GraphGrasp: Lightweight And Efficient Graph-Guided 6-DoF Robotic Grasp Pose Estimation Network

Sheng Yu[1], **Di-Hua Zhai**[1*], **Yuanqing Xia**[1,2]

[1]School of Automation, Beijing Institute of Technology, Beijing, China
[2]Zhongyuan University of Technology, Zhengzhou, Henan, China
yusheng@bit.edu.cn, zhaidih@bit.edu.cn, xia_yuanqing@bit.edu.cn.

Abstract

6-DoF object grasping is a crucial skill for embodied intelligent robots. Previous methods often rely on large-scale networks for feature extraction, followed by grasp pose prediction, which increases the network's parameter count and overlooks the geometric and graph features of the point cloud. To address these challenges, we propose GraphGrasp, a graph-guided 6-DoF grasping pose prediction method. It performs graph analysis from the perspectives of scene, object, and grasping graphs. First, we introduce a graph feature embedding method based on local-global features to model the scene graph effectively. Then, we use a graph transformer strategy to represent spatial relationships between objects in the object graph. Finally, we propose a multi-metric, multi-level grasp pose evaluation algorithm to predict and explore graspable points, enabling effective construction of grasp graphs and accurate grasp pose evaluation. We test GraphGrasp on the GraspNet-1Billion dataset, and the results show that, compared to previous methods, it achieves nearly the same performance with about $\frac{1}{5}$ of the parameters of state-of-the-art methods, significantly improving grasp pose prediction speed. Additionally, in real-world robot grasping scenarios, GraphGrasp outperforms previous methods in practical grasp pose prediction tasks.

Code — https://github.com/BIT-robot-group/GraphGrasp

1 Introduction

Robot grasping is a vital skill for embodied intelligent robots, widely used in object manipulation (Fang et al. 2020; Kumra, Joshi, and Sahin 2020) and human-robot interaction (Wu et al. 2024). Traditional methods, such as (ten Pas et al. 2017; Hu et al. 2017; Herzog et al. 2012), depend on the 3D model or physical parameters of objects and predict grasp poses. While effective for some objects, these methods struggle with generalization to unknown ones.

Recent studies have proposed 2D planar grasp pose prediction methods, such as (Lenz, Lee, and Saxena 2015; Kumra, Joshi, and Sahin 2020; Yu et al. 2022; Cao et al. 2023; Zhou et al. 2022). While these methods enable fast grasp pose prediction, they often result in collisions between the robot gripper and objects in cluttered environments.

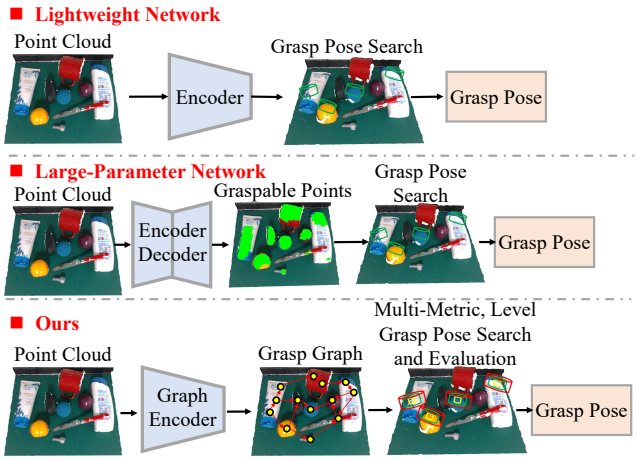

Figure 1: Comparison with previous related methods' pipeline. We present the pipeline for both lightweight and large-parameter networks. Unlike previous methods, we achieve a balance between speed and accuracy in grasp pose prediction by constructing a grasp graph network and a multi-metric, multi-level grasp pose prediction algorithm.

To address these challenges, researchers have proposed 6-DoF grasp pose prediction methods, such as (Fang et al. 2020; Zhao et al. 2021; Liang et al. 2019; Breyer et al. 2021; Dai et al. 2023; Mousavian, Eppner, and Fox 2019; Qin et al. 2020). Unlike 2D planar methods, 6-DoF approaches offer greater flexibility in object grasping. Notably, grasp pose estimation using the GraspNet-1Billion dataset has gained significant attention. In (Fang et al. 2020), Fang et al. introduce GraspNet, a lightweight method that extracts and analyzes point cloud features end-to-end to predict grasp poses. Subsequent research, such as (Gou et al. 2021; Lu et al. 2022; Li et al. 2021), has built on this approach, proposing additional lightweight grasp pose prediction methods.

Although lightweight and fast, these methods struggle with identifying graspable areas in point clouds, leading to lower grasp pose prediction accuracy. To address this, researchers have proposed graspable region prediction methods, followed by grasp pose prediction on these regions, such as (Wang et al. 2021; Ma and Huang 2022; Yu, Zhai, and Xia 2025). These methods often use larger backbone

*Corresponding author.

networks, like MinkUNet (Choy, Gwak, and Savarese 2019), to build segmentation networks similar to UNet (Ronneberger, Fischer, and Brox 2015), predicting graspable locations in the point cloud. While these methods improve grasp pose prediction, they increase the number of parameters and inference time.

Balancing network prediction performance (accuracy) and efficiency (speed) remains an unresolved issue in prior methods. This paper addresses this challenge by proposing a grasp pose prediction method that optimizes both accuracy and speed. For objects in the scene, they are placed on a table, and there is a spatial relationship between them. Graspable positions of objects are also interconnected, meaning that if one position is graspable, nearby positions are likely to be as well. Therefore, instead of relying on large parameter backbones for grasp pose prediction, we can directly predict graspable regions by utilizing these interrelations. This paper proposes a graph-guided grasp pose prediction method that balances both speed and accuracy.

