IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 43, No. 2, April, 2026, pp. 935-942
Journal homepage: http://iieta.org/journals/ts

Multi-Object Grasp Pose Estimation and Simulation via an Enhanced IWGNet Framework Integrated with Robot Operating System

Jianbo Hou[1], Mingkang Zhou[1], Yuhang Xiao[2*], Xinlong Ji[2]

[1] Peter the Great St. Petersburg Polytechnic University Joint Polytechnic Institute, Xi'an Technological University, Xi'an 710021, China
[2] College of Electronic Information Engineering, Xi'an Technological University, Xi'an 710021, China

Corresponding Author Email: xiaoyuhang@st.xatu.edu.cn

https://doi.org/10.18280/ts.430230

Received: 10 November 2025
Revised: 12 January 2026
Accepted: 26 February 2026
Available online: 30 April 2026

Keywords:
robotic manipulator, multi-object grasping, grasp pose estimation, Inception–Weighted Feature Fusion–Generative Grasping Convolutional Neural Network, robot operating system simulation

ABSTRACT

In complex multi-object environments, grasp pose estimation accuracy is often limited, while prioritization in multi-object grasping and execution stability remain insufficiently addressed. To overcome these challenges, based on the Generative Grasping Convolutional Neural Network (GGCNN), an improved Inception–Weighted Feature Fusion–GGCNN (IWGNet) architecture was proposed. A multi-object grasping simulation and validation system was constructed using the robot operating system. Within the proposed framework, an enhanced inception module was introduced to deepen the network structure and improve spatial feature representation. In addition, a weight-adaptive weighted feature fusion module was designed to enable effective cross-level feature integration, thereby enhancing the stability of grasp pose prediction. Furthermore, a perception-guided grasp planning strategy was incorporated, in which a weighted evaluation mechanism was constructed by jointly considering object depth and grasp stability. The proposed method was validated within a robot operating system-based simulation environment, where grasping experiments were conducted using a UR5 robotic manipulator under conditions involving 5, 7, and 10 objects. Experimental results demonstrate that an average grasp success rate of 92.7% was achieved, confirming the strong effectiveness and robustness of the proposed approach in complex multi-object scenarios.

1. INTRODUCTION

In industrial human–robot collaboration systems, stable and reliable grasping capability is regarded as a fundamental prerequisite for achieving efficient production. The accuracy of grasp detection algorithms has been shown to directly affect both the operational safety of robotic manipulators and the success rate of task execution [1]. Consequently, a variety of vision-based algorithmic frameworks have been developed in the field of robotic grasp detection. Early work by Jiang et al. [2] introduced deep learning into vision-guided grasp pose estimation, where a seven-degree-of-freedom grasp representation was proposed, thereby establishing the foundation for data-driven approaches. To address the issue of parameter redundancy, Lenz et al. [3] designed a cascaded dual-network architecture, in which candidate grasp regions were first generated by a shallow network and subsequently refined by a deeper network to select optimal grasp poses. The adoption of a five-degree-of-freedom representation was shown to significantly improve computational efficiency. Building upon these advancements, Wang et al. [4] identified limitations in conventional methods related to gripper width constraints and discretized prediction strategies. To overcome these issues, a continuous four-dimensional grasp representation was proposed, which enhanced the continuity of parameter prediction. With increasing demands for real-time performance, Morrison et al. [5] developed a depth-image-based pixel-wise grasp prediction network, enabling millisecond-level inference by effectively leveraging three-dimensional scene information. Furthermore, a hierarchical detection framework was proposed by Chu et al. [6], in which the collaborative optimization strategy of classification and regression—commonly employed in object detection tasks—was adapted to robotic grasp detection. Accordingly, a dual-branch prediction structure was designed.

A two-stage object detection network was developed by Jiang et al. [7] to achieve accurate localization of regions of interest with orientation information, along with grasp bounding box prediction. External disturbances, including inter-object interference and illumination variations, were effectively mitigated. However, the training process was conducted on a self-constructed dataset consisting of single-object and sparsely distributed multi-object scenarios, while grasp prediction in densely stacked multi-object environments was not adequately addressed. Guo et al. [8] proposed a collaborative detection framework integrating visual perception and grasp decision-making. A dual-branch convolutional neural network combined with a single-shot prediction mechanism was employed, resulting in enhanced grasp accuracy and reliability in complex and stacked

environments. An improved Single Shot MultiBox Detector framework was adopted by Mei et al. [9], in which object detection and grasp localization were jointly achieved through a cascaded pose detection model. However, grasp orientation was treated as a classification problem, which introduced angular errors. A method based on You Only Look Once version 3 (YOLOv3) and depth information was presented by Zhang et al. [10], where object contours were segmented and grasp poses were subsequently computed. Nevertheless, no explicit grasp prioritization or sequencing strategy was incorporated. Through hierarchical detection, graspability evaluation, binocular pose estimation, and dynamic grasp sequencing, both grasping efficiency and reliability were significantly improved in structured environments. In addition, Zhang et al. [11] integrated an improved Mask Region-based Convolutional Neural Network framework with object detection, where grasp poses were estimated by combining surface normal vectors and orientation information. A depth-based strategy was further introduced to determine grasp execution order.

