RESEARCH ARTICLE

Enhancing Robotic Grasp Detection with a Novel Two-Stage Approach: From Conceptualization to Implementation

Zhe Chu[1], **Mengkai Hung**[2] and **Xiangyu Chen**[3,*]

[1] School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China
[2] School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
[3] School of Mechanics, Civil Engineering and Architecture, Northwestern Polytechnical University, Xi'an 710129, China

Abstract

This study introduces a novel two-stage approach for robotic grasp detection, addressing the challenges faced by end-to-end deep learning methodologies, particularly those based on convolutional neural networks (CNNs) that require extensive and often impractical datasets. Our method first leverages a particle swarm optimizer (PSO) as a candidate estimator, followed by CNN-based verification to identify the most probable grasp points. This approach represents a significant advancement in the field, achieving an impressive accuracy of 92.8% on the Cornell Grasp Dataset. This positions it among the leading methods while maintaining real-time operational capability. Furthermore, with minor modifications, our technique can predict multiple grasp points per object, offering diverse grasping strategies. This adaptability and high performance suggest substantial potential for practical applications in robotic systems, enhancing their efficiency and reliability in dynamic environments.

Keywords: Robotic grasp detection, Convolutional neural network, Two-stage cascaded system, Particle swarm optimizer.

Academic Editor:
Jinchao Chen

Submitted: 02 June 2024
Accepted: 13 July 2024
Published: 27 July 2024

Vol. 1, **No.** 1, 2024.
10.62762/TIUS.2024.777385

***Corresponding author:**
✉ Xiangyu Chen
chxy95@mail.nwpu.edu.cn

Citation
Chu, Z., Hung, M. & Chen, X. (2024). Enhancing Robotic Grasp Detection with a Novel Two-Stage Approach: From Conceptualization to Implementation. IECE Transactions on Intelligent Unmanned Systems, 1(1), 44–54.

1 Introduction

Robotic grasping is a fundamental function for intelligent robots to perform various autonomous manipulation tasks [1]. Humans can instinctively perceive unstructured environments, identify the characteristics of an object, and grasp it directly. However, robots lack this capability, and modeling the robot's surrounding environment is both time-consuming and challenging.

Recently, the development of deep learning has led to many end-to-end robotic grasp detection approaches based on convolutional neural networks (CNNs). Most of these new approaches are one-stage methods, which identify a good grasp in a single step using CNNs and achieve high accuracy on datasets. However, these end-to-end approaches [2, 3] rely heavily on the accuracy of the trained CNN models, which in turn depends significantly on the quality of the datasets. In practical applications, creating high-quality datasets is difficult. Therefore, to achieve better practical results, we need a robotic grasp detection approach that reduces the dependency on model accuracy.

Sliding window detection, a method often used in the past for robotic grasp detection, involves using a classifier to test selected parts of the image sequentially. The part with the highest score is considered the best grasp for the object [4]. Although this method is somewhat time-consuming, it can mitigate the impact of model inaccuracies on the final result by using techniques such as randomization and conditional constraints.

Building on sliding window detection and CNNs, we propose a two-stage robotic grasp detection approach that extracts the optimal grasp from an object's RGB image using a PSO candidate estimator and CNN. First, we develop a robotic grasp identification model based on a deep CNN to determine if the input represents a suitable grasp. Next, we identify promising candidate grasps using the candidate estimator, which employs a PSO algorithm to find appropriate candidates. This approach reduces the dependency on model accuracy. We evaluated our method on the Cornell Grasp Detection Dataset (see Fig. 1), achieving an accuracy of 92.8%.

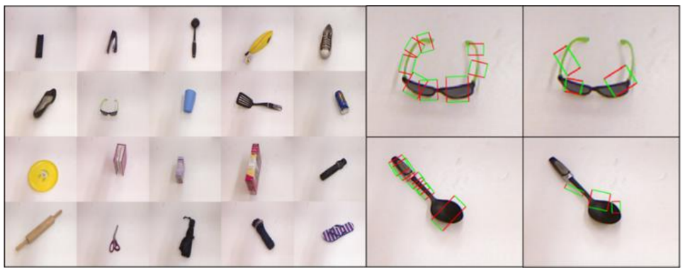


Figure 1. The Cornell Grasp Detection Dataset contains a diverse array of objects, each labeled with multiple graspable and ungraspable feature labels.

This paper is organized as follows: In Section 2, we discuss related work. Section 3 presents our identification models. In Section 4, we describe our candidate estimator. Section 5 details our experiment and evaluation. In Section 6, we present our results and compare them with other approaches. Section 7 describes our experiment on real robots. Finally, we provide our conclusions.

2 Related Work

In the past few decades, with the advancement of robotics [13–17] and artificial intelligence [18], numerous robotic grasp detection approaches have been developed. Most early approaches relied on accurate information about the environment and objects to identify good grasps. Based on this accurate information, some methods [5–8] have been used to achieve successful and stable grasps. However, a significant drawback of these methods is that robots often lack prior knowledge of the object's model, making it challenging to construct complex 3D models of the objects.

In recent years, with the development of deep learning, many researchers have proposed using deep learning for grasp detection. Here are some notable approaches:

Lenz proposed a two-stage approach based on sliding window detection [1]. Initially, candidate grasp rectangles are identified. These candidate grasp rectangles are then tested using a simple CNN to obtain a set of high-probability grasps. Finally, a more complex CNN ranks the grasps filtered out in the first stage to find the best grasp for the object.