First, to achieve a comprehensive understanding of the grasping scene, we propose a graph feature embedding method based on local-global features. This method constructs an initial graph network connecting object and background point clouds for an overall scene understanding. Then, to separate irrelevant point clouds and model spatial relationships between objects, we introduce an object graph construction strategy using a graph transformer. This strategy enhances the network's graspable position decision-making by connecting object point clouds. Finally, we propose a multi-metric, multi-level grasp pose evaluation algorithm to accurately predict and evaluate graspable points on object surfaces.

We evaluate GraphGrasp on the GraspNet-1Billion dataset (Fang et al. 2020). Experimental results demonstrate that our method is lightweight and efficient, achieving performance comparable to SOTA methods with only about $\frac{1}{6}$ of their parameters. Furthermore, GraphGrasp outperforms previous SOTA methods in real-world robotic grasping experiments, showing better generalization and accuracy.

In summary, the main contributions of this paper are summarized as follows:

- We propose a new graph feature embedding construction method based on local-global features, enabling the network to gain an initial understanding of the grasping scene and construct a scene graph.
- We propose an object graph construction strategy based on a graph transformer, allowing the network to fully understand the positional relationships between objects and connect object point clouds using the object graph.
- We propose a multi-metric, multi-level grasp point evaluation algorithm to achieve accurate prediction and evaluation of graspable points on object surfaces.
- We evaluate the performance of GraphGrasp on the GraspNet-1Billion dataset and in real-world scenarios. Results demonstrate that GraphGrasp achieves comparable grasp pose prediction performance to state-of-the-art methods, with significantly fewer parameters, and shows better generalization and accuracy in real-world settings.

2 Related Works

2.1 2D Planar Grasping

Traditional robotic grasping methods, such as (ten Pas et al. 2017; Hu et al. 2017; Herzog et al. 2012), typically require 3D object models and force closure analysis to assess grasp feasibility. A major drawback of this approach is its reliance on object models and physical information, limiting its generalization to unknown objects.

To address this issue, researchers have proposed data-driven robotic grasping methods that leverage neural networks to learn from human grasping experiences, guiding robots in grasping tasks (Lenz, Lee, and Saxena 2015; Kumra, Joshi, and Sahin 2020; Morrison, Corke, and Leitner 2020; Tong et al. 2024; Yang et al. 2024). Among these, 2D planar grasping methods were the first to be introduced and widely applied. In (Lenz, Lee, and Saxena 2015), Lenz et al. proposed a two-step grasp detection network: the first generates grasp candidates, and the second selects the best pose. However, this method is time-consuming due to sliding window search (Wang et al. 2019). In (Cao et al. 2023), a lightweight grasp detection network was introduced that uses RGB-D images to predict grasp quality, rotation angle, and opening width.

However, 2D planar grasping methods limit the ways in which robots can grasp objects, as they can only perform grasps from a vertical angle relative to the object. This significantly restricts the operational flexibility of the robot and increases the likelihood of collisions between the gripper and the object.

2.2 6-DoF Grasping

To address this, researchers have proposed 6-DoF grasp pose prediction methods, such as (ten Pas et al. 2017; Zhao et al. 2021; Liang et al. 2019; Ma and Huang 2022; Wang et al. 2021; Lu et al. 2022). Unlike 2D planar grasping, 6-DoF poses allow object grasping from any angle, offering greater flexibility and aligning with human grasping behavior. In (Liang et al. 2019), Liang et al. introduce PointNetGPD, which uses scene point clouds to sample grasp points and estimate 6-DoF poses. In (Fang et al. 2020), Fang et al. present the GraspNet-1Billion dataset and a baseline method using ApproachNet for approaching vectors and OperationNet for operational parameters. In (Zhao et al. 2021), Zhao et al. propose REGNet, which first predicts the grasp region from the point cloud and then estimates the 6-DoF grasp pose.

Although these methods achieve good results in 6-DoF grasp pose estimation, they struggle to effectively balance the network's accuracy and speed. For lightweight networks, the accuracy is often poor, while for those with higher accuracy, larger-scale network parameters are required, which affects the inference time. To address these issues, in this paper, we propose a grasp pose estimation method called GraphGrasp, which balances both speed and accuracy.

3 Method

3.1 Problem Statement

Given the scene point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$, our goal is to build a lightweight and efficient graph network to predict the 6-