With the rapid advancement of intelligent manufacturing, multi-object grasping tasks in unstructured environments have become increasingly prevalent. The disordered distribution and dense stacking of target objects have been shown to impose significant challenges on traditional grasp detection algorithms, particularly in terms of accuracy and operational safety [12]. Although the generative grasping convolutional neural network enables end-to-end grasp pose estimation, limitations remain in its ability to sufficiently extract spatial features and effectively fuse multi-level representations. As a result, grasping accuracy is constrained in complex scenarios [5]. In addition, existing multi-object grasping approaches generally lack a prioritization mechanism that jointly considers depth information and grasp stability. Consequently, execution efficiency is reduced when robotic manipulators operate in densely cluttered and stacked environments [13]. To address these limitations, an improved grasp pose estimation network based on Inception–Weighted Feature Fusion–Generative Grasping Convolutional Neural Network is proposed, in which feature extraction and cross-level feature fusion mechanisms are systematically enhanced. Furthermore, a multi-object grasping simulation system is constructed based on the robot operating system framework. By incorporating a perception-driven decision-making strategy, efficient and sequential grasp execution is achieved in complex multi-object scenarios. The proposed approach provides a robust technical foundation for the automation and intelligent upgrading of industrial applications, including object sorting and warehouse logistics systems [14].

2. IMAGING MODELING

2.1 Kinematic modeling of the robotic manipulator

The Denavit–Hartenberg representation is widely recognized as a fundamental method for kinematic modeling of robotic manipulators. By employing four standardized parameters, namely d_i, a_i, α_i, and θ_i, the spatial transformation between adjacent link coordinate frames is systematically described, thereby enabling geometric parameterization of serial robotic manipulators. The definitions of these parameters are provided as follows (with respect to joint i):

• Link offset (d_i): Defined as the translational displacement along the z_{i-1}-axis from the x_{i-1}-axis to the x_i-axis.

• Link length (a_i): Defined as the distance along the x_{i-1}-axis from the z_{i-1}-axis to the z_i-axis.

• Link twist (α_i): Defined as the angle required to rotate the coordinate frame of link i-1 about its own x_{i-1}-axis until it is parallel to the x_i-axis of the coordinate frame of link i.

• Joint angle (θ_i): Defined as the angle through which link i rotates about the z_{i-1} -axis to achieve alignment with the coordinate frame of link i-1.

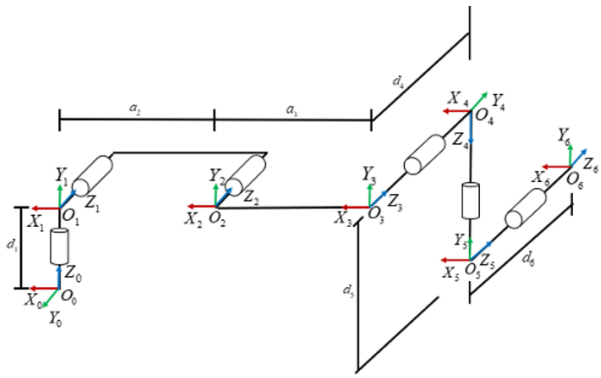


Figure 1. Denavit–Hartenberg model of the UR5 robotic manipulator

Figure 1 shows the Denavit–Hartenberg model of the UR5 robotic manipulator. Robotic manipulator kinematics consists of both forward kinematics and inverse kinematics. In forward kinematics, the pose of the end-effector is determined from the given joint angles θ_i. The homogeneous transformation matrix from coordinate frame $i-1$ to frame i, denoted as ${}^{i-1}_{i}T$, is defined in Eq. (1). The overall transformation matrix T from the base coordinate frame to the end-effector frame is obtained through successive matrix multiplication. Consequently, the spatial pose of the end-effector can be explicitly determined.