Redmon et al. introduced a new CNN model by simplifying AlexNet [4]. This model directly regresses the grasp and classifies the grasp object. With further developments in deep learning, Kumra et al. proposed a novel CNN model for robotic grasp detection [3]. This model consists of two 50-layer CNNs and a shallow CNN. The outputs of the two 50-layer CNNs are merged and fed into the shallow CNN to predict the grasp configuration.

However, due to the time-consuming nature of sliding window detection, Wang et al. proposed a novel robotic grasp detection system [9]. This system reuses the results of the previous grasp detection for feedback to identify candidate grasps with higher probability. A CNN then evaluates these candidate grasps to determine the best grasp.

Among these approaches, Ian Lenz and Zhichao Wang's methods are two-stage but exhibit moderate accuracy (see Table 1). In contrast, Joseph Redmon and Sulabh Kumra's methods are end-to-end, requiring highly accurate datasets, which is challenging to achieve in practical use. Therefore, we decided to pursue a more effective two-stage approach.

3 Identification Model

We employ a CNN as the grasp identification model, initially training the network as a binary classifier to enable the model to learn to differentiate between grasping and non-grasping features. After the training process, the classification probability from the softmax layer is used as the model output. To reduce computing costs and increase computing speed, we utilized smaller networks and experimented with the following models.

3.1 Simplified AlexNet model

The foundational architecture of our model is inspired by the streamlined version of AlexNet introduced by Krizhevsky [10], a design that excelled in tackling target recognition challenges with notable accuracy. Despite sharing its origins with the AlexNet framework, our model introduces a unique configuration. As depicted in Figure 2, this model is composed of three convolutional layers, three max-pooling layers, and two fully connected layers. The final step involves the application of a softmax activation function to yield the output.

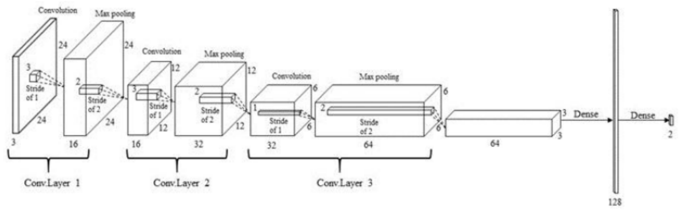


Figure 2. Simplified AlexNet model.

We trained the model, achieving a highest training accuracy of 83.39% and a highest validation accuracy of 82.12%, which are not ideal. To identify issues within the model, we employed feature map visualization. We outputted the feature maps after each layer (see Fig. 3). Observing the input image on the far left of Fig. 3, it is evident that the key features of the image are located in the middle. However, the critical feature learned by our model (highlighted in yellow) was positioned to the left of the image.

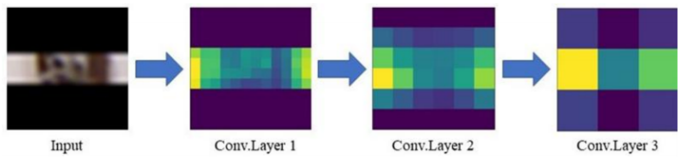


Figure 3. The result of Simplified AlexNet model's feature map visualization.

This indicates that our model only extracts some of the texture details, while failing to capture the critical features of the image. Furthermore, due to the presence of pooling layers, less image information is retained in the deeper layers.

3.2 Original GraspingNet model

Motivated by the shortcomings of the simplified AlexNet model, we opted for a more sophisticated architecture aimed at extracting more intricate and essential features. This decision led us to expand both the dimensions and quantities of the convolution kernels. As detailed in Figure 4, the convolutional layers in our updated model now feature kernels of sizes 3, 5, and 7, equipped with 32, 64, and 128 kernels, respectively. Notably, we discontinued the Maxpooling layer following each convolutional stage, thus preserving a greater amount of information. To expedite the learning process, we incorporated Batch Normalization immediately after each convolution operation and before applying the activation function. Post the two fully connected layers, we employed the softmax function to generate our output. For the activation function within the convolution layers, we chose the Rectified Linear Unit (ReLU).

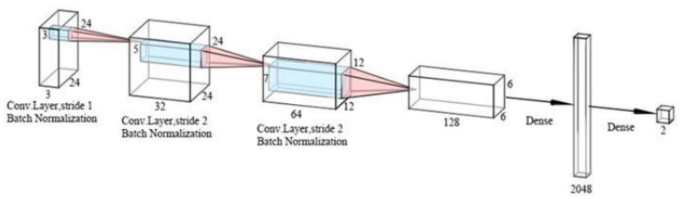


Figure 4. Original GraspingNet model.

After training, our model displayed a commendable training accuracy of 98.50%, yet a less satisfying validation accuracy of 85.50%, suggesting a pronounced issue of overfitting. Moreover, the computational demands of this model were notably higher, requiring four times the processing time compared to our initial model. Through the feature map visualization process (Fig. 5), we observed that the Original GraspingNet model continued to struggle with effectively extracting critical features. As depicted in Fig. 5, these essential features remained disproportionately located towards the left side of the images, indicating room for improvement in our model's feature extraction capabilities.