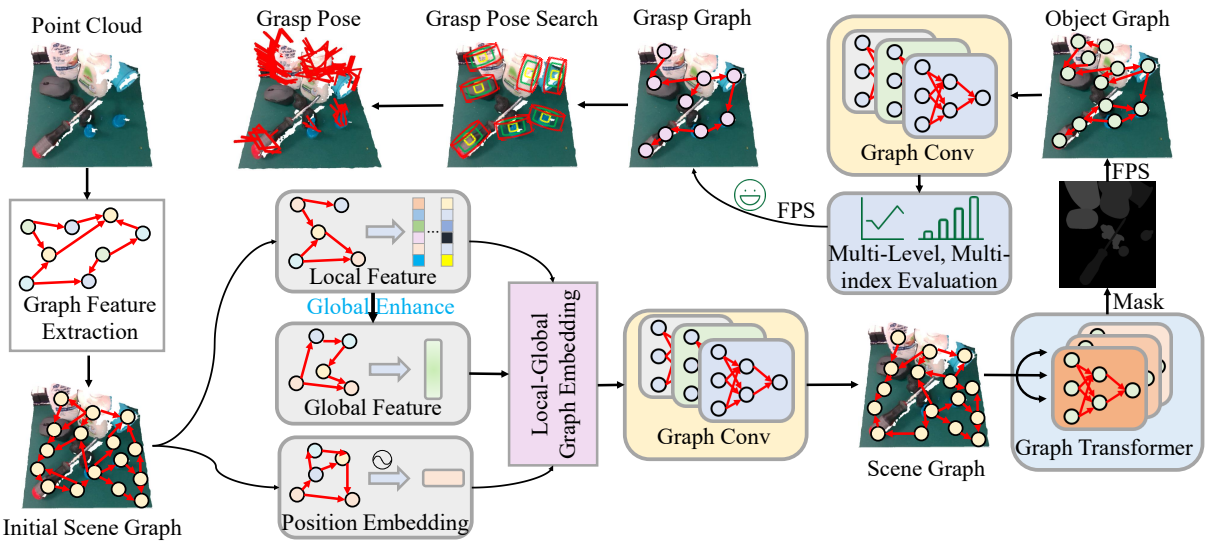

Figure 2: The pipeline of the GraphGrasp. We use scene point clouds as input and extract graph features graph convolution to construct the initial scene graph. For the initial graph and its features, we design a local-global graph feature embedding method, extracting features from both perspectives, and apply position encoding to the point cloud to obtain local-global scene graph feature embeddings. The scene graph embeddings are adjusted through graph convolution to produce the final graph. The transformer module is then used to further adjust and extract features, predict the object mask, and filter object nodes in the scene graph, yielding the object graph. Subsequently, we evaluate graspable points through multi-level, multi-metric pose search, selecting high-confidence graspable points to form the grasping graph. Finally, we perform grasp pose search on the grasping graph nodes to determine the grasp poses in the scene.

DoF graspable pose of objects in the scene. For each grasp pose g, it can be represented as $g = [R, t, w]$, where $R \in \mathbb{R}^{3\times3}$ denotes the rotation component of the grasp pose, $t \in \mathbb{R}^3$ denotes the translation component, and $w \in \mathbb{R}$ represents the opening width of the gripper.

3.2 Pipeline Overview

The network structure of GraphGrasp is shown in Figure 2. We construct a point cloud graph of the grasping scene at three levels. First, we build the scene graph, including both object and background point clouds, using a graph feature embedding method based on local-global features for initial construction. Next, we construct the object graph from the scene graph, designing a graph transformer-based strategy for object point cloud construction. Finally, we design a multi-level grasping graph construction algorithm to create the grasping graph, enabling thorough exploration of graspable points on the object surface. Using these graspable points, we predict and explore multi-scale grasp poses by constructing multi-scale cylindrical regions.

3.3 Scene Graph

In the initial state, we downsample the input point cloud $\mathcal{P}_{init} \in \mathbb{R}^{N\times3}$ by random sampling to obtain a cloud with N points and construct the initial graph $\mathcal{G}$ by randomly connecting the points. The graph $\mathcal{G} = (\mathcal{V}, \epsilon)$ consists of N nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \epsilon$. The adjacency matrix of $\mathcal{G}$ is $A \in \mathbb{R}^{N\times N}$, and the diagonal degree matrix is $D = \sum_j A_{ij}$, where $D \in \mathbb{R}^{N\times N}$. We calculate the symmetrically normalized adjacency matrix $\mathcal{A} \in \mathbb{R}^{N\times N}$ with h hops as $\mathcal{A} = D^{-\frac{1}{2}}AD^{\frac{1}{2}}$, and $\mathcal{A}_h = \mathcal{A}^h$.

The h-hop features learned by the network can be calculated as $\mathcal{F}_h = \lambda_h(\mathcal{A}_p\mathcal{F})$, where λ_h indicates the mapping function, $\mathcal{F}_h \in \mathbb{R}^{N\times d_h}$, and $\mathcal{F} \in \mathbb{R}^{N\times d}$ indicates the point cloud features extracted by PointNet++ (Qi et al. 2017).

The multi-hop features of the scene graph effectively integrate the background and objects across nodes, aiding the network in understanding the overall scene structure. To fully construct and extract these features, we propose a graph feature embedding method based on local-global features.

We first generate a local feature attention matrix $B_{loc} \in \mathbb{R}^{N\times d_h}$ using multi-hop features, which are then locally adjusted based on B_{loc}. To keep the network lightweight, we propose an efficient method to compute the local attention matrix. We separately calculate the maximum pooling and average pooling features of the multi-hop features and use softmax to compute $B_{loc} = \text{softmax}(\mathbf{A}(\mathcal{F}_h) + \mathbf{M}(\mathcal{F}_h))$, where $\mathbf{A}$ represents average pooling and $\mathbf{M}$ represents max pooling.

Then, we adjust the multi-hop features using B_{loc},

$$\mathcal{F}_{loc} = B_{loc} \otimes \mathcal{F}_h \tag{1}$$

where $\otimes$ indicates element-wise product.

In the process of grasp prediction, global information is more important than local information. It can grasp the overall grasp pose of the object, avoiding collisions with other objects. Therefore, in this paper, we also construct a global information self-enhancement module. First, we take the local information $\mathcal{F}_{loc}$ of the scene as the input to the module

and use global average pooling to calculate the global features, resulting in the global average feature $F_{gl} \in \mathbb{R}^{1\times d_h}$,

$$F_{gl} = \text{MLP}(\mathbf{A}(\mathcal{F}_h)) \tag{2}$$

Then, we use F_{gl} to perform global self-enhancement on the local features, calculated as:

$$\mathcal{F}_{gl} = \mathcal{F}_h + \mathbf{N}(\mathcal{F}_h F_{gl}^T)F_{gl} \tag{3}$$

where $\mathbf{N}$ represents feature normalization, and $\mathcal{F}_{gl}$ represents the globally self-enhanced features.

Considering that the position of the point cloud plays an important role in the construction of the scene graph, we also encode the position of the point cloud, calculated as: $\mathcal{F}_e = \text{MLP}(\mathcal{P}_{init})$, where $\mathcal{F}_e \in \mathbb{R}^{N\times d_h}$ indicates the position embedding.

Finally, we concatenate the local features, all features, and position embeddings of the scene graph, and perform further graph feature extraction and graph updates using edge convolution to obtain the final scene graph feature $\mathcal{F}_{scene}$.