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)\,cos(\alpha_i) & sin(\theta_i)\,sin(\alpha_i) & a_i\,cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)\,cos(\alpha_i) & -cos(\theta_i)\,sin(\alpha_i) & a_i\,sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

Inverse kinematics is concerned with solving the joint angles from a desired end-effector pose. This process is typically performed using matrix decomposition and trigonometric analytical methods. Taking the first joint angle θ_1 as an example, it is expressed in Eq. (2). The remaining joint angles can be derived in a similar manner. These formulations provide the theoretical foundation for precise motion control of the robotic manipulator.

$$ \theta_1 = atan2\left(d_6\,a_y - p_y,\, p_x - d_6\,a_x\right) - atan2\left(d_4, \pm\sqrt{\left(d_6\,a_y - p_y\right)^2 + \left(p_x - d_6\,a_x\right)^2 - d_4^2}\right) \quad (2) $$

2.2 Depth camera imaging and coordinate transformation model

An imaging model is constructed based on a depth camera. The color imaging process follows the pinhole camera model. A three-dimensional point in the world coordinate system, denoted as $P_W(X_W,Y_W,Z_W)$, is transformed into the camera

coordinate system as $P_C(X_C,Y_C,Z_C)$ through the extrinsic parameter matrix, as expressed in Eq. (3), where R_W^C denotes the rotation matrix and T_W^C represents the translation vector [15].

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R_W^C & T_W^C \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3}$$

Depth imaging is performed based on the principle of binocular stereo vision. The depth of a spatial point is estimated by computing the disparity between corresponding points in the left and right infrared images. The depth value Z is calculated in Eq. (4), where f denotes the effective focal length, b represents the baseline distance, and x_1 and x_2 correspond to the horizontal projection coordinates of the target point in the left and right images, respectively [16].

$$Z = \frac{f \cdot b}{x_1 - x_2} \tag{4}$$

To achieve accurate mapping from image pixels to the robotic manipulator base coordinate system, a three-stage coordinate transformation process is required [17]. In the first stage, transformation to the camera coordinate system is performed. Based on the camera intrinsic parameter matrix M_{ci}, pixel coordinates (u_i,v_i) are converted into three-dimensional camera coordinates (X_C,Y_C,Z_C). In the second stage, transformation from the camera coordinate system to the robot base coordinate system is conducted. The transformation matrix T_{cam}^{base}, obtained through hand–eye calibration, is applied to establish the transformation between the camera coordinate system and the robotic manipulator base coordinate system. In the third stage, the grasp representation is converted into a three-dimensional grasp pose. The two-dimensional grasp parameters (x,y,θ,w), obtained from grasp detection, are mapped into a physically executable three-dimensional grasp pose of the robotic manipulator. In particular, the gripper opening width W is computed as follows:

$$W = \frac{Z_{cam} \cdot d_x}{f} \cdot w \tag{5}$$

3. GRASP POSE ESTIMATION NETWORK DESIGN

3.1 Design of the improved inception module

To address the limited spatial feature representation capability of the generative grasping convolutional neural network, an improved inception module is introduced to deepen the network architecture. The original inception module is characterized by high computational cost and complex gradient propagation. To overcome these limitations, several optimizations are implemented to enhance performance [18]. First, two consecutive 3 × 3 convolutional layers are employed to replace a single 5 × 5 convolutional layer. This substitution preserves the receptive field while significantly reducing computational complexity. Second, an additional average pooling branch is incorporated and combined with max pooling. Max pooling is utilized to capture salient features such as edges and contours, whereas average pooling preserves global and smooth information, thereby reducing feature loss. Third, 1 × 1 convolutions are extensively applied to integrate channel-wise information and facilitate efficient information flow. At the same time, the use of smaller convolutional kernels effectively reduces the number of parameters, thereby mitigating the risk of overfitting.

