Development of Annulus-Object Random Bin Picking System based on Rapid Establishment of RGB-D Images

BO-RUI ZHU[1], JIN-SIANG SHAW[1], SHIH-HAO LEE[2]
[1]Institute of Mechatronic Engineering,
National Taipei University of Technology,
Taipei,
TAIWAN

[2]Institute of Manufacturing Technology
National Taipei University of Technology
Taipei,
TAIWAN

Abstract: - With the development of the automation industry, robotic arms, and vision applications are no longer limited to fixed actions of the past. Production lines increasingly require the recognition and grasping of objects in complex environments, emphasizing quick setup and stability. In this paper, a rapidly constructed eye-hand system for robotic arm grasping, which enables fast and efficient object manipulation, particularly for stacking, is introduced. Initially, images were captured using a camera to generate extensive datasets from a limited number of images. Objects were subsequently segmented and categorized using deep learning networks for object detection and instance segmentation. Three-dimensional position information was obtained from an RGB-D camera. Finally, object poses were determined based on plane normal vectors, and gripping positions were manually marked. This reduced the time required for grab-point identification, model training, and pose localization. Based on experimental results, the grasping procedure proposed in this paper is suitable for various object-grasping scenarios. It achieved impressive picking success rates of 96% for unstacked annular objects and 90.86% for random bin annular objects, respectively. In the final experiment, following depth information filtering, a success rate of 95.1% was attained with random bin annular object picking.

Key-Words: - Robot Manipulator, Random Bin Picking, Deep Learning, Instance segmentation, Grasping Strategies, Collaborative Robotics.

Received: May 21, 2023. Revised: December 22, 2023. Accepted: January 16, 2023. Published: February 28, 2024.

1 Introduction

Object grasping is an important research topic in the field of robotics. Random sorting (RBP) is an important research direction for robotic arm grasping, and it generally needs to be applied in the industry. Although this action is simple for humans, it is challenging for a robotic arm. Owing to the random stacking of objects, many occlusions exist in the box, and several problems, such as collisions between the robotic arm, object, and box, increase the difficulty of clamping. However, with recent advances in machine vision and artificial intelligence, this type of research has become increasingly reliable for practical applications. Many companies have actively invested in this development.

Among the traditional methods, the triangulation laser measurement method was employed to generate two distance images under a dual laser system, synthesized unobstructed distance images within the two distances and used each beam of light reaching the surface of the object via the laser transmitter. The geometric relationship between the points was used to obtain the three-dimensional (3D) information of the point cloud on the surface of the object. The point cloud information was combined with the sample consensus (SAC) and iterative closest point (ICP) algorithms to transform the object point cloud. This information was matched with a computer-aided design (CAD) model of the object to assess its pose and determine the grip points, [1]. Other researchers utilized a CAD model and a depth image from an RGB-D camera, converting them into a scene point cloud, [2]. A growing number of recent algorithms use deep learning convolutional neural networks (CNN) for sampling and evaluation, such as that proposed by Pinto and Gupta. Through evaluation, they obtained the optimal clamping position and clamping posture, [3]. Various contact mechanics models, including those introduced in previous studies, are also available for evaluation, [4]. Similarly, used a 3D model to reconstruct a 3D scan from multiple views and simulated an object model to obtain a synthetic dataset. Finally, the object was cut out using the R-CNN to obtain the object point cloud information. The algorithm calculated the gripping position, [5]. The above goals were to obtain correct identification and clamping stability and to achieve a high clamping success rate. However, these methods require the creation of a large number of images and labeled data; therefore, they are relatively time-consuming and costly. However, in certain applications, setting up the clamping process quickly is desirable.

In this paper, an object-grabbing program that can be built quickly is introduced. It was expected to grab annulus-shaped objects, commonly found in factories; however, tape was used to test the reusability of subsequent objects. First, the clamping position of the object was preset, and the center point of the object was expected to be the clamping position. The desired object was cropped from the RGB image using deep learning, and the 3D position of the object was obtained through algorithms and camera depth information. Subsequently, the posture and position of the object relative to the camera were obtained from the 3D information of the object, and the information obtained by the camera was converted into the information required by the robotic arm, such that the robotic arm can obtain the grasping position the object and the grasping posture of the arm. Objects can also be captured.

We accelerated the data collection and data labeling time for deep learning and did not use methods, such as generative grasp convolutional neural networks (GGCNN) for grasp visualization, [6] or [7], where a hybrid deep structure of visual and tactile sensing using a reference rectangle method to identify graspable regions of an image was proposed. Various methods, used the powerful learning ability of large convolutional networks to predict the global grasp of a complete image of an object, [8]. ResNet50 was used for feature extraction, [9]. Although the accuracy of ResNet50 is good, its speed is low. We chose a lighter Inception model architecture and only used depth information to compute the grasped objects. This resulted in a fast and low-cost grasping system.

2 System Description

The system, hardware, and software architectures are introduced in this section.