3.4 Object Graph

Constructing the scene graph helps establish the interaction between object and background point clouds, aiding in overall scene analysis. During grasp pose prediction, the robot focuses on the object, requiring the exclusion of irrelevant points, such as the background. Objects can be classified into objects and non-objects, with the robot focusing only on the object's position. Thus, constructing the object graph becomes a classification problem.

Since the object graph directly influences grasp pose estimation, precise classification and graph construction are essential. Thus, we propose a new transformer graph module to predict and adjust the attention of the object graph features.

First, we take the scene graph feature $\mathcal{F}_{scene} \in \mathbb{R}^{N\times\mathcal{D}}$ as input, and generate the *query, key,* and *value*, which indicate by $q \in \mathbb{R}^{N\times\mathcal{D}_q}$, $k \in \mathbb{R}^{N\times\mathcal{D}_k}$, and $v \in \mathbb{R}^{N\times\mathcal{D}_v}$ respectively, where $\mathcal{D}_*, * \in \{q, k, v\}$ indicates the feature dimension.

Next, we normalize the q and v tensors, and apply the softmax function to the k tensor. To generate multiple graph features, we first divide q into several heads, resulting in $q \in \mathbb{R}^{N\times H\times(\mathcal{D}_q/H)}$, where H denotes the number of heads. We set $\mathcal{D}_q/H = \mathcal{D}_k = \mathcal{D}_v$ and utilize the Einstein summation convention (*einsum*) for performing matrix multiplication in the transformer.

Then, we follow the calculation process of the transformer and calculate the attention map, which can be got by $\alpha = k \odot v$, where $\odot$ indicates the *einsum*, $\alpha \in \mathbb{R}^{\mathcal{D}_k\times\mathcal{D}_v}$ indicates the attention map.

Next, we generate the adjusted features with the attention map: $\mathcal{F}_\alpha = q \odot \alpha$, where $\mathcal{F}_\alpha \in \mathbb{R}^{N\times H\times\mathcal{D}_v}$ indicates the adjusted feature map.

Then, we generate multi-graph features on the *value* branch. First, we create a random graph adjacency matrix A_{rand} to establish a graph G, which can be computed by $G = v \odot A_{rand}$, where $A_{rand} \in \mathbb{R}^{\mathcal{D}_v\times\mathcal{D}_v\times\mathcal{D}_k}$, $G \in \mathbb{R}^{N\times\mathcal{D}_k\times\mathcal{D}_v}$.

To establish a graph on itself, we add a self-loop to the v tensor by $v + I \to v$, where I represents the identity matrix.

We reshape v into $\mathbb{R}^{N\times\mathcal{D}_v\times\mathcal{D}_v}$ and add it to the graph G. Using G, we establish the relationship between v and q by $\mathcal{F}_G = q \odot G$, where $\mathcal{F}_G \in \mathbb{R}^{N\times H\times\mathcal{D}_v}$ indicates the graph features.

Finally, we add the $\mathcal{F}_G$ and $\mathcal{F}_\alpha$ together to obtain $\mathcal{F}_f$. Then, we can get the final object graph feature $\mathcal{F}_{obj}$ by $\mathcal{F}_{obj} = \text{MLP}(\mathcal{F}_f) + \mathcal{F}_{scene}$.

After obtaining the object graph features, we use convolution to build a binary classifier that categorizes node types in the scene graph, retaining only nodes that correspond to object surface points. These surface points form the nodes of the object graph $\mathcal{G}_{obj} = (\mathcal{V}_{obj}, \epsilon_{obj})$. During training, we continuously adjust $\mathcal{G}_{obj}$ to ensure the nodes lie on the object surface and establish necessary connections between them.

3.5 Grasp Graph

The prediction and exploration of graspable points on object surfaces are crucial for robot grasping. To evaluate grasp poses effectively, we establish a multi-metric, multi-level grasp pose evaluation mechanism. This evaluation is performed on the dataset, where each grasp pose is scored, guiding the network to prioritize higher-scoring poses.

Multi-Metric Grasp Pose Evaluation: First, we examine the flatness of the contact surface. A flat contact surface between the gripper and the object enhances stability during grasping. Let the initial contact point be p_i. We then search for the K-nearest points around p_i, forming the point set $P^K = \{p_k | k = 1, \ldots, N\}$. Next, we compute the normal vectors of p_i and each point in P^K, and determine the cosine distance between them: $S_f = \frac{1}{2N}\sum_{i=1}^{2}\sum_{k=1}^{N}\frac{<F_i,F_k>}{||F_i||\cdot||F_k||}$, where F_i indicates the normal vector of the contact point, F_k indicates the normal vector of the neighborhood points. Since we use a parallel gripper to perform grasp, so we have two contact points. We use S_f as a evaluation metric to evaluate the quality of the grasp pose.

We also need to assess the stability of the grasp pose when the gripper closes. If the normal vector of the contact surface aligns with the closing direction of the gripper, the object remains stable. Otherwise, it may shift or collide with other objects. Therefore, we use a stability metric to evaluate the grasp pose, calculated by the following equation: $S_s = \frac{1}{2}\sum_{i=1}^{2}\frac{|<F_i,F_g>|}{||F_i||\cdot||F_g||}$, where F_g indicates the close vector of the gripper.

We assess the grasp pose quality using the centroid metric. If the gripper's closing vector aligns with the object's center of gravity, the grasp is assumed to be more stable and reliable. To achieve this, we consider the two contact points, p_l and p_r, between the gripper and the object. By determining the contact line between these points, we evaluate the grasp quality based on the distance from the center of gravity to the line. This distance is calculated as follows: $S_g = 1 - \frac{||\overrightarrow{p_l p_c}\times\overrightarrow{p_r p_c}||}{||\overrightarrow{p_l p_r}||}$, where p_l and p_r indicate the left contact point and right contact point respectively, p_c indicates the center of gravity.