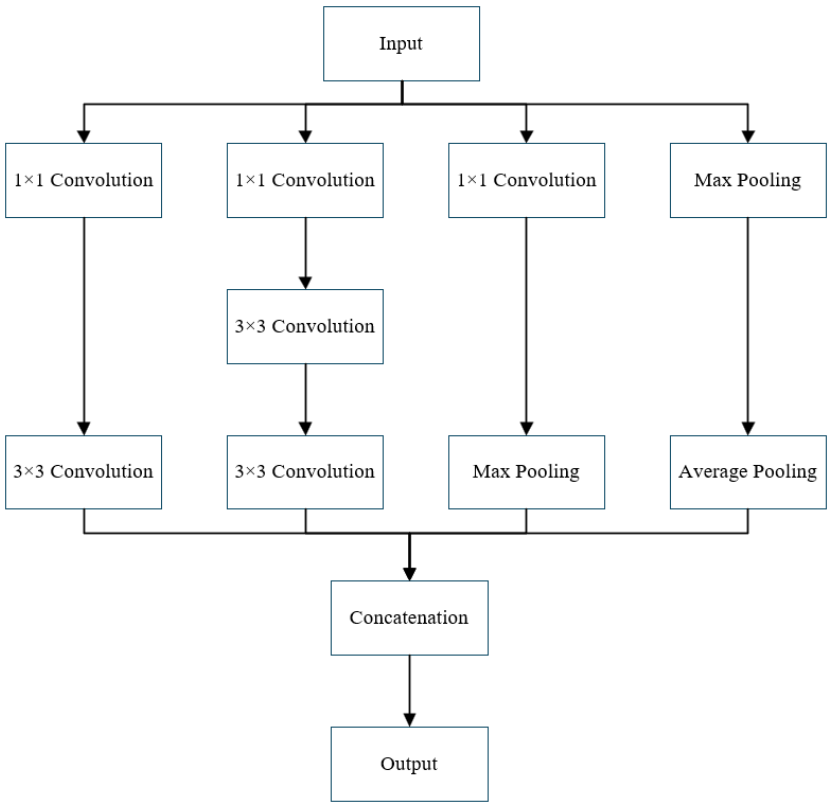


Figure 2. Architecture of the improved inception module

The improved inception module, as illustrated in Figure 2, consists of four parallel branches:

• Branch 1: 1 × 1 convolution followed by a 3 × 3 convolution;

• Branch 2: 1 × 1 convolution followed by two consecutive 3 × 3 convolutions;

• Branch 3: 1 × 1 convolution followed by max pooling;

• Branch 4: 3 × 3 max pooling followed by average pooling.

The outputs of all branches are concatenated along the channel dimension to achieve multi-scale feature fusion. The mathematical formulation of each branch is described as follows.

Branch 1:

$$X_1=W_{1\times1,1}*X+b_{1\times1,1} \tag{6}$$

$$X_{11}=W_{3\times3,1}*X_1+b_{3\times3,1} \tag{7}$$

Branch 2:

$$X_2=W_{1\times1,2}*X+b_{1\times1,2} \tag{8}$$

$$X_{21}=W_{3\times3,2}*X_2+b_{3\times3,2} \tag{9}$$

$$X_{22}=W_{3\times3,3}*X_{21}+b_{3\times3,3} \tag{10}$$

Branch 3:

$$X_3=W_{1\times1,3}*X+b_{1\times1,3} \tag{11}$$

$$X_{31}=M_{3\times3}(X_3) \tag{12}$$

Branch 4:

$$X_{41}=M_{3\times3}(X) \tag{13}$$

$$X_{42}=A(X_{41}) \tag{14}$$

Final output:

$$Y=\odot(X_{11},X_{22},X_{31},X_{42}) \tag{15}$$

where, X denotes the input tensor, * represents the convolution operation, $M_{3\times3}$ denotes a 3 × 3 max pooling operation, A represents the average pooling operation, $W_{k\times k,i}$ denotes the $k \times k$ convolution kernel in the i-th branch, and b denotes the bias term. This module enhances the network's capability to analyze the geometric features of object surfaces through multi-scale feature extraction.

3.2 Design of the weight-adaptive weighted feature fusion module

To address the limitations associated with ineffective cross-level feature fusion, a weight-adaptive weighted feature fusion module is designed, as illustrated in Figure 3. Within this module, a channel attention mechanism is introduced to enable adaptive fusion between low-level features, which primarily encode fine-grained texture and edge information, and high-level features, which capture global contextual semantics [19].

