Robotic grasp detection using effective graspable feature selection and precise classification

Jiahao Zhang
School of Automation Science and Engineering
South China University of Technology
GuangZhou, China
jiahaolovelife@outlook.com

Miao Li
Suzhou Institute of Wuhan University
Wuhan University
Suzhou, China
miao.li@whu.edu.cn

Chenguang Yang*
Bristol Robotics Laboratory
University of the West of England
Bristol, UK
cyang@ieee.org

***Abstract*—It is necessary to implement real-time grasp detection in robotic grasping tasks. To this end, in this paper we propose a method for effective graspable feature selection and precise classification. In a robotic grasping scene, our method can effectively select graspable rectangles and further extract useful features from them to generate a feature set. A convolutional neural network (CNN) is then developed to score and classify the elements in the feature set. Finally, we compute the desired robotic grasp pose based on the graspable feature that gets the highest score. In the test phase the proposed CNN network achieves an accuracy of 96.5% on the Cornell Grasping Dataset. In real-world grasping experiments 105 frames per second (fps) for the object's grasp detection and a grasp success rate of 89.9% have been achieved with our method.**

***Index Terms*—Grasp detection, effective graspable feature selection and precise classification**

I. INTRODUCTION

Robotic grasping can be applied in many aspects like social services and industries. However, it is difficult for robots to implement precise grasp like humans. An intuitive way for humans to grab an object is to see it, estimate it and grasp it. According to the way that humans grasp an object, robotic grasping can be divided into three parts, which is:

- **See:** The best way for robots to "see" objects is to use sensors like RGB-D cameras. With the fast development of vision sensors, robots can get a lot of information to estimate objects' feature.
- **Estimate:** When involving how to estimate the graspable feature of an object, deep learning methods like convolutional neural network (CNN) show great power on it.
- **Grasp:** When robots finish identifying graspable features, there are some parameters that represent the robotic grasp. Robots need to convert the grasp representation to its end-effector poses, and then plan its trajectory.

Our proposed solution is based on three parts. Firstly, we use the same grasp representation in [1], which has five individual parameters. Secondly, RGB-D sensor is used to capture images in real-world, then, an image processing method is employed to generate effective grasp rectangles, finally, we extract a feature set from the grasp rectangles and classify every item in the feature set with our CNN model. The CNN classifier is trained on the Cornell Grasp Dataset, which has numerous RGB-D data and labeled representation of grasps.

When regarding the grasps as a classification problem, the widely shared method is a sliding window selection approach. This approach searched the small patches globally, and used a classifier to score every patch.

Compared to previous works, our methods can avoid globally searching and has more accurate classification. In this paper, we improved the morphological image processing (MIP) algorithm in [2] to do effective graspable feature selection and train an accurate classifier to find the best grasp. The biggest advantage of our methods are:

- Compared to the sliding window searching and our previous MIP method in [2], our improved MIP algorithm can select graspable features more effectively, and for some real-world scenes, it can decide the final grasp configuration directly. The select graspable features are constituted into a set, which is used for our CNN classifier.
- We train a CNN classifier on Cornell Grasp Dataset with an accuracy of 96.5%. This classifier is very helpful for us to choose the best grasp configuration from the graspable feature set, which is generated by our improved MIP algorithm.

The architecture of our method is illustrated in Figure (1). We focus on searching graspable feature with a effective way and use the searched feature set to decide the best grasp configuration in real time.

Our paper is organized as follows: Section II introduces the related works on robotic grasps. And Section III describes how our algorithm perform grasp detection. To evaluate our algorithm, we implement real-world grasp experiments with Microsoft Kinect and Baxter robot. Finally, we draw the conclusion in Section V.

II. RELATED WORK

There has been a lot of researches on robotic grasp over the last few years. These researches are aimed to solve the grasping problem from the three parts we mentioned in Section I. RGB-D sensors have been widely used to get real-world data, but it is hard to get the full information by a single shot.

*Corresponding author is C. Yang. Email: cyang@ieee.org.

This work was partially supported by Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/S001913 and Natural Science Foundation of Jiangsu Province under Grant BK20180235.