2.1 System Architecture

Figure 1 shows the architecture of the robotic arm system. It is divided into the following parts: preprocessing, hardware calibration, image recognition, information integration, and communication. First, we set the object to be grasped and its grasping point. In this study, the center point of the tape was spread out with jaws for gripping. The hardware was then corrected, including image correction processing for the RGB-D camera and hand-eye correction for the robotic arm and camera. Next, deep learning was used, which included fast data collection and annotation. Finally, RGB-D cameras were combined to complete object recognition.

In the follow-up, we used a calculation method designed to integrate and calculate the RGB-D 3D information obtained from image recognition to determine the grasping posture of the robotic arm. Finally, the entire grasping process was completed through communication between the PC and the robotic arm.

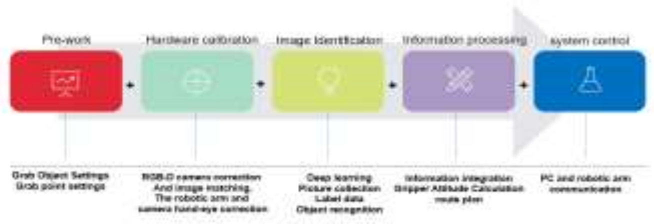

Fig. 1: System architecture

2.2 Hardware Architecture

The main hardware used in this study included a robotic arm (Kuka iiwa 7 R800), an RGB-D camera (Kinect V2), and a controller (TOYO CHS2-S40), as shown in Figure 2. The robotic arm has seven degrees of freedom and exhibits high flexibility, fast moving speed, and high efficiency. The controller was mounted on the flange of the robot arm and controller using a self-designed fixture and gripper, respectively. The robotic arm and RGB-D camera had an eye-to-hand architecture.

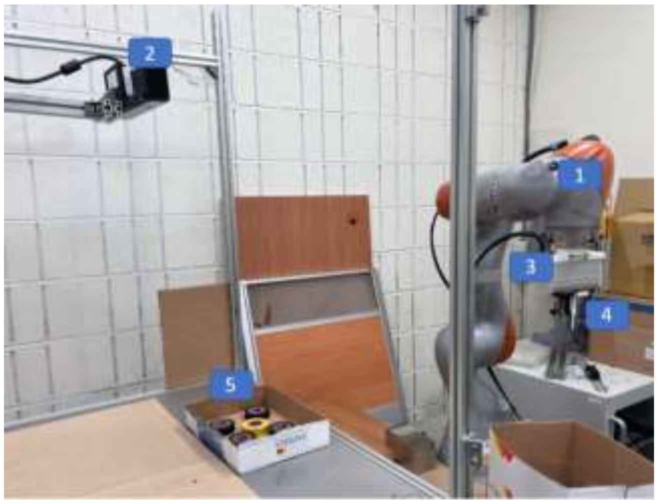

Fig. 2: Robotic painting system: (1) robotic arm, (2) RGB-D camera, (3) controller, (4) gripper, (5) grab object

2.3 Software Architecture

In this study, multiple hardware and software techniques were combined. Depending on technical requirements, selecting an appropriate development platform and communication method is important. Python 3.0 was used for deep learning, image recognition, and data processing. Java was used to compile the robotic arm program. The RGB-D camera and computer only used USB 3.0 for connection, and the robot arm and gripper used EtherCAT, which is an easy-to-configure automation system, as a communication protocol.

To enable system communication, integrating the communication between the controller, sensor, and actuator software is important. In this study, the host system consisted of a personal computer (client) and a robot (server). The personal computer was the client, and the results of the deep learning and image recognition were converted by calculation and sent to a robotic arm on the server. Figure 3 shows the communication architecture of the system.

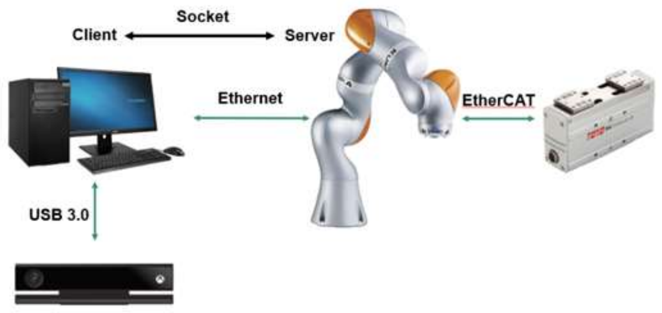

Fig. 3: System communication

3 Methodology

The research methodology is described in this section in four parts: system integration, deep learning, grasping position calculation, and robotic grasping control.

3.1 System Integration

3.1.1 Image Correction

Since RGB-D cameras have image distortion problems, we referred to the correction method proposed in [10], and then used the Zhang correction method proposed at the International Conference on Computer Vision (ICCV). This method uses a plane checkerboard for camera calibration, [11], overcomes the shortcomings of the high-precision 3D correction required by the photographic correction method, and solves the problem of poor robustness of the self-correction method. In this study, the self-printing checkerboard was adopted for calibration for convenience, and 20 checkerboard photographs were captured from different directions as shown in Figure 4.