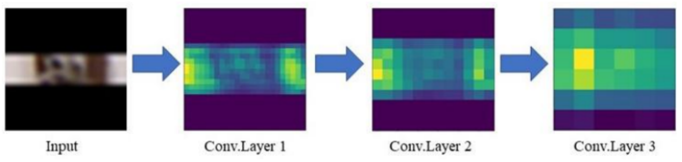


Figure 5. The result of Original GraspingNet model's feature map visualization.

3.3 Final GraspingNet model

Building upon the insights from our preceding models, we optimized our identification system by modifying a conventional CNN architecture. Typically, a CNN features convolutional layers, pooling layers, and fully connected layers. In contrast, our Final GraspingNet model eschews pooling layers and introduces Batch Normalization between the convolution and activation stages. Furthermore, to streamline the model and mitigate overfitting concerns, we replaced the resource-intensive fully connected layers with a Global Average Pooling (GAP) layer. This strategic alteration enhances the model's efficiency and classification accuracy, while also accelerating its processing speed. As a result, our final identification model comprises an 8-layer CNN, predominantly consisting of convolutional layers, a GAP layer, and an output layer (Fig. 6).

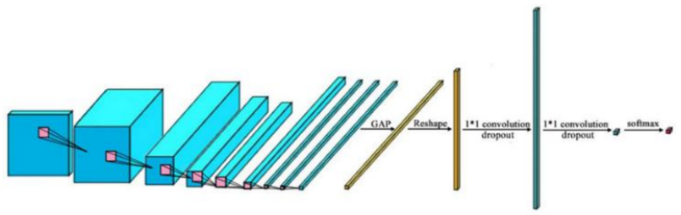


Figure 6. The architecture of the identification model. There are 8 convolution layers in the hidden layer. And in each convolution layer, we do convolution operation, Batch Normalization operation and Relu operation.

3.3.1 Convolution layer

In our proposed model, we have implemented significant modifications to the Original GraspingNet architecture. The primary alteration involves the replacement of large convolution kernels with more efficient 3x3 kernels. This strategic change not only enhances the model's feature extraction capabilities but also substantially improves its computational speed.

The revised architecture employs a sequential approach in each convolution layer. The data undergoes three consecutive operations: convolution, Batch Normalization, and ReLU activation. This sequence is maintained throughout the network before the processed data is fed into the subsequent convolution layer.

In the first convolution layer, we use 32 3x3 convolutional kernels with a stride of 1, and we ignore the edge pixels of the image by setting padding to "valid." After the convolution operation, Batch Normalization and ReLU activation functions are applied. The operation of the first convolution layer is defined as $\mathrm{Fc1}(\cdot)$, the input is defined as $\mathrm{Ic1}$, and the output is defined as $\mathrm{R_{c1}}$. The operation of the first convolution layer is as follows:

$$\mathrm{R_{c1}} = \mathrm{F_{c1}}\left(\mathrm{I_{c1}}\right) \tag{1}$$

This redesigned architecture, with its emphasis on smaller kernels and standardized layer operations, demonstrates a marked improvement in both feature extraction efficiency and overall model performance.

Following the initial layer, our model incorporates a series of three successive convolutional layers, each designed with identical structural parameters. These layers, denoted as the second, third, and fourth convolution layers, employ a uniform configuration of 64 3×3 convolution kernels. A stride of 2 is implemented across all three layers, effectively downsampling the feature maps and reducing spatial dimensions. This operation is defined as $\mathrm{Fc2}(\cdot)$, and the outputs of the second, third, and fourth layers are defined as $\mathrm{Rc2}$, $\mathrm{Rc3}$, and $\mathrm{Rc4}$, respectively. The operations for the second, third, and fourth convolution layers are as follows:

$$\mathrm{R_{c2}} = \mathrm{F_{c2}}\left(\mathrm{R_{c1}}\right) \tag{2}$$
$$\mathrm{R_{c3}} = \mathrm{F_{c2}}\left(\mathrm{R_{c2}}\right) \tag{3}$$
$$\mathrm{R_{c4}} = \mathrm{F_{c2}}\left(\mathrm{R_{c3}}\right) \tag{4}$$

In the last four convolution layers, we use 128 3x3 convolution kernels with a stride of 2. This operation is defined as $\mathrm{Fc3}(\cdot)$, and the outputs of these four convolution layers are defined as $\mathrm{Rc5}$, $\mathrm{Rc6}$, $\mathrm{Rc7}$, and $\mathrm{R_{c8}}$. The operations for these final four convolution layers are as follows:

$$\mathrm{R_{c5}} = \mathrm{F_{c3}}\left(\mathrm{R_{c4}}\right) \tag{5}$$
$$\mathrm{R_{c6}} = \mathrm{F_{c3}}\left(\mathrm{R_{c5}}\right) \tag{6}$$
$$\mathrm{R_{c7}} = \mathrm{F_{c3}}\left(\mathrm{R_{c6}}\right) \tag{7}$$
$$\mathrm{R_{c8}} = \mathrm{F_{c3}}\left(\mathrm{R_{c7}}\right) \tag{8}$$

3.3.2 GAP layer

The Global Average Pooling operation conducted by the GAP layer is defined as $\mathrm{FGAP}(\cdot)$, and its output is defined as RGAP. Thus, the operation can be expressed as follows:

$$\mathrm{R_{GAP}} = \mathrm{F_{GAP}}\left(\mathrm{R_{c8}}\right) \tag{9}$$

3.3.3 Output layer

Before the final output of the model, we first use 1x1 convolution to perform dimensionality reduction and expansion operations on the output of the GAP layer to enhance the combination of information across channels. Additionally, dropout is applied to further prevent overfitting before these dimensionality operations.

