2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Object Detection Approach for Robot Grasp Detection

Hakan Karaoguz and Patric Jensfelt

***Abstract*— In this paper, we focus on the robot grasping problem with parallel grippers using image data. For this task, we propose and implement an end-to-end approach. In order to detect the good grasping poses for a parallel gripper from RGB images, we have employed transfer learning for a Convolutional Neural Network (CNN) based object detection architecture. Our obtained results show that, the adapted network either outperforms or is on-par with the state-of-the art methods on a benchmark dataset. We also performed grasping experiments on a real robot platform to evaluate our method's real world performance.**

I. Introduction

Robot grasping is a well-known and widely studied problem. With the recent advancements in robot technology, the robots are now being deployed in real and unconstrained environments. Thus, the ability to perform successful grasps on a variety of objects is crucial for a robot to be able to accomplish useful tasks such as goods delivery, elderly care and rescue operations. However, despite being such a popular research topic, it is largely an unsolved problem due to constraints in several relevant areas such as perception, motion planning and end effector design.

In order to perform successful grasping with a robot the following steps need to be followed [1]:

- Grasping pose detection
- 6-D grasping pose estimation
- Motion planning
- Motion execution

The first step which is the grasp pose detection is a perception problem in which, based on the sensory data, an optimal grasping pose for the end effector is predicted. For this task, mostly RGB-D cameras are used [2]. When the optimal grasping pose is calculated in the image plane, it is transformed into world coordinates using image to world transforms. Afterwards, motion planning is performed and the grasping motion is executed. Fig. 1 shows such an example grasping scenario. Here, as seen in the top-left figure, an object lays on a surface in the workspace of the robot. On the top-right figure, the optimal grasping pose in the image plane is described using the *grasping rectangles* representation [3], [4]. This representation is formulated by five parameters $\mathcal{G} = (x, y, w, h, \theta)$. (x, y) represent the center coordinates of the rectangle, while w, h represent the width (red edges) and height (blue edges) (see Fig. 1 top right) of the rectangle. The position of the gripper fingers correspond to the height of the rectangle while the gripper opening is denoted by the width. The angle of the rectangle w.r.t the horizontal axis θ represents the wrist orientation of the gripper. In the final stage, motion planning is performed and the grasping motion is executed by the robot.

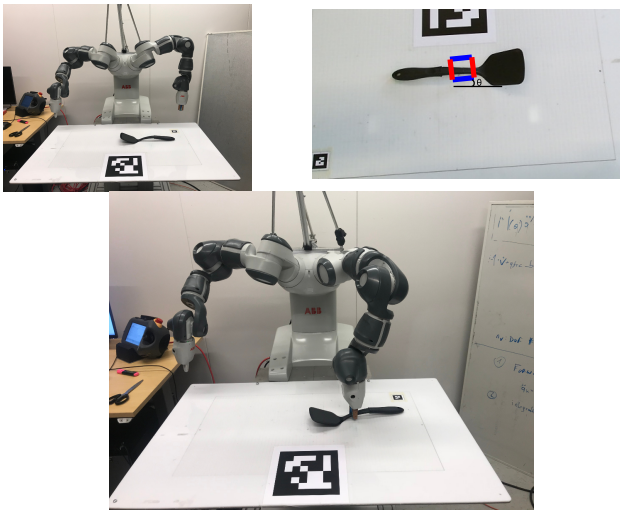

Fig. 1: An example of a robot grasping an object using grasping rectangle. Top left: Robot setup with the object in its workspace. Top right: Using the overhead camera image, a grasping rectangle for the object is proposed. Bottom: The robot executes the grasping motion based on the proposed grasping rectangle.

In order to solve the perception problem for grasping, several approaches have been used such as object segmentation, classification, object tagging etc. However, none of these approaches were generalizable to a large variety of objects.

Recently, deep learning based approaches are becoming popular for solving the robot grasping problem [1]. These approaches seem more promising compared to the previous methods as the initial results indicate that they generalize better for a given task. For example, Levine et al. [5] use a deep neural network for learning the end-to-end mapping from pixel coordinates to actual grasping motion execution.

However, deep neural networks require vast amount of data during training. Thus, most of the deep learning related research for grasping focus on simulations. One exception is [6] where Mahler et al. uses a synthetic point cloud dataset to train a deep neural network for predicting the grasp success from depth images. This trained network is successfully tested on a real robot.

In order to decrease the need of large datasets for deep

[1]All authors with is with School of Electrical Engineering and Computer Science, Royal Institute of Technology KTH, SE-100 44 Stockholm, Sweden hkarao,patric@kth.se

neural networks, transfer learning [7] and domain adaptation [8] became active research areas. The idea in these approaches is that instead of learning everything from scratch, previously learned knowledge in one domain is utilized for another domain. As such, the new task can be learned with minimal amount of training data. Utilizing the previous knowledge is indeed a very important aspect since it can lead to more generalized machine learning methods that can be used in various other tasks.

In this work, we exploit the idea of transfer learning and employ a CNN which is trained originally for textbox detection in images to detect the grasping rectangles [4] for unknown objects. The primary contribution of our approach is that we successfully perform transfer learning between object detection and robot grasp detection tasks and outperform the previous state-of-the art methods on the standart benchmark dataset with RGB modality. Secondly, we tested this method on a real-robot platform in order to evaluate whether our approach can be generalized for different real world settings.