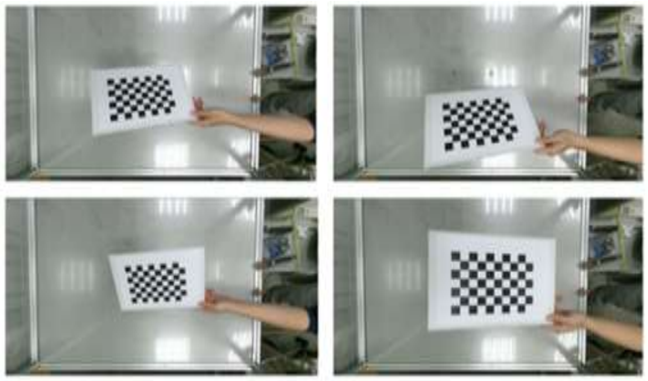

Fig. 4: Photographs of the checkerboard captured from different directions

3.1.2 RGB-D Image Matching

Since the color and depth lenses of the Kinect V2 are not in the same position, the color image pixel size was 1920 × 1080, whereas the depth image pixel size was 512 × 424. Therefore, matching the color and depth images was necessary. The same point in the world coordinates was observed using the color and depth lenses, as shown in Figure 5. The calibration results are presented in Figure 6.

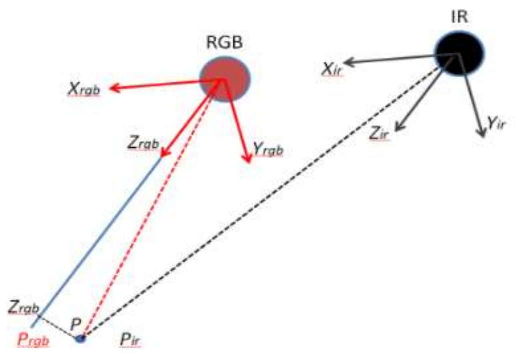

Fig. 5. Schematic of the same point observed by color and depth lens

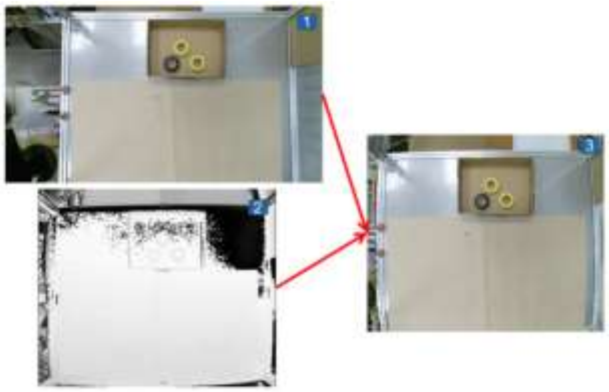

Fig. 6. (1) Color image, (2) Depth image, (3) Image matching resul

3.1.3 Hand-eye Correction of Robotic Arm and Camera

In the vision of robotic arms, one of the key issues is hand-eye correction, which is divided into eye-to-hand and eye-in-hand corrections according to the definition of its position fixed by the camera. In this study, an eye-to-hand system architecture was adopted, that is, the camera was installed outside the robotic arm, and the bases of the camera and robotic arm were fixed in a position. We considered the quick solution to the classic problem of hand-eye correction. Many researchers, [12], [13], [14], [15] have studied the calculation of Formula (1). Various solutions exist to this problem. Since the Open Source Computer Vision Library (OpenCV) includes related open-source libraries that are convenient for quick use and calculation, we used the calculation solution proposed in [12], to obtain the relative position of the robotic arm and camera.

$$AX = XB \tag{1}$$

3.2 Deep Learning

3.2.1 Data collection

Owing to the need for subsequent deep learning, collecting object image data for training deep learning networks is necessary to identify objects. Here, a Kinect V2 was used to capture photographs, which were used as a dataset. We randomly placed a large number of power-supply tapes on the workbench, as shown in Figure 7(1), to create a random stacking environment. To quickly label and meet the requirements of this study, after each data map was obtained, several objects were removed to obtain a new data map, as shown in Figure 7(2) for the follow-up data. Figure 7(1) shows a data graph comparison of removed objects. Although this method only reduces the objects for humans, it provides a new data map to the system. This method was used 11 times in this study to rapidly increase the amount of data, using different random stacking environments. Thus, the number of objects in the stacking environment was reduced individually to generate a total of 122 images as the deep learning network training dataset.