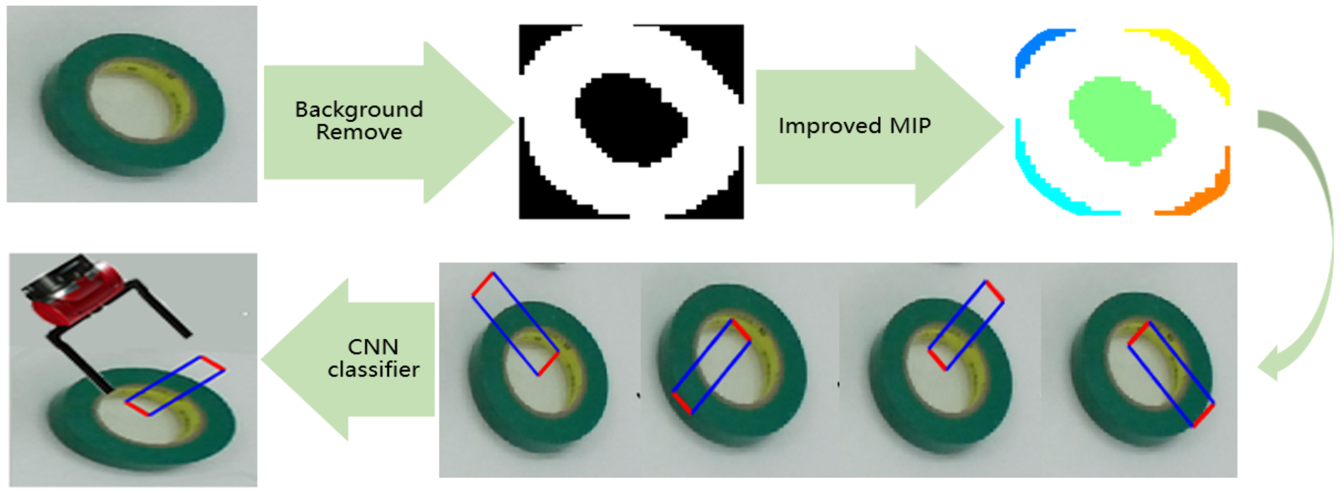

Fig. 1. The architecture of our proposed system

Some researches in [3] [4] try to get complete information of objects by three dimensional simulation. They can have good performance in simulation, but it is not easy to achieve the performance of simulation in real world. To better understand of objects' geometry, [5] uses two cameras to synthesize the point cloud of data. In terms of the generality robotic grasp, complex models of objects are not helpful to grasp unknown objects.

For estimating problems, a lot of machine learning and deep learning methods are used in analyzing graspable features. For classification problems, a number of previous researches show that support-vector machine (SVM) in [6] has great power on it. There are a lot of grasp solutions were based on SVM. [7] [8] [9] use SVM to score the grasp configuration, and then choose the one got the highest score to execute robotic grasp. SVM is also employed in classifying antipodal grasps in [5]. Deep learning method is also applied in grasp classification, like sparse auto encoder (SAE) in [1]. Lenz et al. [1] trained a SAE model on multimodal data and get a high classification accuracy of 93.7%. But their sliding window approach for feature selection limits the speed of detection. Deep learning methods has been widely developed on object detection such as YOLO [10], Faster R-CNN [11], Mask R-CNN [12], and semantic segmentation such as FCN [13], UNet [14].

Lots of interesting work based on these methods have been used on grasp detection. Chu et al. [15] propose a grasp proposal network and their idea is to transfer the grasp rectangle detection to object detection. And their results show high classification performance. [16] achieves pixel-wise grasp rectangle detection. Their work uses the fully convolutional network like U-net to predict rectangle for every pixel. Their network is significantly smaller than other network because of no fully connected layers. [17] uses Gaussian Mixture Model to detect robotic grasp.

After deciding the final grasp configuration, it is of significance to plan the grasp. [18] [1] used a seven or five dimensional rectangle to represent the grasps, then converted it to the grasp pose of objects, moved the robot's end-effector to the pose and finally closed the grippers.

III. THE SYSTEM OF GRASP DETECTION

This section is divided into four subsections. The first part describes what the problem of robotic grasp is. The next three parts introduce how we solve the problems.