We define the dropout operation as $F_{\mathrm{D}}(\cdot)$, the 1x1 convolution for dimensional expansion as $F_A(\cdot)$, and the output of the dropout and dimensional expansion operations as $\mathrm{R_A}$. Thus, the operations can be expressed as follows:

$$\mathrm{R_A} = \mathrm{F_D}\left(\mathrm{F_A}\left(\mathrm{R_{GAP}}\right)\right) \tag{10}$$

We define the 1x1 convolution reduction operation as $F_R(\cdot)$, and the output of the dropout and dimension expansion operation as R_R. Thus, the operations can be expressed as follows:

$$\mathrm{R_R} = \mathrm{F_D}\left(\mathrm{F_R}\left(\mathrm{R_A}\right)\right) \tag{11}$$

Finally, we activate the output using the softmax function, which provides the model's identification results in probabilistic form. We define the softmax function operation as $\mathrm{F_S}(\cdot)$, and the output of the output layer as O. Thus, the operation can be expressed as follows:

$$\mathrm{O} = \mathrm{F}_S\left(\mathrm{R_R}\right) \tag{12}$$

3.3.4 Model training

The training process of a convolutional neural network model aims to find the global optimal solution in the parameter space. During this process, many local optima may be encountered, but it is crucial to bypass these to reach the global optima and train the best model.

To achieve this, we used the stochastic gradient descent method to train the CNN model and set the learning rate in stages. During training (see Fig. 7), the learning rate decayed every 60 epochs. This approach allows the model to skip some local optima during the initial faster training phase. As the model approaches the optimal solution, the learning rate decreases significantly, allowing it to converge to the optimal solution at a very low learning rate. Additionally, we increased the momentum value during training. This adjustment helps the model escape local optima and enhances the stability of the training process to some extent.

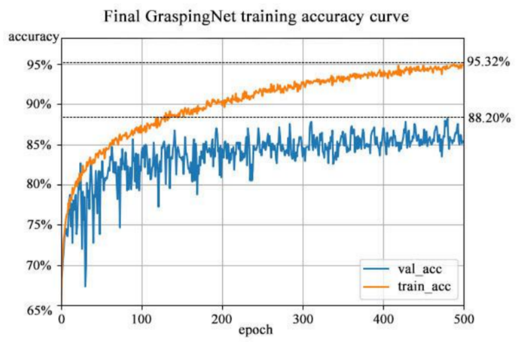


Figure 7. Final GraspingNet training curve.

4 Identification Model

4.1 Five-dimensional representation

There are primarily two types of grasp representations. The first is a seven-dimensional representation proposed by Yun Jiang [11], and the second is a five-dimensional representation proposed by Ian Lenz [1]. The five-dimensional representation is a simplified version of the seven-dimensional representation. For computational convenience, we use the five-dimensional representation. In this representation, the grasp is defined as a rectangle determined by its location, size, and direction, as follows:

$$\mathrm{g} = \{\mathrm{x}, \mathrm{y}, \theta, \mathrm{h}, \mathrm{w}\} \tag{13}$$

In this representation, (x, y) denotes the location of the rectangle, θ represents the direction of the rectangle relative to the horizontal axis, h indicates the length of the gripper, and w specifies the width of the gripper opening (see Fig. 8).

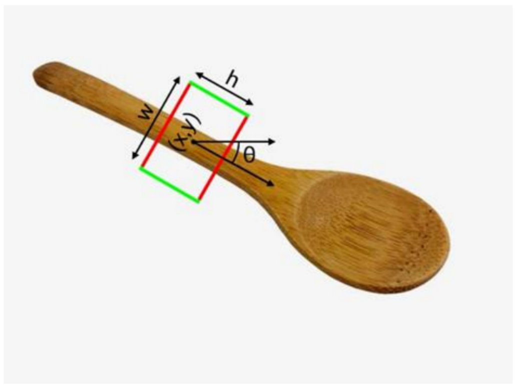


Figure 8. Five-dimensional grasp representation.

4.2 Optimizing Grasp Detection using Particle Swarm Optimization

In the context of sliding window detection, each image to be detected represents a vast and continuous state space, necessitating an efficient approach to search for the optimal grasp. A naive enumeration approach would require evaluating an enormous number of candidate grasps, resulting in a prohibitively high computational cost. For instance, considering an image of size (224×224), we would need to examine 50,176 locations. Furthermore, assuming 70 different gripper lengths, gripper opening widths ranging from 30 to

100, and gripper angles ranging from 0° to 180°in one-degree increments, the total number of candidates would reach an astronomical 632,217,600 ($50,176 \times 70 \times 180$).