The collected 122 images as a dataset were still slightly insufficient for deep learning networks. To train image recognition with good accuracy and generalization, in addition to the structure of the model, the most important aspect is the number of datasets. The larger the amount of data, the more complete it is; the more complex the structure, the better the training results. Generally, the collection of image data is often the most time-consuming process, and in the case of limited data, the method used in this study is the simplest and most widely used method: augmentation. Data augmentation can improve the effectiveness of deep-learning object detection, [16]. We randomly enhanced the original images with noise reduction, flipping at different angles, and brightness processing, as shown in Figure 8. After data enhancement, 976 images were added, including 1098 original images that were used for subsequent deep learning.

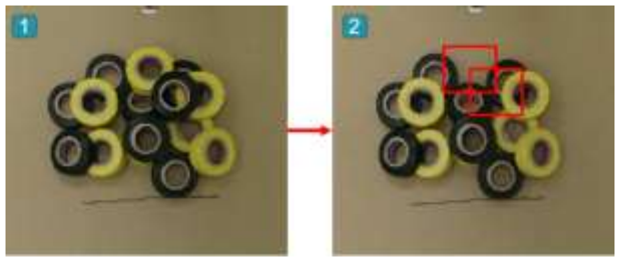

Fig. 7: (1) Stacked environment data graph, (2) New data graph with objects removed

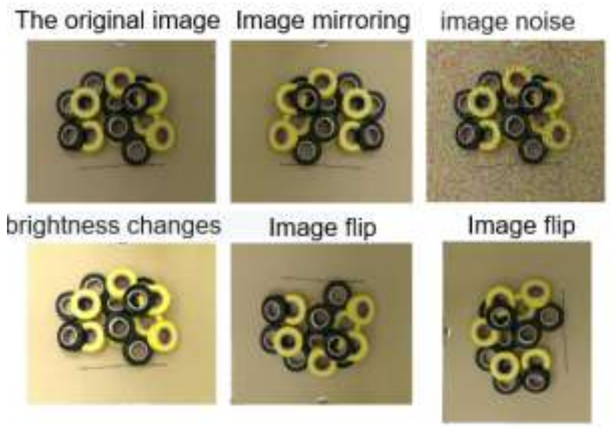

Fig. 8: Image data enhancement

3.2.2 Image Data Annotation

Before deep learning, apart from data collection, image annotation is the most time-consuming task. Humans can easily perceive the location and type of objects. However, before entering deep learning, first, the objects must be circled, and the machines need to be trained on the object types. This process is called data annotation. To expedite object labeling, the aforementioned data collection method can be applied for expediting image labeling. The same method can be used to copy the frame selection object frame, on the next image data, delete the frame selection frame of the removed object, or select a similar frame selection frame. It can be directly applied to a new data map to expedite its labeling.

To improve the success rate of gripping by the robotic arm and avoid gripping collisions, the training objects could only be recognized when the occluded range was less than 20%. This setting improves the object recognition rate and significantly reduces the error rate and number of annotations required for the image, as shown in Figure 9.

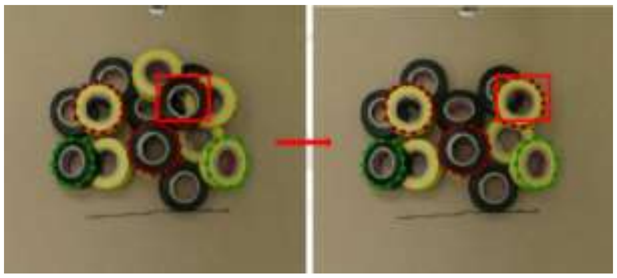
Fig. 9: Schematic of image annotation

3.2.3 Object Recognition

In deep learning object recognition, we used the Mask R-CNN proposed in [17]. Mask R-CNN is a very flexible framework that can add different branches to achieve different tasks, such as target detection, target classification, instance segmentation, semantic segmentation, and human body posture recognition. We used the COCO pre-training model, [18], to reduce the training time and a lightweight CNN model, Inception, to identify various features of graphics with perfection and importance, [19]. A power supply tape, with simple features, was the object trained in this study. According to the characteristics of Inception, the training graph has larger features. The training weight is negligible for relatively few smaller features. Therefore, in theory, the training speed of Inception is faster than that of ResNet.

3.3 Grasping Position Calculation

3.3.1 Object Grab Position Evaluation

To establish a fast gripping process, a method of self-defining gripping points was adopted in this study. This method is different from that proposed in [20]. They proposed a five-dimensional grasping box as a grasping representation and a system with a two-level deep network based on a two-dimensional image. As shown in Figure 10, a shallow CNN determined all possible grasping rectangles, retained some grasping rectangles with higher scores, and then determined the highest-scoring grasping rectangles among the remaining grasping rectangles through a deep CNN. That is, the shallow CNN determined and obtained the best grasping rectangles. After capturing the rectangular frame, the normal direction of the point cloud in the center of the rectangular frame was used as the approach vector of the manipulator; the detection accuracy rate reached 75%, and the processing time of each image was approximately 13 s, [20]. The processing time of this detection algorithm is relatively low, and the computation is extremely large and time-consuming.