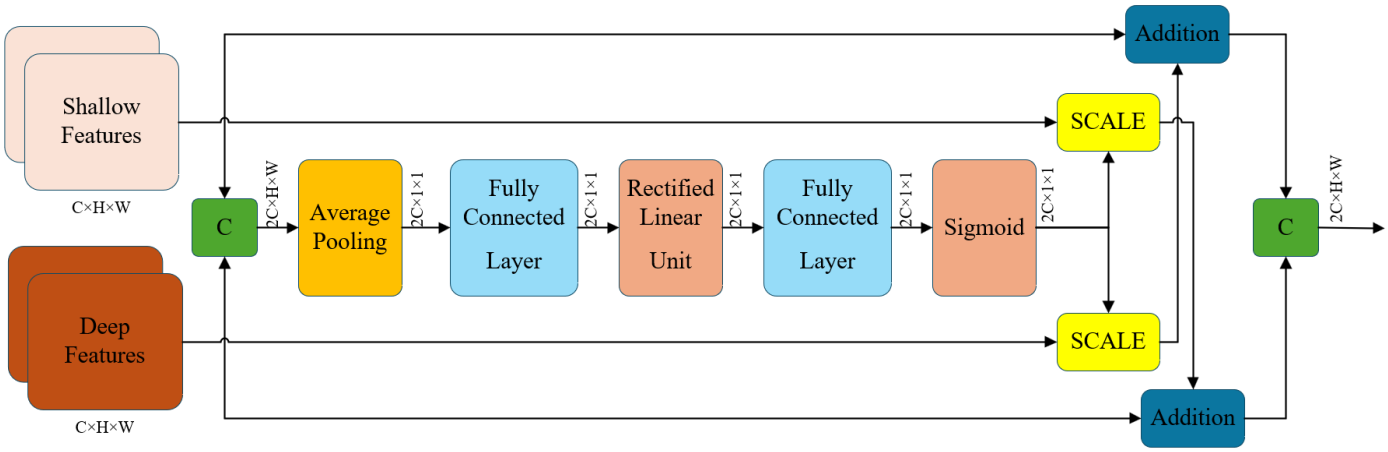


Figure 3. Architecture of the weighted feature fusion module

The module consists of three core stages. In the first stage, feature concatenation is performed. The shallow feature map $X_{shallow}\in R^{b\times C_1\times H\times W}$ and the deep feature map $X_{deep}\in R^{b\times C_2\times H\times W}$ are concatenated along the channel dimension to form a fused feature representation $X_{concat}=Concat(X_{shallow},X_{deep})$. Through this operation, high-resolution spatial features from shallow layers and high-level semantic features from deeper layers are effectively integrated.

A channel attention mechanism is employed. First, global average pooling is applied to the concatenated feature map to compress the spatial dimensions, resulting in a vector Z of size 1 × 1 × (C1+C2). The pooled output of the c-th channel is computed as:

$$Z_c=\frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W}X_{concat,c}(i,j) \tag{16}$$

Subsequently, channel attention weights $\hat{s}\in R^{b\times(C_1+C_2)}$ are learned through two fully connected layers, followed by rectified linear unit and sigmoid activation functions. The obtained attention weights are then reshaped into (b,(C1+C2),1,1) and applied to the original concatenated feature map via channel-wise scaling, yielding the weighted feature representation $\widehat{X_{concat}}$.

A cross-reweighting strategy is subsequently applied. The weighted feature map is split into two parts corresponding to the channel dimensions of the original input feature maps. These two parts are then multiplied channel-wise with the

original shallow and deep feature maps, respectively, to achieve cross-weighting. Finally, the fusion output is obtained via element-wise summation: $Y=X_{shallow}+\widehat{X_{deep}}$. Through adaptive channel weight adjustment, the weighted feature fusion module enhances the response of object regions while suppressing background noise and irrelevant features. As a result, the stability of grasp pose estimation is significantly improved.

3.3 Perception-driven grasp planning strategy

3.3.1 Analysis of decision factors

In multi-object grasping scenarios, decision-making must simultaneously account for grasp feasibility and stability. To this end, two core evaluation metrics are selected: the depth distance D and the distance between the object's center of mass and the grasp point G. A smaller depth distance indicates that the object is closer to the camera, which generally corresponds to reduced grasping difficulty, lower collision risk, and higher execution priority. In addition, when the grasp point is located closer to the object's center of mass, greater grasp stability is achieved, thereby reducing the likelihood of object rotation or slippage during manipulation [20].

3.3.2 Weighted scoring model

A multi-factor weighted scoring model is constructed to evaluate the grasping priority of each object. For object i, the overall score S_i is defined in Eq. (17), where α and β denote the weighting coefficients associated with the depth distance and the distance between the grasp point and the center of mass, respectively. The optimal values of α and β are determined through comparative experimental analysis.

$$S_i=\alpha\cdot D_i+\beta\cdot G_i \tag{17}$$

To compute the center of mass, the object is assumed to be composed of a homogeneous material, such that the center of mass coincides with the geometric centroid. The centroid coordinates (u,v) are estimated from the pixel distribution within the detected bounding box, as expressed in Eqs. (18) and (19), where N denotes the number of valid pixels within the bounding box.

$$u=\frac{\sum_{(x,y)\in box} x}{N} \tag{18}$$

$$v=\frac{\sum_{(x,y)\in box} y}{N} \tag{19}$$

Subsequently, distance computation is performed. The pixel coordinates of the geometric centroid are first transformed into three-dimensional coordinates (x_C,y_C,z_C) in the camera coordinate system. The Euclidean distance G_i between the centroid and the predicted grasp point (x_g,y_g,z_g), obtained from the grasp pose estimation network, is then calculated as follows:

$$G_i=\sqrt{\left(x_c-x_g\right)^2+\left(y_c-y_g\right)^2+\left(z_c-z_g\right)^2} \tag{20}$$

Finally, priority ranking is conducted. The overall score S_i is normalized and inverted to obtain the final grasping score S_i^*, defined in Eq. (21). A higher value of S_i^* indicates a higher grasping priority. The grasp execution sequence is determined by sorting objects in descending order of the scores.

$$S_i^*=1-\frac{S_i-S_{min}}{S_{max}-S_{min}} \tag{21}$$

4. MULTI-OBJECT GRASPING SIMULATION BASED ON THE ROBOT OPERATING SYSTEM

4.1 Construction of the simulation environment

4.1.1 Virtual grasping environment setup

The experimental framework was developed based on the robot operating system, a modular communication middleware architecture widely adopted in robotics. Initially proposed by the Stanford Artificial Intelligence Laboratory in 2007, the robot operating system has evolved through continuous development into a mature meta-operating system for robotic applications. By leveraging its loosely coupled distributed architecture and multi-language compatibility, a virtual simulation environment was constructed within the Gazebo three-dimensional physics engine. At the algorithm implementation level, deep learning frameworks, including PyTorch and TensorFlow, were employed for the deployment of object detection and grasp pose estimation models. In addition, the MoveIt! motion planning toolkit was integrated to enable trajectory optimization and real-time collision avoidance for the robotic manipulator.