Finally, we identify a grasp pose that satisfies the metrics mentioned above. However, the gripper's proximity to the object may cause collisions in real-world applications. To

address this, we use a collision metric to evaluate the grasp quality. In this paper, the collision metric is determined by the distance between the gripper's endpoint and the contact point, calculated as follows: $S_c = \min(||\overrightarrow{p_l p_{el}}||, ||\overrightarrow{p_r p_{er}}||)$, where p_{el} and p_{er} indicate the left endpoint and right endpoint of the gripper respectively.

The force-closure metric is also an important metric to directly evaluate the quality of the grasp pose. The final grasp score of the grasp pose can be calculated by $S = \beta_1 S_F + \beta_2 S_f + \beta_3 S_s + \beta_4 S_g + \beta_5 S_c$, where S_F indicates the score of the force-closure metric, and we set $\beta_1 = 0.7, \beta_2 = 0.05, \beta_3 = 0.1, \beta_4 = 0.05, \lambda_5 = 0.1$.

Multi-Level Grasp Pose Evaluation: In cluttered scenes, graspable nodes at the center of the scene can easily hop to surrounding nodes, while nodes at the edges find it harder to do so. Therefore, we assign higher scores to grasp nodes capable of multiple hops: $S_h = \left\lfloor \frac{S - S_{min}}{S_{max} - S_{min}} \times H_S \right\rfloor$, where H_S is the maximum number of hops (up to 10), and S_{max} and S_{min} are the maximum and minimum grasp quality scores in the scene, respectively.

Grasp Graph Construction: Using the object graph features, we build a 3-layer graph convolutional network to refine the object graph and form a preliminary grasp graph. Multiple convolutional layers serve as regression heads to predict the grasp and level scores for each node on the initial grasp graph, and the Top-k nodes are selected as the grasp graph nodes $\mathcal{V}_{grasp}$.

$$\mathcal{V}_{grasp} = \text{Top}(\text{Conv}(F_{graph}(\mathcal{F}_{obj}))) \tag{4}$$

where F_{graph} indicates the graph convolution,

During training, by adjusting the parameters of the graph convolutional and convolutional neural networks, we continuously refine the grasp graph $\mathcal{G}_{grasp} = (\mathcal{V}_{grasp}, \epsilon_{grasp})$ and its features $\mathcal{F}_{grasp}$. The nodes in the grasp graph represent the predicted graspable points in the scene. To maintain network efficiency, we predict only a subset of graspable points. Increasing the input point cloud scale allows for the exploration of more graspable points.

3.6 Grasp Pose And Loss Function

In the process of grasp pose prediction, we follow the grasp pose generation method in GraspNet (Fang et al. 2020), predicting the relevant parameters of the grasp pose by constructing multi-scale cylindrical regions.

First, we compute the binary classification loss during object graph construction: $L_{mask} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{cls}(\hat{m}, m)$, where m is the predicted mask, $\hat{m}$ is the ground truth, and $\mathcal{L}_{cls}$ is the cross-entropy loss. Next, we use Huber loss to calculate the regression losses for the multi-metric and multi-level grasp scores, denoted as L_{grasp} and L_{lv}, respectively. We also apply Huber loss to compute the losses for the opening widths and rotation angles, denoted as L_{width} and L_{angle}, respectively. Finally, we calculate the view loss L_v for the gripper's approaching directions, following (Fang et al. 2020).

The final loss is calculated by $L = \gamma_1 L_{mask} + \gamma_2 L_{grasp} + \gamma_3 L_{lv} + \gamma_4 L_{width} + \gamma_5 L_{angle} + \gamma_6 L_v$, where γ_* indicates the weights for the loss function.

We have provided more details for the loss functions, please refer to the supplementary materials.

4 Experiment

4.1 Dataset

Our experiments for grasp pose detection are conducted on the GraspNet-1Billion dataset (Fang et al. 2020). This large-scale dataset comprises 190 scenes with 256 views captured by both RealSense and Kinect cameras. We train our network using 100 of these scenes and tested on the others, categorized as seen, similar, and novel groups of 30 scenes each. This evaluation approach enable us to assess the performance and generalization ability of the network. As RealSense and Kinect camera images are similar, we choose to use only the RealSense portion for training and testing, as is also done in most previous methods.

4.2 Implementation Details

The point cloud contains $N = 15000$ points, and we construct a KNN graph, where each node is randomly connected to 32 neighboring nodes. To ensure the real-time performance of the computation, we select 7000 points on the object's surface as the nodes of the object graph, and then select 1024 nodes from them as the nodes of the grasp graph. We set $\gamma_1 = 0.5, \gamma_2 = 0.2, \gamma_3 = 0.2, \gamma_4 = 0.2, \gamma_5 = 0.2, \gamma_6 = 0.3$. The weight of the loss function is determined following previous related methods, such as (Fang et al. 2020; Wang et al. 2023; Lu et al. 2022). The evaluation metrics is the same as (Fang et al. 2020).

4.3 Results on GraspNet-1Billion Dataset

In the experiment, we compare the performance of related methods on the GraspNet-1Billion dataset, with the obtained experimental results shown in Table 1. We compare the method performance on the seen, similar, and novel test sets and use the evaluation metrics set in (Fang et al. 2020) for grasp pose evaluation.

Table 1 categorizes methods into three types: lightweight models, large-parameter models, and the proposed method. Lightweight models have fewer parameters and faster inference, ensuring better real-time performance in robot grasp pose prediction. However, experimental results show that these models perform poorly on the dataset, failing to accurately estimate object grasp poses. This is primarily due to their inability to effectively evaluate and predict graspable points, as well as their neglect of graph information in the point cloud, resulting in inefficient use of point cloud data for grasp pose prediction.

To address this issue, researchers have proposed grasp pose prediction networks with larger parameters, such as GSNet. Unlike lightweight models, large-parameter models use a UNet-like network to predict graspable points and extract features, followed by grasp pose prediction. While these models significantly improve grasp pose prediction performance, they come with the trade-off of requiring more inference time.