Therefore, in this study, the center point of the object was defined as the gripping point and used to open the gripper as the gripping method. At this point in the box, the object can be gripped under most postures. Thus, the collisions between the gripper and the object can be effectively avoided during gripping, and the success rate of gripping can be improved. This allowed clamping in a stacked environment. This method saved the time of labeling the clipped frame and modeling the clipping system as only RGB-D was used for calculation. This reduced the equipment requirements for image processing, and the computing time was also higher than when using a large number of point cloud computing methods.

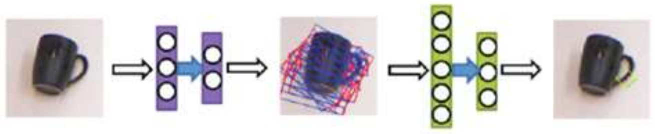

Fig. 10: The common grab detection process

3.3.2 Grasping Pose Evaluation of Objects

Evaluating the power tape in the box and the posture of the object, which is related to the gripping and success rate of gripping, is necessary. The scattered and unstacked objects usually have two attitudes: lying flat or standing, as shown in Figure 11. Among these, grabbing objects in a flat state does not face problems. Since an eye-to-hand system was used in this study, the camera could not move with the robotic arm. The system failed to identify the central grasping position of a standing object and to comprehend whether the four sides of the box would follow the robotic arm when grasping. This is called an interference problem. Thus, the robotic arm could not grasp the standing object at a position relative to the center point of the object. Therefore, if the object was standing or its center point was unrecognizable, the object could not be grasped, and a mechanical arm had to be used. The arm pushes the object down to a flat state, re-identifies it, and clamps it.

In a stacking state of the objects in a box, usually, the objects are inclined to a certain angle, as shown in Figure 12. The attitude angle of the object to the camera was converted into the attitude of the object to the robot arm after the conversion of the hand-eye correction. Finally, designing the grasping angle by considering a situation in which the four sides of the box do not interfere with the grasping is necessary. The robotic arm was expected to grab the object at the same attitude angle.

In addition to calculating the gripping pose, depth information was used to determine the object pose. First, the objects cut by the Mask R-CNN instance were divided into four regions, as shown in Figure 13. Then, we extracted seven 3D spatial information points from each of the four regions and one 3D spatial information point from each of the three regions.

We set them as $A(x_1, y_1, z_1), B(x_2, y_2, z_2), C(x_3, y_3, z_3)$and set the plane normal vector as $\vec{n} = (x, y, z)$. Next, the angle of its normal vector was calculated using Formula (2), and A, B, and C were substituted into the three-point coordinates to obtain Formula (3), which can be obtained by the Formula (4) plane equation vector and calculate its angle, as shown in Figure 14. Then, the center point of the object was set as the origin of the plane angle, and another azimuth quadrant and the angle in the quadrant were determined based on the object information and calculated normal vector angle, as shown in Figure 15. Finally, the thus obtained object position and attitude were converted into X、Y、Z、R_x、R_y、R_z of the robot arm through calculations and sent to the robot arm.

$$\begin{cases} \vec{n} \times \overline{AB} = 0 \\ \vec{n} \times \overline{AC} = 0 \end{cases} \quad (2)$$

$$\begin{cases} x(x_1 - x_2) + y(y_1 - y_2) + Z(z_{1-}z_2) \\ x(x_1 - x_3) + y(y_1 - y_3) + Z(z_{1-}z_3) \end{cases} \quad (3)$$