To mitigate this issue, we propose a novel candidate estimator based on Particle Swarm Optimization (PSO), which transforms the grasp search into an optimization problem. Specifically, the parameters of this optimization problem are the five-dimensional representations of each candidate, with the processed image serving as the state space and image edges acting as constraints. Our objective is to identify a candidate that optimizes the grasp score function, which is defined as the probability of a successful grasp, as output by the grasp identification model described in Section 3. By applying PSO to solve this optimization problem, we can efficiently search for the optimal grasp, thereby reducing the computational cost.

4.3 PSO algorithm

Particle Swarm Optimization (PSO) is a stochastic optimization technique, first proposed by Eberhart and Kennedy in 1995, which draws inspiration from the collective behavior of biological populations, such as insects, herds, and birds, when foraging for food. This cooperative search process is characterized by individual members adapting their strategies based on their own experiences and those of others. In the context of optimization, each particle in the swarm independently searches for the optimal solution within the search space, marking its current individual extremum. The swarm then shares these individual extrema, identifying the optimal one as the current global optimal solution.

Through this process, each particle adjusts its velocity and position based on its individual extremum and the shared global optimal solution, enabling the algorithm to effectively navigate optimization problems involving multivariable functions with multiple local optima. By leveraging the collective intelligence of the swarm, PSO can efficiently explore the search space, converging towards the global optimum.

In this study, we use a particle xi to represent a candidate. Each particle has a velocity v_i^k at iteration k. The particle's movement is determined by its current individual extremum pbesti and the current global optimal solution gbest. The formulas for updating particle velocity and position are as follows:

$$\mathrm{v_i^{k+1} = wv_i^k + c_1r_1p_{best_i} + c_2r_2g_{best}} \tag{14}$$

$$\mathrm{x_i^{k+1} = x_i^k + v_i^{k+1}} \tag{15}$$

W is the inertia factor. $\mathrm{c_1}$ and $\mathrm{c_2}$ are accelerating factors. $\mathrm{r_1}$ and $\mathrm{r_2}$ are 2 random numbers, which are used for increasing the randomness of the search. The is shown in Algortithm 1.

Algorithm 1: PSO Algortithm.

```
g_fit =0, initial_time =0.
while (g_fit < init) and (initial_time < max_init) do
    Randomly initialize particle swarm $\mathrm{x_i, v_i}$;
    Use identification model to caculate score for $\mathrm{x_i, v_i}$;
    Update $\mathrm{p\_fit_i}$, g_fit;
    initial_time = initial_time +1;
end
while (g_best < prob) and (iteration < max_iter) do
    $\mathrm{v_i^{k+1} = wv_i^k + c_1r_1p_{best_i} + c_2r_2g_{best}}$;
    $\mathrm{x}_i^{k+1} = \mathrm{x}_i^k + \mathrm{v}_i^{k+1}$
    Use identification model to caculate score for $\mathrm{x_i, v_i}$;
    Update $\mathrm{p\_fit_i, g\_fit}$;
    Update g_best_i, g_best;
    iteration = iteration +1;
end
Output the best candidate grasp.
```

In Algorithm 1, g_fit represents the highest score among all particles, and p_fit represents the highest score for each individual particle. initial_time is the number of iterations during initialization, and iteration is the number of searches. max_init is the maximum number of initializations, and max_iter is the maximum number of searches. init is the minimum threshold that g_fit must meet during initialization, and prob is the minimum threshold that g_fit must meet during searching. This candidate estimator allows us to quickly find a good grasp.

The iterative convergence process of particles is illustrated in Figure 9. Initially, particles are randomly distributed within the image. As the algorithm iterates, the particles gradually converge towards the optimal location. Notably, particles that exceed the image boundaries during initialization or iteration are eliminated and replaced with new randomly generated particles. The convergence condition is met when either the maximum number of iterations is reached or the global optimal particle adaptation value reaches the specified threshold.

By making a small adjustment to the candidate estimator, it can output particles whose scores exceed

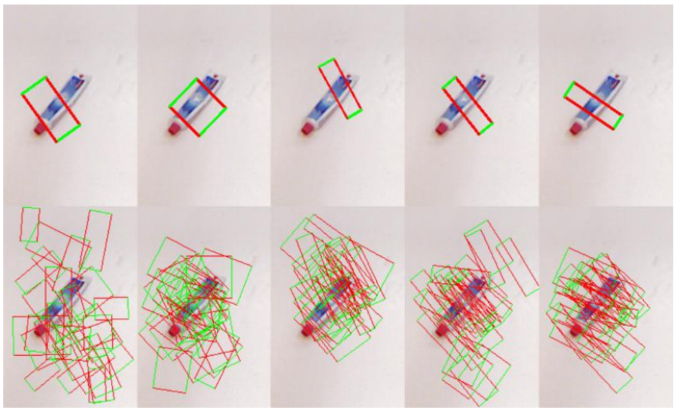


Figure 9. PSO iteration process. The first behavior of the image is the current global best position, and the second behavior is the current position of all particles.

a certain threshold and are ranked high among all particles. This modification allows us to identify multiple grasp points simultaneously.