Compared to large-parameter networks like GSNet and MTGrasp, GraphGrasp's prediction performance is nearly

Method	Seen			Similar			Novel		
	AP	$AP_{0.8}$	$AP_{0.4}$	AP	$AP_{0.8}$	$AP_{0.4}$	AP	$AP_{0.8}$	$AP_{0.4}$
GPD(ten Pas et al. 2017)	22.87	28.53	12.84	21.33	27.83	9.64	8.24	8.89	2.67
PointnetGPD(Liang et al. 2019)	25.96	33.01	15.37	22.68	29.15	10.76	9.23	9.89	2.74
GraspNet(Fang et al. 2020)	27.56	33.43	16.95	26.11	34.18	14.23	10.55	11.25	3.98
gou et al.(Gou et al. 2021)	27.98	33.47	17.75	27.23	36.34	15.60	12.25	12.45	5.62
li et al.(Li et al. 2021)	36.55	47.22	19.24	28.36	36.11	10.85	14.01	16.56	4.82
GraNet(Wang et al. 2023)	43.33	52.56	34.03	39.98	48.66	32.00	14.90	18.66	7.76
FGC-GraspNet(Lu et al. 2022)	49.68	53.06	33.73	40.09	38.40	23.31	16.01	17.37	5.03
GSNet(Wang et al. 2021)	67.12	78.46	60.90	54.81	66.72	46.17	24.31	30.52	14.23
MTGrasp(Yu, Zhai, and Xia 2025)	**72.62**	**84.06**	**67.30**	**63.39**	**75.97**	**54.98**	**28.61**	**35.61**	**15.94**
ours	64.88	75.05	58.64	56.91	67.84	48.94	24.83	30.60	13.13

Table 1: Comparison with other methods on the GraspNet-1Billion dataset.

identical to these SOTA methods and even surpasses them in some metrics. For instance, on the same test set, GraphGrasp outperforms GSNet, achieving better pose estimation with fewer parameters. This is due to GraphGrasp's full utilization of point cloud graph information, allowing autonomous exploration of graspable points through point cloud node connections without relying on a large-parameter backbone.

At the same time, we also provide a comparison of the parameter quantities of some methods, as shown in Table 2. Based on the results, it can be seen that GraphGrasp achieves better pose estimation performance with only a parameter quantity similar to that of lightweight models such as GraspNet. Meanwhile, compared to large-parameter models, GraphGrasp achieves comparable pose estimation performance with only about $\frac{1}{5}$ of the parameters.

Methods	Light	Parameters (M)	AP
GraspNet(Fang et al. 2020)	✓	1.7	27.56
GraNet(Fang et al. 2020)	✓	2.8	43.33
FGC-GraspNet(Lu et al. 2022)	✓	1.8	49.68
GSNet(Wang et al. 2021)		15.4	67.12
MTGrasp(Yu, Zhai, and Xia 2025)		17.6	**72.62**
Our	✓	3.2	64.88

Table 2: The parameter comparison of related methods.

Finally, we provide a visual comparison of the grasp poses of GraphGrasp and other related methods on the GraspNet-1Billion dataset, as shown in Figure 3. During the grasp pose visualization process, we only plot the top-30 scoring grasp poses. Here, we provide the most representative grasp pose prediction methods, GraspNet and GSNet. The three rows of images in Figure 3 are selected from the seen, similar, and novel test sets, respectively, to test the network's generalization ability.

GraspNet and GSNet share some common drawbacks: (1) They fail to estimate grasp poses for certain objects in the scene. (2) For some objects, they predict effective grasp poses, but these poses violate physical rules, leading to issues such as penetration or collisions, causing grasp failures. (3) In Figure 2, a red color indicates high confidence in completing the grasp. However, both GraspNet and GSNet show predicted grasp poses with colors deviating from red, indicating lower confidence in the grasp pose quality.

In contrast, GraphGrasp effectively predicts grasp poses

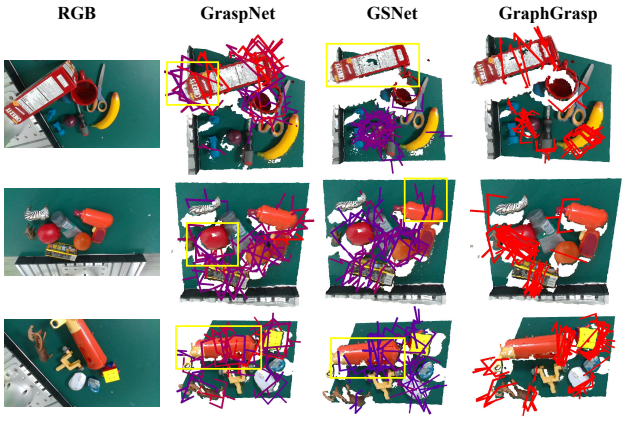

Figure 3: Comparison experiments on the GraspNet-1Billion dataset. The grasp poses within the yellow bounding boxes represent cases of prediction failure.

for most objects, ensuring they adhere to physical rules and reducing collision risks. Additionally, GraphGrasp highlights predicted grasp poses in red, indicating high confidence in their suitability for object grasping.

4.4 Ablation Studies

To further validate the effectiveness of the proposed method, we conduct multiple ablation experiments, and the results are shown in Table 3.

L	**G**	**E**	Seen			Similar			Novel		
			AP	$AP_{0.8}$	$AP_{0.4}$	AP	$AP_{0.8}$	$AP_{0.4}$	AP	$AP_{0.8}$	$AP_{0.4}$
			34.12	42.03	24.68	32.42	37.62	23.13	12.22	16.78	8.53
✓			40.70	48.01	34.57	37.68	45.17	29.43	14.05	18.53	9.81
	✓		49.10	57.73	43.22	44.71	54.09	37.86	16.55	20.89	10.99
✓	✓		52.40	62.00	44.41	46.73	56.29	39.26	18.59	23.11	11.69
✓		✓	46.53	55.62	40.35	51.68	51.32	34.50	16.19	20.18	10.74
	✓	✓	58.63	67.48	51.21	52.92	60.37	44.95	22.72	25.78	12.58
✓	✓	✓	**64.88**	**75.05**	**58.64**	**56.91**	**67.84**	**48.94**	**24.83**	**30.60**	**13.13**

Table 3: Ablation studies on the GraspNet-1Billion dataset. **L** indicates the local-global graph embedding, **G** indicates the graph transformer, and **E** indicates the multi-metric, multi-level grasp pose evaluation.