A. Problem Description

The input of our method is a RGB image and the aligned depth image captured from a tabletop scene, and the output is the five-dimensional grasp rectangle that can be convert into the grasp pose by eye-to-hand or eye-in-hand calibration. The five-dimensional rectangle is shown in [1], which can be represented by equation (1).

$$g = \{\mathrm{x}, \mathrm{y}, \theta, \mathrm{h}, \mathrm{w}\} \tag{1}$$

where g is the grasp rectangle representation, $(x, y), \theta$ are the center location and orientation respectively, and the three parameters can be converted into the object's grasp pose. h, w are the thickness and open width of the gripper in our experiments.

Because the thickness of our Baxter robot's gripper is fixed, we can ignore the parameter h. To use the rectangle representation, we set the relationship between h and w as $h = 0.25 * w$.

We use an improved MIP algorithm to choose grasp rectangle g. For every selected g, we extract features such as RGB data, depth and surface normal information as f_i and get a feature set F. Equation (2) indicates the relationship of f_i and F. After generating the feature set, our CNN classifier score and classify the whole set and determine the best grasp configuration.

$$F = \{\mathrm{f}_1, \mathrm{f}_2, \ldots\ldots, \mathrm{f}_n\} \tag{2}$$

B. Background Removing

In order to reduce the time for detection, we need to remove the useless data. For the tabletop scene, it is practical to use the random sample consensus method (RANSAC) [19] or voxel cloud connectivity segmentation algorithm (VCCS) [20] to remove the information except for objects. But the two methods are relatively time-consuming.

$$\begin{aligned} depth_{bg} &= getCached() \\ depth_{mask} &= (depth_{fg} - depth_{bg}) > threshold \end{aligned} \tag{3}$$

For quick background removing, our methods are shown in equation (3). First, we cache a depth image before placing the objects on the table, and then use the foreground depth image which contains objects subtract the cached image. Finally, we set a threshold 8 to binarize the result of subtracting and we obtain the binary mask for the input of our improved MIP algorithm. The threshold can be adjusted according to different real-world situation. And we visualize the binarized mask in Figure (1).

C. Graspable Feature Selection

After removing the background, we acquire a binary mask. Our idea is to generate a grasp feature set effectively. We propose an improved MIP method, it can quickly generate grasp rectangle hypothesis by identify possible patches that a robot gripper can fall. This improved MIP algorithm is based on our previous work in [2] [21].

Algorithm 1 Improved Morphological Image Processing

```
Input:
    a binary image, $depth_{mask}$
Output:
    a graspable feature set, $F$
 1: Gain the properties of all regions in $depth_{mask}$
 2: for $reg$ in regions do
 3:     grasp rectangle ($g$) = None
 4:     if Pixel($reg$) / Pixel(ConvexArea($reg$)) $>$ 0.95 and
        Pixel($reg$) / Area of $reg's$ boundingBox $>$ 0.95 then
 5:         Adjust height and width of the bounding box of $reg$
 6:         Regard the modified bounding box as $g$
 7:     else
 8:         Identify all possible patches where a robot gripper
            could fall
 9:         Calculate the centroid coordinate of every patch
10:         Set $N_p$ as the number of all possible patches.
11:         if $N_p$ = 2 then
12:             Transform the two different centroids into $g$.
13:         else
14:             Transform every two different centroids into $g$.
15:             For each $g$, extract its features and append to $F$.
16:         end if
17:     end if
18: end for
```

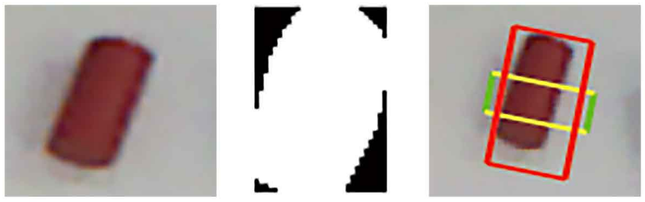

Fig. 2. A special case for our algorithm. From left to the right shows the original object, the convex hull of the object and the result of our algorithm. The red rectangle in the far right image is the bounding box for the red round rod and another rectangle is the generated grasp rectangle.