4.4 Improvements to PSO

To ensure the PSO algorithm converges faster, we impose certain constraints on the particles during initialization and iteration. Firstly, we aim to distribute particles as close to the object as possible. Therefore, when initializing particles, we position them within the center of the image, ensuring that at least one particle has a score greater than 0.7. Secondly, before initializing the particles, we generate a histogram of the gray image corresponding to the RGB image. This histogram helps us roughly estimate the size of the target object. Based on this estimate, we initialize the particles differently according to the target object's size. This ensures the particles are appropriately sized, reducing the effect of improper particle size on particle score, thereby accelerating the algorithm's convergence. Lastly, we constrain the size, aspect ratio, and area of the particles to a specific range. If a particle falls outside these limits during initialization or iteration, we adjust the particle size back to a reasonable range by multiplying it with a corresponding correction factor.

5 Experiment and Evaluation

5.1 The cornell grasp detection dataset

The dataset comprises 885 images featuring 400 objects across 240 different categories, along with their corresponding grasp labels. This dataset is specifically designed for parallel grippers. Each image includes multiple grasp labels that are comprehensive and varied in terms of orientation, location, and scale. However, it does not encompass all possible grasps. Additionally, there are some errors in the labeling of both positive and negative samples. Despite these shortcomings, the dataset provides excellent examples of grasps. Therefore, we have chosen to analyze and evaluate our approach based on this dataset.

5.2 Image preprocessing

Before inputting the image into the identification model, we preprocess it by cropping the original image to focus on the center of the object, maintaining a size of 300×300 pixels. To ensure comparability with previous work, we then scale the image down to 224×224 pixels.

For the grasp identification model, we horizontally align the rectangular image along its long axis, fill the top and bottom ends with 0 pixels, and scale the image to 24×24 pixels before inputting it into the model. Prior to training, we perform necessary augmentations on the training set, including translation, scaling, and rotation.

5.3 The grasp identification model

Before inputting the feature image into the model, we preprocess it by scaling it to 24×24 pixels and filling it accordingly. This allows us to input the preprocessed feature image directly into the model.

Our identification model outputs in the form of probabilities, specifically the classification probabilities from the softmax layer regarding the graspable features (see Fig. 10).

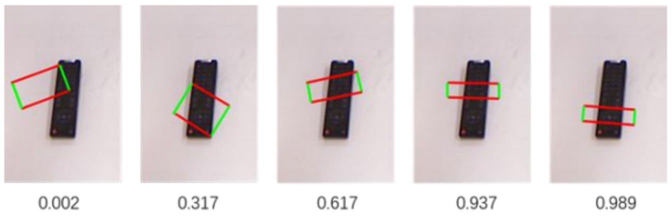


Figure 10. The output of the identification model.

If we use direct classification, as seen in the third example in Fig. 10, the model will classify it as graspable because the output is 0.617, which is greater than 0.5. Since the model only outputs probabilities, this does not impact subsequent iterations. Therefore, we do not directly evaluate models based on their accuracy alone; instead, the probability output serves as a reference for selecting the identification model.

5.4 Multigrasp detection approach

We have made a slight modification to the original method. Instead of outputting only the highest-scoring grasp, we now output several of the highest-scoring grasps simultaneously. These grasps converge on different positions of the target object, providing multiple ways to grasp the same object.

5.5 Grasp evaluation

When evaluating on the Cornell Grasp Detection Dataset, two different evaluation criteria have been used. The first approach is point evaluation, which determines the success of the predicted grasp by judging whether the distance between the center of the predicted grasp point and the center of the labeled point is below a certain threshold. This evaluation criterion has been widely discussed. The main issue with this criterion is that it does not consider the size and angle of the grasp. Additionally, previous work has seldom disclosed the thresholds used for evaluation comparisons, making it difficult to compare results. Therefore, we do not use this evaluation criterion.

The second approach is rectangle evaluation, which considers the entire grasp rectangle. According to this criterion, a grasp is deemed correct if it meets the following two conditions:

The angle between the predicted grasp and the labeled grasp does not exceed 30°.

The intersection ratio between the predicted grasp and the labeled grasp is at least 20

In our work, we use the second approach to evaluate the model. Although this method is more appropriate for evaluation, it still has some limitations. For instance, some predicted grasps may not meet the above two criteria but are still actually feasible grasps (see Fig. 11).

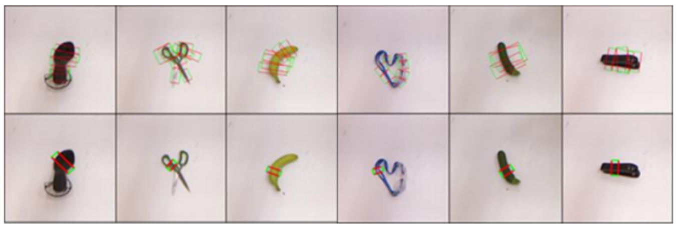


Figure 11. The first line is the grasps labeled by the dataset, and the second line is the result of our detection. Take the results of column 1 and column 4 from the left in the picture as an example. Although they do not meet the requirements of rectangle evaluation, it can still be considered to be graspable.

6 Results and Comparisons

We tested our model on a computer with a single CPU (i9-9900k 3.6GHz), a single GPU (NVIDIA 1080Ti), and 16 gigabytes of memory. According to the evaluation criteria outlined in Part III, our model achieved excellent results on the Cornell dataset.