First, we validate the impact of LGE and GT on the network performance. The experimental results obtained after removing all modules are shown in the first row of Table 3. According to the results, we can observe that after removing all modules, the network performance significantly declines.

We then apply either LGE or GT to the network, replacing EVA with a classification module to determine graspable points. Experimental results show that the network's performance improves initially, demonstrating that LGE and GT enhance grasp pose prediction. Moreover, GT provides a greater performance boost than LGE, as it helps the network better distinguish object nodes from non-object nodes, aiding in subsequent grasp pose prediction. The application of both LGE and GT further improves network performance.

We validate the impact of the EVA algorithm on network performance by applying it with either LGE or GT. Experimental results are shown in the second-to-last and third-to-last rows of Table 3. The results demonstrate that EVA enhances the network's performance, primarily due to its multi-metric, multi-level grasp pose evaluation, which guides the network to more accurately evaluate and learn high-quality grasp poses.

Finally, when we apply all the modules to the network, the network's performance reaches its optimal value, proving the effectiveness of the modules proposed in this paper.

4.5 Grasp Pose Detection in the Real World

To test the generalization of the methods in real-world scenarios, we apply GraspNet, GSNet, and GraphGrasp for grasp pose estimation. Predicted grasp poses are shown in Figure 4. The experiment uses an Intel RealSense D435 camera to capture RGB images and point clouds of previously unknown objects.

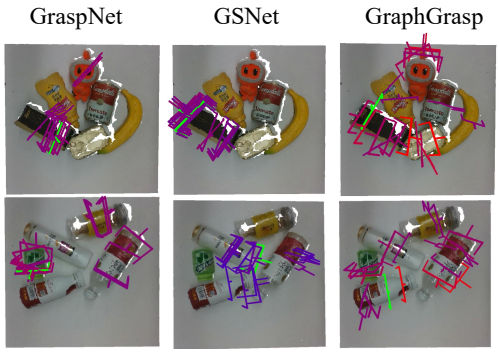

Figure 4: Grasp pose estimation in the real world. The green gripper indicates the best grasp pose.

Based on the grasp pose visualization results, we observe that for GraspNet and GSNet, there are certain limitations in their generalization. GraspNet misses some graspable objects and fails to predict grasp poses effectively. While GSNet performs better than GraspNet, its predicted poses may lead to collisions between the gripper and object. In contrast, GraphGrasp can effectively predict the grasp poses of objects in the scene, and the grasp poses are distributed near the object center, demonstrating high reliability.

4.6 Robotic Grasping Experiments

We apply GraphGrasp to real-world grasping experiments with a UR3 robot. In the experiment, we test the network's grasp pose prediction performance in single-object and multi-object cluttered scenes, with the objects randomly placed on a tabletop. During grasping, the robot performs the grasping task based on the highest-scoring grasp pose. The grasp poses, optimal grasp poses, and robot performance in some scenes are shown in the Figure 5.

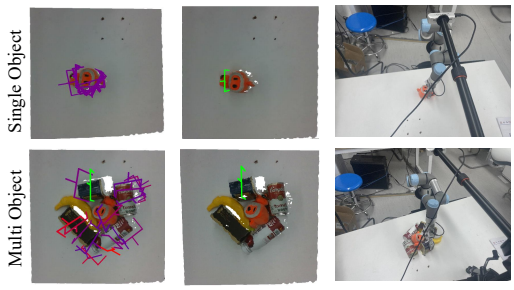

Figure 5: Robot grasping experiments in the real world.

We compare GraphGrasp to GraspNet and GSNet, delivering the grasp success rates in Table 4. Based on the experimental results, we can conclude that GraphGrasp outperforms GraspNet and GSNet in real-world applications.

Authors	Success Rates (%)
GraspNet(Fang et al. 2020)	85.7
FGC-GraspNet(Lu et al. 2022)	86.9
Scale-Grasp(Ma and Huang 2022)	87.2
GSNet(Wang et al. 2021)	89.8
Our	**92.1**

Table 4: Average grasp success rates in the robotic grasping experiments.

Although GraphGrasp achieves good results, there are still some failures in practical grasping experiments. First, for some objects with complex shapes, the graspable geometric features are not obvious. Then, for some small-scale objects, due to factors such as camera noise, the point cloud often contains only a small number of points, resulting in fewer graspable pose predictions, which sometimes leads to the object being ungraspable.

5 Conclusion

In this paper, we propose GraphGrasp, a graph-guided 6-DoF grasp pose estimation method. First, a local-global feature-based graph embedding method is introduced to model the grasping scene. An object graph construction method using a transformer graph module connects object point clouds. Finally, a multi-metric, multi-level grasp pose evaluation algorithm is designed to improve pose evaluation. Tested on the GraspNet-1Billion dataset and real-world scenarios, GraphGrasp achieves comparable grasp pose prediction and better generalization with fewer parameters than SOTA methods.

Acknowledgments

The work was supported by the National Natural Science Foundation of China under Grant 62173035, Grant 61803033 and Grant 61836001.

References

Breyer, M.; Chung, J. J.; Ott, L.; Siegwart, R.; and Nieto, J. 2021. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In *Conference on Robot Learning*, 1602–1611.

Cao, H.; Chen, G.; Li, Z.; Feng, Q.; Lin, J.; and Knoll, A. 2023. Efficient Grasp Detection Network With Gaussian-Based Grasp Representation for Robotic Manipulation. *IEEE/ASME Transactions on Mechatronics*, 28(3): 1384–1394.