Our whole processing pipeline is illustrated in Algorithm 1. Step 1 use $depth_{mask}$ as input, and implement blob detection to gain every single region from it. Each region corresponds to an object. Step 2 is an iterative procedure, which is used to process every region (object) parallelly. Step 4 to 6 is a special case. The $Pixel()$ in Step 4 is a function to calculate the number of pixels. When the condition in Step 4 is satisfied, we think this is a rectangular object such as cuboid and it is put with vertical or horizontal direction on the table. And we set the width of our grasp rectangle as $1.3 * shortEdgeLen$, where $shortEdgeLen$ indicates short edge length of the region's bounding box.

An example of our special case is illustrated in Figure (2). And Figure (3) shows the detection result of some rectangular objects in our work. The generated rectangles from Step 4 to 6 can be used to do robotic grasp directly.

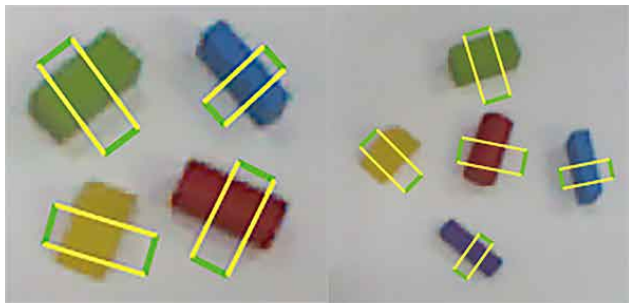

Fig. 3. The detection results for rectangular objects.

Step 8 to 14 is the general case of our grasp rectangle selection. From Step 1, we can get the convex hull of reg, and the pixel in the convex hull is our region of interest. The region of interest is shown in the upper right of Figure (3). In Step 8, for every region, we filter the convex hull with a two-dimensional convolutional kernel that is a 5×5 all-one matrix. This filter operation can help our algorithm get the patches near the object-wise region. We get the all possible patches by using the convex hull region to subtract the original mask region, which is visualized in the bottom left of Figure (3). Step 11 and Step 14 is to generate all grasp rectangles by the patches and every two different patches can generate one grasp rectangle.

In Step 11, we only generate one rectangle and we use it as the final grasp rectangle directly. Compared to Step 11,

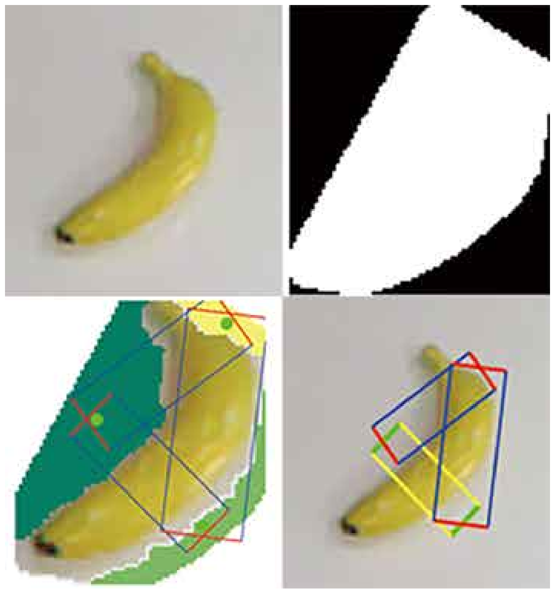

Fig. 4. The detection results of a banana. The top-right image is the convex hull of the banana, the bottom-left image is the output of our MIP algorithm and the bottom-right image is the results of our CNN classifier (The yellow and green grasp rectangle is the best grasp of all).

Step 14 generates a lot of grasp rectangles, which is shown in Figure (4) and in Step 15, we extract all features for the grasp rectangles and use the features with the image pre-processing method in Section III-D1 as input for our CNN classifier.

In addition, our improved MIP algorithm can extract graspable features whith a rate of 164 fps. This is indispensable for our CNN classifier to achieve real time robotic grasp detection.

Our improved MIP algorithm has the powerful ability to implement robotic grasp without any additional classifier. We replace Step 14 and 15 in Algorithm 1 with randomly choosing two centroids and transform the centroids to the final grasp rectangle. Then the improved MIP can generate grasp rectangle for every single object. And in Table (II), our experiments show it gets a unexpected grasp success rate about 78.0 %. The reason that we can get this success rate is that the choosing centroids from two different patches is the approximate location where the gripper falls, and the final grasp has a high probability to be an antipodal grasp.