A virtual environment was constructed based on the robot operating system framework and the Gazebo three-dimensional physics engine. The experimental system was deployed on the NVIDIA Jetson Nano Developer Kit edge computing platform. The YOLOv5s-r object detection model [20] was deployed in a lightweight manner using the TensorRT inference framework. A cross-platform model conversion strategy was adopted, in which the network weights trained in PyTorch were transformed into a standardized intermediate representation. Based on this representation, an optimized inference engine suitable for edge devices was constructed. The implementation process consisted of several key stages. First, an optimized and trained model was exported into a universal model file compliant with the Open Neural Network Exchange (ONNX) standard. Subsequently, architecture parsing and compilation-level optimization were performed using the intermediate representation. An accelerated engine tailored to the Nano edge computing device was then generated. The ONNX-formatted YOLOv5s-r model was further converted into a TensorRT engine compatible with the Jetson Nano and serialized into an ".engine" file. Through the TensorRT SDK, layer fusion and precision calibration interfaces were applied to convert the floating-point model into a hardware-executable binary instruction set. Finally, memory mapping techniques were utilized to store the optimized computational graph, resulting in an inference engine file embedded with hardware-specific instructions.

The experimental platform consisted of a six-degree-of-freedom UR5 collaborative robotic manipulator, a Robotiq-2F-85 gripper, and an Intel RealSense D455 depth camera. The UR5 robotic manipulator was characterized by a rated payload of 3 kg and a working radius of 500 mm. The Robotiq-2F-85 end-effector provided a gripper opening range of 0–85 mm and a grasping force ranging from 20 to 80 N. The Intel RealSense

D455 depth camera operated with a red, green, and blue resolution of 1280 × 800 at 30 frames per second and a depth resolution of 1280 × 720 at 90 frames per second. As illustrated in Figure 4, the camera is mounted at a height of 0.6 m above the operating plane. Synchronized color and depth data were captured to provide input for the perception task.

(a) (b) (c)

Figure 4. Simulation environment, (a) Simulated grasping environment, (b) Color image of target objects, (c) Depth image of target objects

4.1.2 Construction of the robot operating system communication framework

A distributed asynchronous communication architecture was established based on the publish–subscribe paradigm. The system consisted of five core nodes. In the visual acquisition node (Node 1), scene data are acquired by driving the sensor system and are broadcast via Topic 1. In the perception node (Node 2), object detection and grasp pose estimation networks are invoked to generate two-dimensional grasp parameters, which are transmitted through Topic 2. In the spatial transformation node (Node 3), coordinate transformation from the image coordinate system to the robot base coordinate system is performed. The resulting grasp pose and gripper parameters are published via Topic 3 and Topic 4, respectively. In the motion control node (Node 4), joint trajectories are computed and published to the robotic manipulator through Topic 5. In the gripper control node (Node 5), the end-effector is actuated, and corresponding control signals are transmitted via Topic 6.

4.1.3 Hand–eye calibration

An eye-to-hand configuration was adopted, as illustrated in Figure 5. In this setup, the camera was fixed above the workspace, while the robotic manipulator was controlled to move a calibration target to multiple poses within the field of view. The transformation matrix from the camera coordinate system to the robot base coordinate system was estimated using the visp_hand2eye_calibration package. The resulting transformation matrix is expressed as:

$$T_{cam}^{base}=\begin{bmatrix} 0.9999 & 0.0001 & 0.0003 & -0.0002 \\ 0 & -0.9999 & 0.0008 & 0.0003 \\ 0.0001 & 0 & -1 & 0.5998 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (22)$$

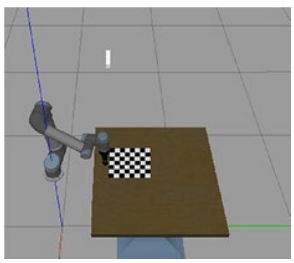
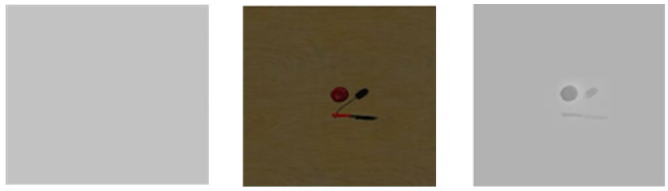


Figure 5. Hand–eye calibration configuration

4.2 Design of multi-object grasping simulation experiments

The simulation experiment was conducted according to the following procedure:

Step 1: Depth images of the empty scene, as well as red, green, and blue and depth images of the stacked objects, were acquired, as illustrated in Figure 6. The red, green, and blue images were then provided as input to the object detection network, through which the spatial locations and category information of the target objects were obtained, as shown in Figure 7.