$$A_x + B_y + C_Z + D = 0 \quad (4)$$

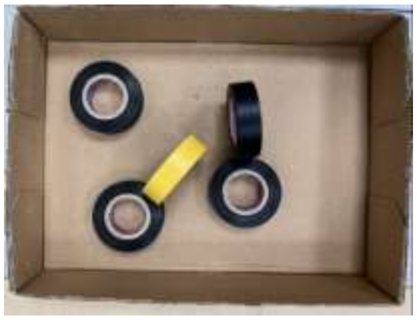

Fig. 11: Flat and standing object indication

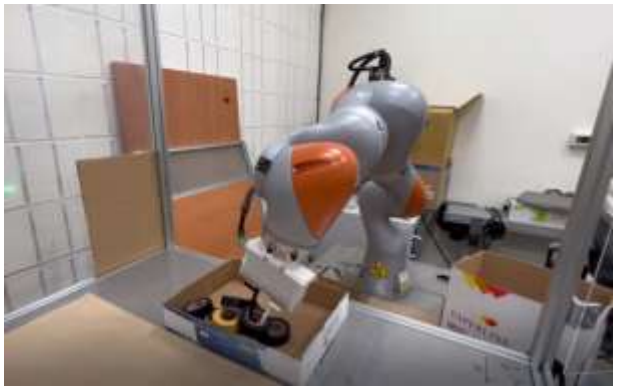

Fig. 12: Grab posture when objects are stacked

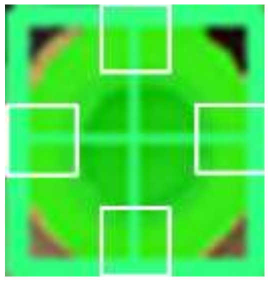

Fig. 13: Four regions of the instance cutting object

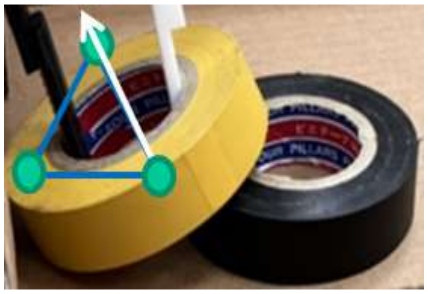
Fig. 14: Schematic of normal vector angle calculation

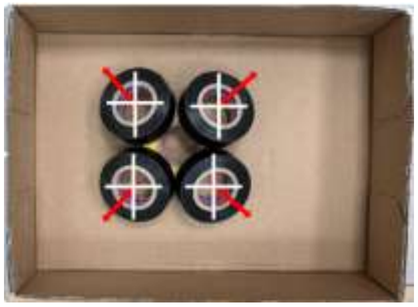
Fig. 15: Object Plane Quadrant Angle

3.4 Robotic Grasping Control
The robotic arm was set to move from the object placement area to the object grasping area for six movements. To simplify the entire process, the robotic arm posture was calculated by reading the PC. The robotic arm was then moved to the top of the object through three pre-set positions, and finally, the arm was allowed to perform the grasping action.

4 Result and Discussion
From the experiments, we verified the results and training time of deep learning using Inception.

The objects in the box were stacked, and the robotic arm grabbed them until all the objects were clamped or their image could not be recognized. When the algorithm incorrectly judged the calculated position of the object as not within the gripping area, the gripping system was stopped. The grasping failure occurred when the pose calculated by the algorithm based on depth information differed from the required position pose. Failure to grasp indicates that the object cannot be successfully grasped or stably placed in the area after grasping. In an unrecognized state, image recognition faces algorithmic errors, which prevent the correct position from being assessed.

Finally, we improved the experimental results using the depth information, and consequently, the grabbing rate increased from 90.86% to 95.1%.

4.1 Results of Deep Learning
In this study, Inception was used as the training model for the Mask R-CNN. The total number of data maps was 1098, of which 997 and 101 were used as the training and test sets, respectively. The first training result was obtained after approximately 380,000 iterations. When the most commonly used 50% IoU was used, the accuracy reached 0.962 at approximately 30,000 iterations; at 45,000 iterations, the accuracy reached 0.962. 0.985. At 75% IoU and approximately 40,000 iterations, the accuracy reached 0.955. When the number of iterations was approximately 50,000, the accuracy reached 0.983, as shown in Figure 16. In terms of speed, recording every 1,000 iterations required approximately 17 s. Recording of 500,000 iterations was completed in approximately 2.5 hours plus the storage and operation time. With the previous data collection, the entire process could be completed in approximately 4–5 hours, and the speed could be considered significantly high. The recognition results are presented in Figure 17.

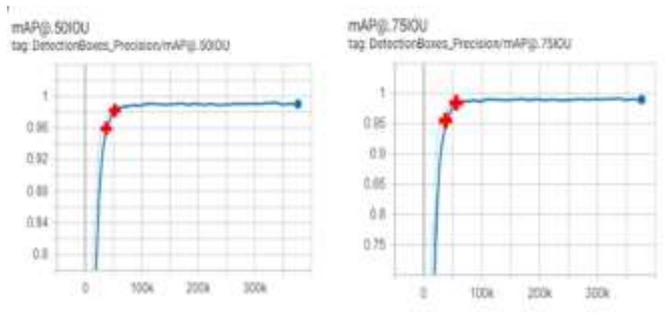

Fig. 16: Deep Learning Results (Average Precision)

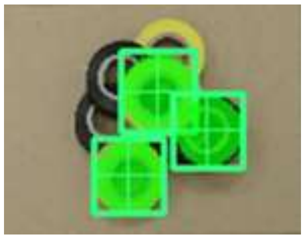
Fig. 17: Mask R-CNN recognition results

4.2 Stacked Gripping Results
In the stacking state, 350 clamping experiments were performed, and the recognition rate of deep learning object recognition in the stacking state was found to be good. However, in various instances, the object recognition was insufficient. This resulted in errors in the clamping position calculations. Although the attitude evaluation was successful 314 times, a positional deviation that was different from the plane placement was still observed. Out of the 36 errors in attitude evaluation, in 33 instances grasping failure occurred, including both

the correct and incorrect attitude evaluation, failure to grasp, and misplacement errors caused by incomplete object recognition. Finally, the number of successful clippings, including the evaluation errors, totaled 317. The identification and gripping rates are summarized in Table 1.