D. CNN Classification

1) Image Pre-processing: We train and evaluate the classification model on the Cornell Grasping Dataset. Our processing method is learned from the approach in [1]. [1] extract the image information which has totally seven channels included the RGB space (3 channel) , depth (1 channel) and surface normal data (3 channels) from a 24×24 rectangle. We use the $24 \times 24 \times 7$ data with a rescale operation as our CNN classifier input. Because the depth and surface normal information are unlike the RGB information, to the get better performance on CNN network, before feeding into our network, we normalize every channel by subtracting its mean value and deviding its standard deviation.

2) Grasp Classification: In Section II, recent works have shown CNN has great power on grasp rectangle regression. We try to regard the robotic grasp as CNN classification problem.

Our network starts with four convolutional layes and ends with three fully connected layers. There are also max pooling and batch normalization layers followed with convolutional layers at some stage. The whole architecture of our CNN network is shown in Figure (5).

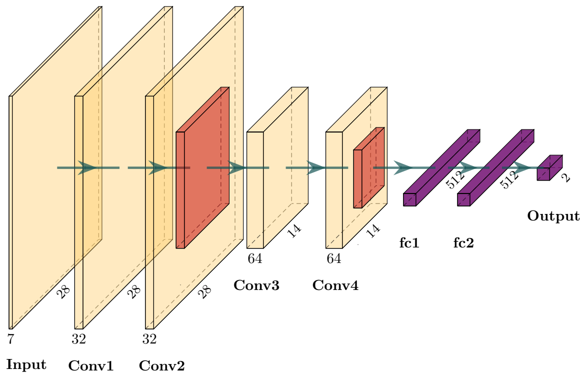

Fig. 5. The structure of our CNN classifier

The output of our model is a two-dimensional tensor. The first dimension and the second dimension are represented for negative and positive grasp respectively. Equation (4) illustrates how we represent grasps.

$$gt = \begin{cases} \{1,\ 0\}, & for\ negative\ grasp \\ \{0,\ 1\}, & for\ positive\ grasp \end{cases} \tag{4}$$

Training: We use binary cross entropy as our loss function and rescale the features(24*24*7) to 28*28*7. The number of training epoch is 25 and the learning rate is set as 0.001 with the Adam optimizer. Figure (6) shows the accuracy on training phase.

We train our network end-to-end on a single Nvidia RTX2080 with 8G GPU memory. Our framework is based on Pytorch with cuda-10.0 and cudnn-7.4.2.

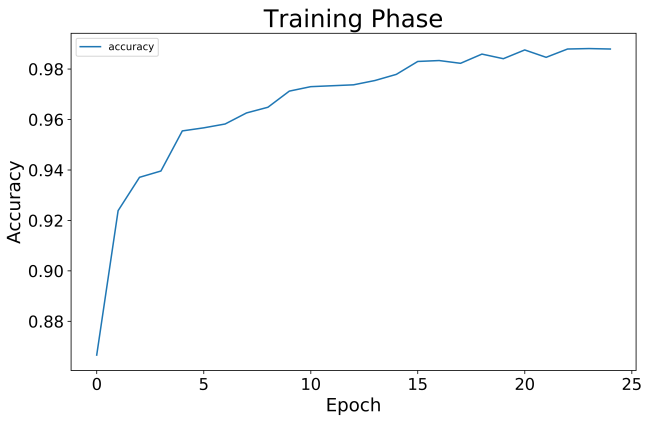

Fig. 6. The training accuracy of our CNN classifier

TABLE I
COMPARISON ON RESULTS OF DIFFERENT ALGORITHMS

Algorithms	Accuracy (%)
Sparse Auto Encoder [1]	93.7
Support Vector Machine [2]	94.7
Random Forest [21]	94.2
Our proposed CNN	96.5

Results: Using the combined features that are RGB, depth and surface normal data helps us get an accuracy of 96.5 %. Our comparison results are shown on Table (I). Our network lead 2% ahead than the accuracy of the current classification approaches.

IV. ROBOTIC GRASP EXPERIMENTS

To show the abilities of our proposed method, we perform two-stage experiments in real-world scene, which is single object grasp and multiple object grasp.

