8): 927-934.

[13] Breyer M, Chung J J, Ott L, et al. Volumetric grasping network: Real-time 6 dof grasp detection in clutter[C]//Conference on Robot Learning. PMLR, 2021: 1602-1611.