Table 1. Object stacked grab rate

	Evaluate
Object recognition success rate	100%
Object pose evaluation is correct	89.71%
Object pose evaluation error	10.29%
The total number of gripping	350 times
	Gripping rate
The object posture assessment is correct and the gripping is successful	87.99%
Object pose assessment fails to grip correctly	1.72%
Object pose evaluation error, the gripping is successful	2.87%
Object pose evaluation error gripping failed	7.42%
The total gripping success rate	90.86%

4.3 Result Improvement

In this section, the ways to improve the experimental results are discussed. In most cases, successful gripping was accomplished using the correct posture. In several cases, although the posture was incorrect, clipping was successful. The majority of gripping failure instances occurred owing to incorrect attitude judgment. This can be attributed to the use of the depth image data for calculating the position and attitude and insufficient accuracy of the Kinect V2 depth image, which resulted in the misjudgment of attitude. In addition, the small number of grasping failures can be attributed to the incomplete cutting of the instance. Even if the grasping posture is calculated correctly, the gripper cannot move within the grasping tolerance of the center of the object; therefore, it cannot grasp an object successfully.

The important data used in attitude evaluation is depth information. However, the error of object attitude evaluation was as high as 10%. In the experimental test, an object was placed in the gripping area to test the stability of the depth information at four points on the object. The depth information was then displayed 100 times at the four points. The results are shown in Figure 18. Even at the same time point, the depth values determined at each time point were significantly different. The algorithm used three of the four points as the normal vector and attitude calculation method. According to the result, the error was ±5 mm, which was also the reason for the wrong attitude evaluation.

To improve the stability of the abovementioned unstable depth information and achieve a higher gripping rate, we modified the original depth information extraction method and used a camera at the same point before extracting the information. The depth information was extracted several times, and the low-pass rate wave method was used to extract the waveform with the highest weight ratio from the depth information. The results are shown in Figure 19; the depth of the information error was significantly improved.

Finally, based on the improved depth information, we performed a gripping test in a stacking environment. A total of 102 grip tests were conducted, and 96 successful attitude evaluations and six attitude evaluation errors including five gripping failures were obtained. The second time included both the correct and incorrect attitude evaluations, clamping failure occurrence, and misplacement error caused by incomplete object recognition. The number of successful gripping events included evaluation errors and 97 successful gripping events. This was also compared with 109 gripping tests of the original depth information acquisition method. The stability and gripping rate improved to a certain extent. The identification and grip rates are listed in Table 2.

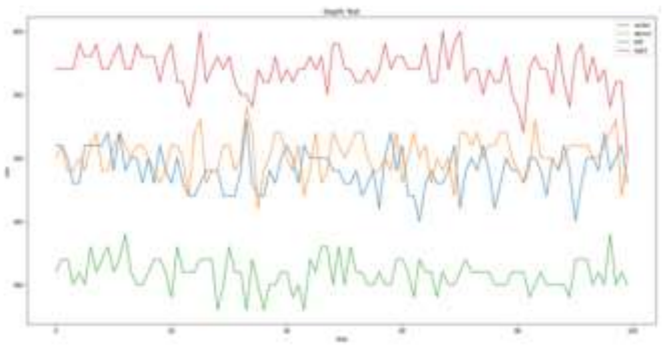
Fig. 18: Depth information before filtering

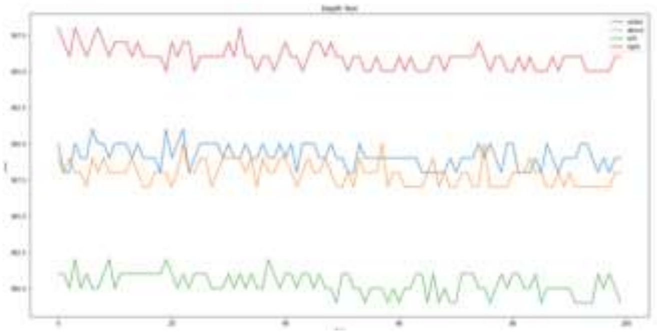
Fig. 19: Depth information after filtering

Table 2. Comparison of the gripping rate before and after filtering

	before filtering	after filtering
Object recognition success rate	100%	100%
Object pose evaluation is correct	89.91%	94.12%
Object pose evaluation error	10.09%	5.88%
The total number of gripping	109 times	102 times
	Gripping rate (before filtering)	Gripping rate (after filtering)
Correct object posture assessment and successful gripping	88.07%	93.14%
Incorrect object pose assessment and incorrect gripping	1.84%	0.98%
Object pose evaluation error; successful gripping	3.67%	1.96%
Object pose evaluation error gripping failed	6.41%	3.92%
The total gripping success rate	91.74%	95.1%

5 Conclusions

In this study, robotic arms, computer vision, and deep learning were integrated to develop a simplified and faster-to-implement grasping system compared to the time-consuming and intricate stacking grasping systems. The focus was on mass-produced objects. In the experimental phase, we opted for simpler objects for grasping tests by employing a manual method to set the gripping points. This approach facilitated swift data collection and labeling and expedited the preliminary steps for deep learning. For the training model, we selected Inception owing to its faster training capabilities. Overall, ResNet outperformed Inception in terms of recognition accuracy; however, Inception was selected owing to its faster model training. The final experimental results were consistent with our expectations.