Choy, C.; Gwak, J.; and Savarese, S. 2019. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE/CVF conference on computer vision and pattern recognition*, 3075–3084.

Dai, Q.; Zhu, Y.; Geng, Y.; Ruan, C.; Zhang, J.; and Wang, H. 2023. Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 1757–1763.

Fang, H.-S.; Wang, C.; Gou, M.; and Lu, C. 2020. Graspnet-1billion: A large-scale benchmark for general object grasping. 11444–11453.

Gou, M.; Fang, H.-S.; Zhu, Z.; Xu, S.; Wang, C.; and Lu, C. 2021. Rgb matters: Learning 7-dof grasp poses on monocular rgbd images. In *IEEE International Conference on Robotics and Automation*, 13459–13466.

Herzog, A.; Pastor, P.; Kalakrishnan, M.; Righetti, L.; Asfour, T.; and Schaal, S. 2012. Template-based learning of grasp selection. In *IEEE International Conference on Robotics and Automation*, 2379–2384.

Hu, Y.; Li, Z.; Li, G.; Yuan, P.; Yang, C.; and Song, R. 2017. Development of Sensory-Motor Fusion-Based Manipulation and Grasping Control for a Robotic Hand-Eye System. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7): 1169–1180.

Kumra, S.; Joshi, S.; and Sahin, F. 2020. Antipodal robotic grasping using generative residual convolutional neural network. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 9626–9633.

Lenz, I.; Lee, H.; and Saxena, A. 2015. Deep Learning for Detecting Robotic Grasps. *The International Journal of Robotics Research*, 34(4-5): 705–724.

Li, Y.; Kong, T.; Chu, R.; Li, Y.; Wang, P.; and Li, L. 2021. Simultaneous semantic and collision learning for 6-dof grasp pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3571–3578.

Liang, H.; Ma, X.; Li, S.; Görner, M.; Tang, S.; Fang, B.; Sun, F.; and Zhang, J. 2019. PointNetGPD: Detecting grasp configurations from point sets. In *IEEE International Conference on Robotics and Automation*, 3629–3635.

Lu, Y.; Deng, B.; Wang, Z.; Zhi, P.; Li, Y.; and Wang, S. 2022. Hybrid Physical Metric For 6-DoF Grasp Pose Detection. In *2022 International Conference on Robotics and Automation*, 8238–8244.

Ma, H.; and Huang, D. 2022. Towards Scale Balanced 6-DoF Grasp Detection in Cluttered Scenes. In *Annual Conference on Robot Learning*.

Morrison, D.; Corke, P.; and Leitner, J. 2020. Learning robust, real-time, reactive robotic grasping. *The International Journal of Robotics Research*, 39(2-3): 183–201.

Mousavian, A.; Eppner, C.; and Fox, D. 2019. 6-DOF GraspNet: Variational grasp generation for object manipulation. In *IEEE International Conference on Computer Vision*, 2901–2910.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Qin, Y.; Chen, R.; Zhu, H.; Song, M.; Xu, J.; and Su, H. 2020. S4g: Amodal single-view single-shot se (3) grasp detection in cluttered scenes. In *Conference on robot learning*, 53–65.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, 234–241.

ten Pas, A.; Gualtieri, M.; Saenko, K.; and Platt, R. 2017. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14): 1455–1473.

Tong, L.; Song, K.; Tian, H.; Man, Y.; Yan, Y.; and Meng, Q. 2024. A novel RGB-D cross-background robot grasp detection dataset and background-adaptive grasping network. *IEEE Transactions on Instrumentation and Measurement*.

Wang, C.; Fang, H.-S.; Gou, M.; Fang, H.; Gao, J.; and Lu, C. 2021. Graspness discovery in clutters for fast and accurate grasp detection. In *IEEE/CVF International Conference on Computer Vision*, 15964–15973.

Wang, H.; Niu, W.; Zhuang, C.; and Guibas, L. J. 2023. Granet: a multi-level graph network for 6-DoF grasp pose generation in cluttered scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 937–943.

Wang, S.; Jiang, X.; Zhao, J.; Wang, X.; Zhou, W.; and Liu, Y. 2019. Efficient fully convolution neural network for generating pixel wise robotic grasps with high resolution images. In *IEEE International Conference on Robotics and Biomimetics*, 474–480. IEEE.

Wu, H.; Jing, Y.; Cheang, C.; Chen, G.; Xu, J.; Li, X.; Liu, M.; Li, H.; and Kong, T. 2024. Unleashing Large-Scale Video Generative Pre-training for Visual Robot Manipulation. In *International Conference on Learning Representations*.

Yang, G.; Jia, T.; Liu, Y.; Liu, Z.; Zhang, K.; and Du, Z. 2024. MCT-Grasp: A Novel Grasp Detection using Multimodal Embedding and Convolutional Modulation Transformer. *IEEE Sensors Journal*.

Yu, S.; Zhai, D.-H.; and Xia, Y. 2025. MTGrasp: Multiscale 6-DoF Robotic Grasp Detection. *IEEE/ASME Transactions on Mechatronics*, 30(1): 156–167.

Yu, S.; Zhai, D.-H.; Xia, Y.; Wu, H.; and Liao, J. 2022. SE-ResUNet: A Novel Robotic Grasp Detection Method. *IEEE Robotics and Automation Letters*, 7(2): 5238–5245.

Zhao, B.; Zhang, H.; Lan, X.; Wang, H.; Tian, Z.; and Zheng, N. 2021. Regnet: Region-based grasp network for end-to-end grasp detection in point clouds. In *IEEE International Conference on Robotics and Automation*, 13474–13480.

Zhou, Z.; Wang, S.; Chen, Z.; Cai, M.; and Kan, Z. 2022. A robotic visual grasping design: Rethinking convolution neural network with high-resolutions. *arXiv preprint arXiv:2209.07459*.