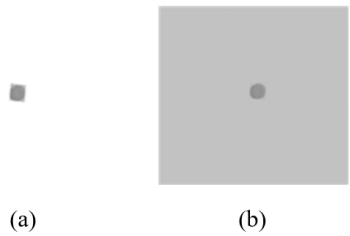


(a) (b) (c)

Figure 6. Camera-acquired information, (a) Depth image of the empty scene, (b) Color image of the scene, (c) Depth image of the scene

Figure 7. Output of the object detection network

Step 2: The depth image was cropped according to the predicted bounding boxes. The cropped depth regions were then merged with the empty-scene depth image to reconstruct a complete depth map, as illustrated in Figure 8. The resulting depth image was provided as input to the Inception–Weighted Feature Fusion–Generative Grasping Convolutional Neural Network framework, from which the two-dimensional grasp configuration was obtained.



(a) (b)

Figure 8. Input to the grasp pose network, (a) Cropped depth information, (b) Concatenated depth image

Step 3: The grasp decision-making algorithm was integrated with depth information, predicted bounding box locations, and grasp point information to compute grasp scores for each object. Based on the computed scores, the grasp execution sequence was determined, as illustrated in Figure 9.

Step 4: Based on the prioritized grasp sequence, coordinate transformation and three-dimensional grasp pose computation were performed. The robotic manipulator then executed

grasping operations sequentially according to the determined order until all target objects were successfully grasped.

(a) (b)

Figure 9. Output of the grasp decision-making algorithm, (a) Computation of object grasp scores, (b) Output of grasp execution sequence

4.3 Simulation results and analysis

Three representative scenarios were designed, including a five-object scattered configuration, a seven-object stacked configuration, and a ten-object stacked configuration, as illustrated in Figure 10. For each scenario, 50 independent trials were conducted. A trial is considered successful if the robotic manipulator executes grasping operations sequentially according to the prioritized grasp sequence without any object slippage or drop during the process. The experimental results are summarized in Table 1.

(a) (b) (c)

Figure 10. Multi-object simulation scenarios, (a) Five-object scenario, (b) Seven-object scenario, (c) Ten-object scenario

Table 1. Simulation results of multi-piston connecting rod grasping experiments

Test Scenario	Number of Trials	Successful Trials	Success Rate
Five-object scattered	50	50	100%
Seven-object stacked	50	46	92%
Ten-object stacked	50	43	86%

Based on the comparative analysis of Figure 10 and Table 1, it can be observed that the grasp success rate exhibits a negative correlation with scene complexity. In scenarios involving a smaller number of target objects and relatively simple spatial arrangements, distinct texture features and limited occlusion enable the visual perception system to achieve reliable region segmentation and accurate grasp pose estimation. Consequently, optimal grasping performance is obtained. However, as the number of target objects increases and the spatial configuration becomes more complex, multiple occlusion effects between objects significantly degrade detection accuracy. These perception errors directly affect the ability of the deep neural network to extract geometric features of grasp regions, leading to deviations in the computed grasp pose of the robotic manipulator end-effector. As a result, grasp failures may occur. Nevertheless, the overall simulation results demonstrate that the proposed multi-object grasp pose estimation method is capable of effectively satisfying the requirements of grasping tasks in multi-piston connecting rod scenarios.

5. CONCLUSION

To address the challenges of grasp pose estimation accuracy and execution stability in complex multi-object environments, an improved Inception–Weighted Feature Fusion–Generative Grasping Convolutional Neural Network framework and a perception-driven grasp planning strategy were proposed. An enhanced inception module and a weighted feature fusion module were designed. In addition, a weighted evaluation mechanism that jointly considers object depth and grasp stability was established, enabling rational prioritization in multi-object grasping tasks. A simulation system was constructed based on the robot operating system framework, and experimental validation was conducted under scenarios involving five, seven, and ten objects. An average grasp success rate of 92.7% was achieved, demonstrating the effectiveness of the proposed approach in complex multi-object environments.

Future work will focus on improving occlusion-robust detection in densely stacked scenarios and developing multimodal perception fusion models, with the objective of further enhancing adaptability in real-world industrial applications.