Furthermore, in contrast to other studies, our approach does not rely on point clouds or CAD modeling. Instead, we utilized color and a limited amount of depth information to establish the posture of an object in a 3D space. This enabled the robotic arm to grasp objects based on the posture direction. In experiments using this grasping method, we achieved success rates exceeding 95% for both unstacked and stacked objects. Consequently, our study offers advantages in terms of the speed at which the entire grasping system can be established and the overall success rate.

Although this study introduced a grasping system that can be quickly deployed and attains a commendable success rate, the system can be improved further.

- Diversifying the types of objects that can be identified. In this study, we focused on a single-object type, which limits its applicability. Our goal was to enable the robotic arm to handle various objects that could be randomly stacked within the grasping area without compromising the success rate.
- Furthermore, we aim to improve the depth point accuracy by employing a more stable RGB-D camera, thus enhancing the overall attitude evaluation success rates and grip efficiency.
- To address challenges, such as collisions, and further improve the success rate, we envisage incorporating a force sensor into a robotic arm in the future. This sensor will allow the system to detect errors during the grasping process by relying on force feedback to determine whether the gripper has contacted the intended object. By analyzing the force data, the robotic arm can assess if a successful grip has been achieved or if it should return to the grasping waiting area for image recognition.

References:

[1] H.Y. Kuo, H.R. Su, S.H. Lai, and C.C. Wu, “3D Object Detection and Pose Estimation from Depth Image for Robotic Bin Picking,”

Proc. of IEEE Int'l Conf. on Automation Science and Engineering, pp. 1264-1269, 2014.

[2] C. Wu, S. Jiang, and K. Song, "CAD-based pose estimation for random bin-picking of multiple objects using an RGB-D camera," *2015 15th International Conference on Control, Automation, and Systems (ICCAS)*, 2015, pp. 1645-1649.

[3] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3406-3413.

[4] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg. 2017. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics, https://doi.org/10.48550/arXiv.1703.09312.

[5] H. Wang, H. Situ, and C. Zhuang, "6D Pose Estimation for Bin-Picking based on Improved Mask R-CNN and DenseFusion," *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1-7.

[6] D. Morrison, P. Corke, and J. Leitner. Closing the loop for robotic grasping: a real-time, generative grasp synthesis approach. 2018, arXiv: 1804.05172, https://doi.org/10.48550/arXiv.1804.05172.

[7] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection," *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore: IEEE, 2017: 1609-1614.

[8] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. 2014, arXiv: 1412.3128, https://doi.org/10.48550/arXiv.1412.3128.

[9] S. Kumra and C. Kanan. Robotic grasp detection using deep convolutional neural networks. 2016, arXiv: 1611.08036, https://doi.org/10.48550/arXiv.1611.08036.

[10] J. Jiao, L. Yuan, W. Tang, Z. Deng, and Q. Wu. "A Post-Rectification Approach of Depth Images of Kinect v2 for 3D Reconstruction of Indoor Scenes," *ISPRS International Journal of Geo-Information*, 2017; 6(11):349.

[11] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, pp. 666-673.

[12] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," in *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345-358.

[13] F. C. Park and B. J. Martin, "Robot sensor calibration: solving AX=XB on the Euclidean group," in *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717-721, Oct. 1994.

[14] R. Horaud and F. Dornaika. Hand-eye Calibration. *The International Journal of Robotics Research*, SAGE Publications, 1995, 14 (3), pp.195–210.

[15] N. Andreff, R. Horaud, and B. Espiau, "On-line hand-eye calibration," *Second International Conference on 3-D Digital Imaging and Modeling(Cat.No.PR00062)*,1999, pp.430-436.

[16] H.Y. Kuo, H.R. Su, S.H. Lai, and C.C. Wu, "3D Object Detection and Pose Estimation from Depth Image for Robotic Bin Picking," *Proc. of IEEE Int'l Conf. on Automation Science and Engineering*, pp.1264-1269, 2014.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980-2988.

[18] Cocodataset. COCO, [Online]. https://cocodataset.org/#home (Accessed Date: February 27, 2024).

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Robinovich, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07–12-June, pp. 1–9

[20] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *International Journal of Robotics Research*, 2015, 34(4/5): 705-724.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

- S.H. Lee carried out the experiment and analyzed the results.
- B.R. Zhu wrote the manuscript. J.S. Shaw conceived the experiment and reviewed the manuscript.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

This study was supported by a grant from the National Science and Technology Council, Taiwan (NSTC 112-2218-E-027-006).

Conflict of Interest

The authors declare no conflict of interest.