6.1 Comparison of different approaches

We compared the accuracy of our method with the results of previous experiments on the Cornell dataset (as shown in Table 1). Our approach achieved a success rate of 92.8%. While our success rate is only slightly lower than that achieved by Guo et al. [12], it represents a significant improvement over existing two-stage methods, increasing the success rate by 11 percentage points. In terms of speed, our algorithm

Table 1. We compare our approach with the previous work.

Algorithm	Accuracy(%)
Jiang et al. [11]	60.5%
Lenz et al. [1]	73.9%
Wang et al. [9]	81.8%
Redmon et al. [4]	88.0%
Asif et al. [2]	88.2%
Kumra et al. [3]	89.2%
Guo et al. [12]	93.2%
Ours	**92.8%**

requires 378 ms to process an image. Although our method is still slower than one-stage methods, it is capable of running in real-time.

6.2 The result of multigrasp

Previously, we mentioned that with a small modification to our method, we could obtain multiple grasp points at once, providing several solutions for grasping an object. We conducted relevant experiments to validate this, and the results are shown in Fig. 12. The figure demonstrates that the multigrasp method provides multiple correct grasps for each target object. We also compiled statistics on the success rate and time consumption (see Table 2). In the evaluation of multigrasp detection, if at least one predicted grasp candidate of an image meets the requirements, the detection of grasp points for that image is considered successful. The results indicate that the multigrasp method has a higher success rate compared to the original method, with only a slight increase in average time consumption by 5 ms.

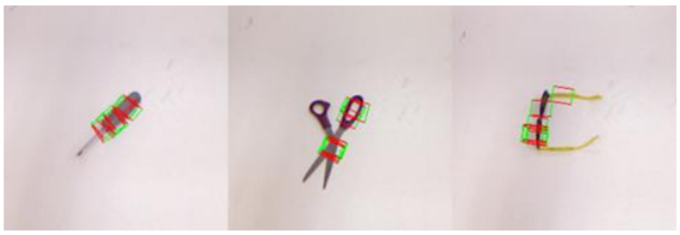


Figure 12. The result of the multigrasp approach.

Table 2. Comparison between mutligrasp method and original method.

	Accuracy(%)	Average time (ms)
Original	92.8%	378 ms
Mutligrasp	94.8%	383 ms

6.3 Summary

We demonstrate that deep learning models can effectively learn grasping features and that the PSO optimization algorithm can successfully address the issue of multiple local optima. By using a straightforward network, we significantly reduce computational costs, making the model easy to train and deploy. We transform the detection problem into an optimization problem, thereby reducing dependency on labeled data and allowing for multiple possibilities in the entire state space. The PSO algorithm excels in handling multivariable complex function optimization problems with multiple local optima, offering low computational complexity, fast convergence, and the ability to find better solutions in a reasonable amount of time. By combining deep network learning characteristics with the PSO optimization algorithm for grasp detection, we achieve advanced results efficiently.

7 The Experiment on the Robot

7.1 The introduction of the experiment

After achieving successful results on the lab computer, we decided to test our approach on a real robot. The process is as follows (see Fig. 13): First, we selected six objects that were not present in the dataset. Next, we used the watershed algorithm to process the original images captured by the robot's camera, extracting the target objects and placing them in the same position on a light white background image. Finally, we applied our method to detect the grasp points on the images obtained in the previous step and drew the grasp regions on the original images.

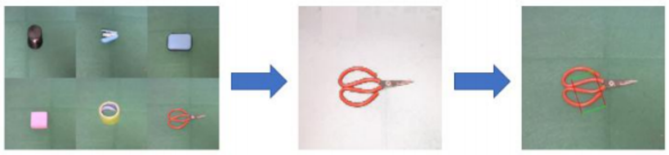


Figure 13. The process of experiment on the robot.

7.2 Result and discussion

We applied our method to detect the grasp points on the robot and obtained the following results (see Fig. 14). While there are no significant differences in the types of objects we selected compared to those in the Cornell dataset, there are notable differences in detail. However, these differences did not significantly impact our detection results. Among the six selected objects, the predicted grasps were correct for all except the pink plastic blocks in the second row and the first column. This demonstrates that our detection approach can be effectively used on the robot and achieve good results with unfamiliar objects, provided that the objects in the dataset belong to the same category as the unfamiliar ones.

In practical applications, we can accurately label a few typical objects within the same category, which will help achieve effective grasp detection for most objects in that category. This approach reduces the number of objects required in the dataset, facilitating the application of our method in real-world detection scenarios.

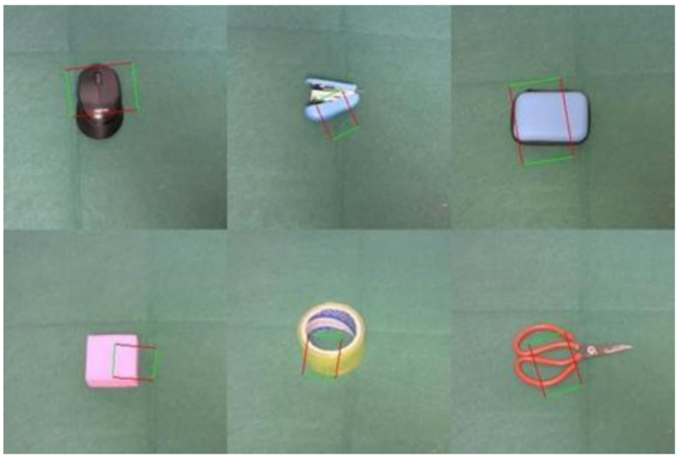


Figure 14. The result of the single grasp detection.