REFERENCES

[1] Soori, M., Arezoo, B., Dastres, R. (2023). Artificial intelligence, machine learning and deep learning in advanced robotics, a review. Cognitive Robotics, 3: 54-70. https://doi.org/10.1016/j.cogr.2023.04.001

[2] Jiang, Y., Moseson, S., Saxena, A. (2011). Efficient grasping from RGBD images: Learning using a new rectangle representation. In 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 3304-3311. https://doi.org/10.1109/ICRA.2011.5980145

[3] Lenz, I., Lee, H., Saxena, A. (2015). Deep learning for detecting robotic grasps. The International Journal of Robotics Research, 34(4-5): 705-724. https://doi.org/10.1177/0278364914549607

[4] Wang, D., Liu, C., Chang, F., Li, N., Li, G. (2021). High-

performance pixel-level grasp detection based on adaptive grasping and grasp-aware network. IEEE Transactions on Industrial Electronics, 69(11): 11611-11621. https://doi.org/10.1109/TIE.2021.3120474

[5] Morrison, D., Corke, P., Leitner, J. (2020). Learning robust, real-time, reactive robotic grasping. The International Journal of Robotics Research, 39(2-3): 183-201. https://doi.org/10.1177/0278364919859066

[6] Chu, F.J., Xu, R., Vela, P.A. (2018). Real-world multiobject, multigrasp detection. IEEE Robotics and Automation Letters, 3(4): 3355-3362. https://doi.org/10.1109/LRA.2018.2852777

[7] Jiang, D., Li, G., Sun, Y., Hu, J., Yun, J., Liu, Y. (2021). Manipulator grabbing position detection with information fusion of color image and depth image using deep learning. Journal of Ambient Intelligence and Humanized Computing, 12(12): 10809-10822. https://doi.org/10.1007/s12652-020-02843-w

[8] Guo, D., Kong, T., Sun, F., Liu, H. (2016). Object discovery and grasp detection with a shared convolutional neural network. In 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, pp. 2038-2043. https://doi.org/10.1109/ICRA.2016.7487351

[9] Mei, F., Gao, X., Deng, S., Li, W. (2022). Target recognition and grabbing positioning method based on convolutional neural network. Mathematical Problems in Engineering, 2022(1): 4360346. https://doi.org/10.1155/2022/4360346

[10] Zhang, G., Jia, S., Zeng, D., Zheng, Z. (2018). Object detection and grabbing based on machine vision for service robot. In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, pp. 89-94. https://doi.org/10.1109/IEMCON.2018.8615062

[11] Zhang, H., Liang, H., Ni, T., Huang, L., Yang, J. (2021). Research on multi-object sorting system based on deep learning. Sensors, 21(18): 6238. https://doi.org/10.3390/s21186238

[12] Kroemer, O., Niekum, S., Konidaris, G. (2021). A review of robot learning for manipulation: Challenges, representations, and algorithms. Journal of Machine Learning Research, 22(30): 1-82.

[13] Mousavian, A., Eppner, C., Fox, D. (2019). 6-dof GraspNet: Variational grasp generation for object manipulation. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), pp. 2901-2910. https://doi.org/10.1109/ICCV.2019.00299

[14] Arents, J., Greitans, M. (2022). Smart industrial robot control trends, challenges and opportunities within manufacturing. Applied Sciences, 12(2): 937. https://doi.org/10.3390/app12020937

[15] Horaud, R., Hansard, M., Evangelidis, G., Ménier, C. (2016). An overview of depth cameras and range scanners based on time-of-flight technologies. Machine Vision and Applications, 27(7): 1005-1020. https://doi.org/10.1007/s00138-016-0784-4

[16] Kytö, M., Nuutinen, M., Oittinen, P. (2011). Method for measuring stereo camera depth accuracy based on stereoscopic vision. In IS&T/SPIE Electronic Imaging, San Francisco Airport, California, United State, pp. 168-176. https://doi.org/10.1117/12.872015

[17] Li, L., Yang, X., Wang, R., Zhang, X. (2024). Automatic robot hand-eye calibration enabled by learning-based 3D vision. Journal of Intelligent & Robotic Systems, 110(3): 130. https://doi.org/10.1007/s10846-024-02166-4

[18] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 2818-2826. https://doi.org/10.1109/CVPR.2016.308

[19] Hu, J., Shen, L., Sun, G. (2018). Squeeze-and-excitation networks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, pp. 7132-7141. https://doi.org/10.1109/CVPR.2018.00745

[20] Hua, J., Guo, W. (2024). A grasping pose estimation method for multi-object stacking scenes. In 2024 8th Asian Conference on Artificial Intelligence Technology (ACAIT), Fuzhou, China, pp. 86-91. https://doi.org/10.1109/ACAIT63902.2024.11021731