In order to evaluate our method in real-world, we implement robotic grasping on a Baxter robot and use Microsoft Kinect to get RGB-D data. The Baxter robot is a two-arm manipulator with all elastic joint and it has seven degrees of freedom (DOF), which makes it kinematically redundant. For convenience, We just use one arm of the Baxter for grasping. Also, the Baxter robot has a parallel one degree of freedom gripper.

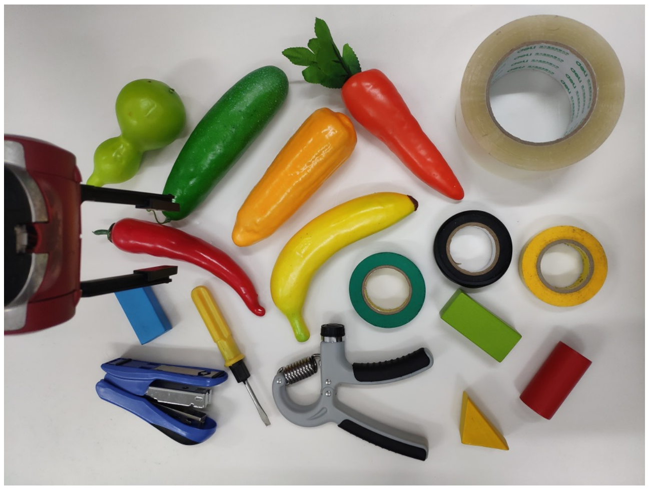
Fig. 7. Typical objects we used in our experiments

Experimental Setup: In order to verify the grasping detection performance of our algorithm effectively, we selected a lot of objects that are common in life to execute robotic grasps. Figure (7) shows some typical objects that we used, and they differ in color, shape, size, and weight. The total number of objects is about 25 and they are unknown for our proposed method. We use a computer which has a memory of 8GB, an Intel (R) Core (TM) i7-6700 CPU and a GTX 1060 6GB graphic card to compute the grasp rectangle. Before starting to perform robotic grasp, we collect a depth image of the tabletop scene as the background information. This will be used for our background removing part in our proposed method.

Our objects grasp experiments are executed by the following steps: first, we place objects on the table with random position and orientation; then, we use our improved MIP algorithm and the CNN classifier to calculate the grasp rectangle; finally, the Baxter robot tries to grasp objects by converting the grasp rectangle to grasp pose.

Experimental Results: We use the single object scene to show how our proposed method detects the grasp rectangle. Figure (8) shows the detection results and the larger the number of all possible patches (N_p), the greater the number of generated rectangles.

We also use multiple objects scene to validate our proposed methods. We test our method on three totally different scenes, which is fruit model scene (Scene 1), building blocks scene (Scene 2) and office tool scene (Scene 3). All the experimental results are shown in Table (II). For every object and scene, we implement 12 attemps to do robotic grasp. And our final results show that without the CNN classifier, our improved MIP algorithm can detect the grasp rectangle with a speed of 164 fps and keep the grasp success rate as 78.0 %. After using the CNN classifier as the decision maker, we can get the grasp success rate as 89.9 % with a speed of 105 fps.

V. CONCLUSIONS

We propose a novel and precise graspable feature detection method to do robotic grasp. Our methods are based on the image processing algorithm and convolutional neural network.

We implement table-cleaning experiments in an RGB-D view to validate the effectiveness of our methods. We use the improved MIP algorithm to search the patches that can be used to generate grasping, this method creatively avoid global grasp searching and save a lot of time to generate grasp rectangles for every object.

Compared to current state-of-the-art approaches, our experiments show that our improved morphological image processing (MIP) algorithm can select graspable features effectively and quickly, and the proposed CNN classifier can help the improved MIP algorithm select the graspable features more precisely.

REFERENCES

[1] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[2] J. Zhang, C. Yang, M. Li, and Y. Feng, "Grasping novel objects with real-time obstacle avoidance," in *International Conference on Social Robotics*, pp. 160–169, Springer, 2018.

[3] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[4] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1824–1829, IEEE, 2003.