8 Conclusion

In this article, we present a robotic grasp detection approach that combines deep learning with the PSO algorithm. We transform the detection problem into an optimization problem. Initially, a CNN is employed to learn graspable features. Subsequently, the identification model is treated as the objective function for optimization, the entire image is considered the state space, and the parameters of the rectangle identification points are used as variables. The PSO algorithm is then applied to solve the optimization problem. Our algorithm achieves top-tier accuracy on the Cornell Grasp Detection Dataset. Additionally, our approach is two-stage, which, compared to end-to-end methods, requires less accuracy from the model and dataset, enhancing its practical usability.

For future work, we aim to improve the speed and robustness of the algorithm to achieve better results in real industrial deployments.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgement

This work was supported without any funding.

References

[1] Lenz, I., Lee, H., & Saxena, A. (2015). Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5), 705-724. [CrossRef]

[2] Asif, U., Bennamoun, M., & Sohel, F. A. (2017). RGB-D object recognition and grasp detection using hierarchical cascaded forests. *IEEE Transactions on Robotics*, 33(3), 547-564. [CrossRef]

[3] Kumra, S., & Kanan, C. (2017, September). Robotic grasp detection using deep convolutional neural networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 769-776). [CrossRef]

[4] Redmon, J., & Angelova, A. (2015, May). Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1316-1322).[CrossRef]

[5] Bicchi, A., & Kumar, V. (2000, April). Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)* (Vol. 1, pp. 348-353). [CrossRef]

[6] Miller, A. T., & Allen, P. K. (2004). Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4), 110-122.[CrossRef]

[7] Miller, A. T., Knoop, S., Christensen, H. I., & Allen, P. K. (2003, September). Automatic grasp planning using shape primitives. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)* (Vol. 2, pp. 1824-1829). [CrossRef]

[8] Pelossof, R., Miller, A., Allen, P., & Jebara, T. (2004, April). An SVM learning approach to robotic grasping. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004* (Vol. 4, pp. 3512-3518). [CrossRef]

[9] Wang, Z., Li, Z., Wang, B., & Liu, H. (2016). Robot grasp detection using multimodal deep convolutional neural networks. *Advances in Mechanical Engineering*, 8(9), 1687814016668077. [CrossRef]

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097-1105. [CrossRef]

[11] Jiang, Y., Moseson, S., & Saxena, A. (2011, May). Efficient grasping from rgbd images: Learning using a new rectangle representation. In *2011 IEEE International conference on robotics and automation* (pp. 3304-3311). [CrossRef]

[12] Guo, D., Sun, F., Liu, H., Kong, T., Fang, B., & Xi, N. (2017, May). A hybrid deep architecture for robotic grasp detection. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1609-1614).[CrossRef]

[13] Wang, N., Fang, F., & Feng, M. (2014, May). Multi-objective optimal analysis of comfort and energy management for intelligent buildings. In *The 26th Chinese control and decision conference (2014 CCDC)* (pp. 2783-2788). IEEE.[CrossRef]

[14] Fang, F., & Wu, X. (2020). A win–win mode: The complementary and coexistence of 5G networks and edge computing. *IEEE Internet of Things Journal*, 8(6), 3983-4003.[CrossRef]

[15] Lv, Y., Fang, F. A. N. G., Yang, T., & Romero, C. E. (2020). An early fault detection method for induced draft fans based on MSET with informative memory matrix selection. *ISA transactions*, 102, 325-334.[CrossRef]

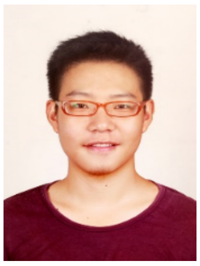
[16] Fang, F., Jizhen, L., & Wen, T. (2004). Nonlinear internal model control for the boiler-turbine coordinate systems of power unit. *PROCEEDINGS-CHINESE SOCIETY OF ELECTRICAL ENGINEERING*, 24(4), 195-199.

[17] Fang, F. A. N. G., Tan, W., & Liu, J. Z. (2005). Tuning of coordinated controllers for boiler-turbine units. *Acta Automatica Sinica*, 31(2), 291-296.

[18] Wang, Y., Deng, J., Fang, Y., Li, H., & Li, X. (2017). Resilience-aware frequency tuning for neural-network-based approximate computing chips. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(10), 2736-2748. [CrossRef]

Zhe Chu is an undergraduate at Northwestern Polytechnical University. He came to Northwestern Polytechnical University in 2017. He is a member of the soccer robot base at Northwestern Polytechnical University. He is interested in robotics,machine learning and computer vision.

Mengkai Hu received the bachelor's degree in electronics and information engineering from Northwestern Polytechnical University in 2017. And he recieved his master's degree from Peking University in 2020. His research focus on signal processing.



Xiangyu Chen received the B.E. degree and the M.E. degree from Northwestern Polytechnical University, Xi'an, China, in 2017 and 2020. Currently he is a research assistant in Shenzhen Institutes of Advanced Technology, Chinese Academy of Science. His research focus on robotics, deep learning and computer vision.