[5] A. ten Pas and R. Platt, "Localizing antipodal grasps in point clouds," *CoRR abs/1501.0*, 2015.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

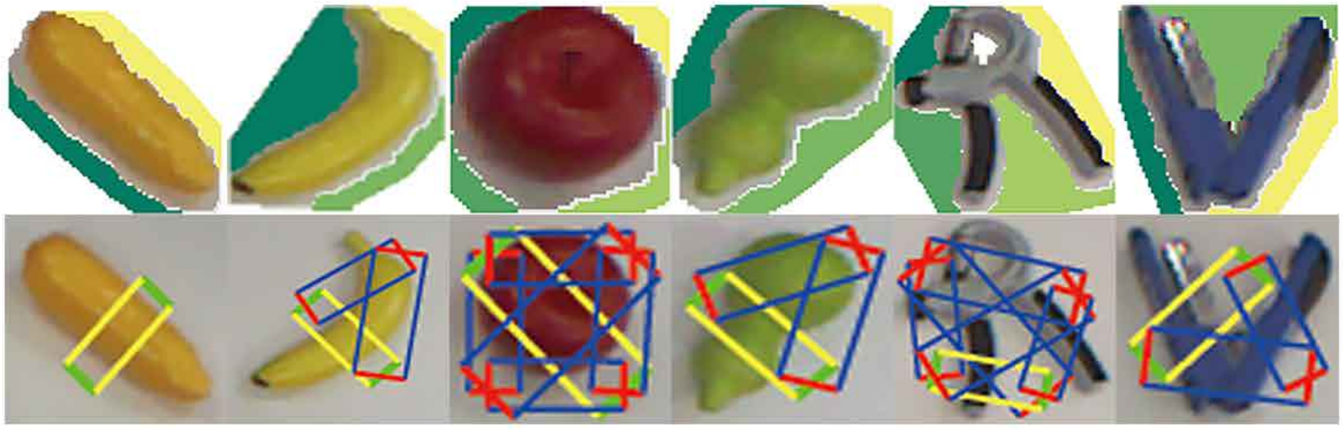

Fig. 8. Grasp detection in different scenarios. The top image shows the detection results of our improved MIP algorithm. The bottom image shows the results of our CNN classifier, which is the final grasp rectangle (The green and yellow rectangle is the final grasp rectangle, and all rectangles in the bottom image are the outputs of our improved MIP algorithm.)

TABLE II
RESULTS OF BAXTER GRASPING IN DIFFERENT SCENE

Experiments				Grasp Attemps	Grasp Success Rate (%)		Average Detection Time (ms)	
					No CNN	With CNN	No CNN	With CNN
Single object scene	Case 1		blue block	12	100	-	9.3	-
			green block	12	100	-	6.5	-
			round rod	12	91.7	-	7.8	-
	Case 2	Np = 2	corn	12	100	-	8.5	-
			ginger	12	91.7	-	8.1	-
			cucurbit	12	100	-	7.2	-
		Np >2	tape	12	58.3	100	9.7	26.5
			banana	12	66.7	91.7	7.5	19.6
			stapler	12	58.3	75	8.7	20.5
			hand grip	12	41.6	83.3	8.9	25.4
Multiple objects scene	Scene 1: five objects			12	71.6	88.3	20.3	33.7
	Scene 2: five objects			12	93.3	-	18.0	-
	Scene 3: five objects			12	63.3	81.6	31.5	48.2

[7] D. Fischinger and M. Vincze, "Shape based learning for grasping novel objects in cluttered scenes.," in *SyRoCo*, pp. 787–792, 2012.

[8] D. Fischinger and M. Vincze, "Empty the basket-a shape based learning approach for grasping piles of unknown objects.," in *iros*, pp. 2051–2057, 2012.

[9] D. Fischinger, A. Weiss, and M. Vincze, "Learning grasps with topographic features," *The International Journal of Robotics Research*, vol. 34, no. 9, pp. 1167–1194, 2015.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[15] F. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3355–3362, Oct 2018.

[16] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.

[17] H. Lin, T. Zhang, Z. Chen, H. Song, and C. Yang, "Adaptive fuzzy gaussian mixture models for shape approximation in robot grasping," *International Journal of Fuzzy Systems*, vol. 21, 02 2019.

[18] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3304–3311, May 2011.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2027–2034, 2013.

[21] J. Zhang, M. Li, Y. Feng, and C. Yang, "Robotic grasp detection based on image processing and random forest," *Multimedia Tools and Applications*, pp. 1–20, 2019.