II. Related Work

A lot of research is being conducted in order to solve the robot grasping problem. However, finding a general solution to this problem is challenging since it involves research in many different areas and current limitations of the grasping hardware makes the evaluation of the proposed methods difficult. As mentioned by Bohg et al. [9], grasping approaches are broadly divided into two categories. The first one is the analytical approaches, in which precise modeling of the gripper and the object is required. For example, [10] extends the concept of two-finger caging to multi-finger grippers for guaranteeing object trapping. Krug et al. [11] propose an efficient algorithm for computing the contact regions when the initial guess for the grasping points and 3-D object model is given.

In data-driven approaches on the other hand, the idea is to learn to propose good grasping configurations for any kind of object by either using an external user feedback [12] or annotated training data [3]. Among the proposed candidates, the best grasping candidate is selected based on some heuristics or measures. In [12], the grasping configurations for unknown objects are synthesized based on the best matching object with a known grasp configuration. The grasp configurations of the known objects are taught to the robot by kinesthetic teaching. The grasp selection is improved over time by using the information from previous grasp attempts. Jiang et al. in [3] propose the *grasping rectangles* representation on the camera image plane for identifying good grasping poses. Using this representation, they propose a two step algorithm in the image feature space to automatically find the best oriented grasping rectangle for any given object.

Recently, deep learning based methods are developed for solving the grasping problem [1]. Some of these approaches employ deep learning for predicting the best grasping pose using sensory data. Lenz et al. [4] use annotated images to determine the best grasping pose. In their approach, each object in the image is annotated by rotated rectangles for determining the good and bad grasping poses. This data is released as Cornell grasping rectangles dataset to serve as a benchmark for similar approaches. Using this data, they train a two-stage deep neural network that outputs the top scoring grasping rectangle. The first stage of the network learns to propose good grasping candidates. While the second stage learns to refine the obtained candidates and output the best grasping rectangle. Similarly, Redmon et al. [13] use the same dataset and propose three different deep neural network architectures to identify the good grasping candidates. Their obtained results outperform the approach given in [4]. However they do not provide any results with an actual robot. Kumra et al. [2] report the state-of-the-art results on Cornell grasping dataset by using one of the recent CNN networks called ResNet-50 as their backbone. They have proposed different shallow architectures on top of ResNet-50 for unimodal (RGB only) and multimodal (RGB-D) data. Their results show that, the proposed unimodal and multimodal architectures achieve the state-of-the-art results on this benchmark dataset. Guo et al. [14] also propose a deep neural network architecture for detecting grasping rectangles from images. However their approach can only propose horizontally aligned rectangles which limits its usability for rotated objects. Mahler et al. [6] propose a method that predicts the robustness of the grasp candidates. In their approach a grasp location is defined by three parameters, which are the grasp center coordinates and the vertical gripper angle. They use millions of synthetic point clouds to train a deep neural network for that learns to evaluate the robustness of a grasp candidate from depth images. In their method, parallel grippers are used and the 3-D model of the gripper is assumed to be known beforehand. They validated their approach on a real robot and achieved over 90% grasping success.

In some other approaches, deep learning methods are used in the context of reinforcement learning to learn grasping through trial and error. As an example [5] use a deep neural network for learning the end-to-end mapping from pixel coordinates to actual grasping motion execution. During training, the robot learns the hand-eye coordination for grasping by means of reinforcement learning. In order to perform enough trials in feasible time, they run the learning algorithm on multiple identical robotic manipulators in parallel. However, setting up and running this kind of experimental platform is expensive and time consuming.

In our work, we propose a deep learning based approach to efficiently predict good grasping poses by means of *grasping rectangles* using RGB images. Similarly to the previous work, [4], [13], [2] we evaluated our approach on the benchmark Cornell grasping rectangle dataset. Our main contribution in this work is, as opposed to the previous deep learning based approaches, we explicitly employ a deep neural network originally designed for object detection to predict grasping rectangles. As mentioned in [4], grasping is essentially a detection problem. Thus, unlike previous

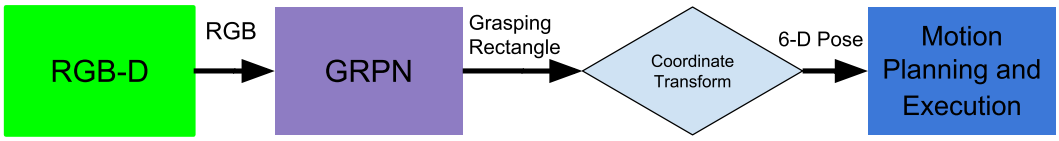

Fig. 2: Overall diagram of the Robotic Grasping approach.

approaches, we can utilize the deep neural network without any modifications since object detection networks are validated architectures that naturally output bounding boxes and associated class probabilities. This simplifies the training and evaluation of the method. Secondly, we tested our method on a real robot platform to evaluate its real-world performance. Finally, we have investigated data augmentation for increasing the performance of the network.

III. Background

Before describing our method we provide some background to the field of object detection, which we will draw from in our work. Object detection is a very popular and extensively studied topic among the computer vision community. Almost all the state-of-the-art methods for object detection employ deep learning [15], [16], [17], [18]. These methods have two approaches for detection. Single stage approaches such as [17] use convolutional feature maps to predict bounding boxes and class probabilities in one forward pass. The convolutional feature maps at the deeper stages act as artificial grids on the real image and the center points for bounding box predictions. The two-stage approaches [15], [16], [18] on the other hand use a shallow network called Region Proposal Network (RPN) that is shown in Fig. 3 in the first stage to propose bounding boxes on the image that can possibly contain objects. In the second stage, the classification and bounding-box regression is applied on these bounding boxes to detect the objects. Although they are slower compared to one stage approaches, they achieve the state-of-the-art results on benchmark datasets [18]. For both kind of approaches, methods like Non-Maximum Suppression are used to discard redundant bounding boxes.

The output of these networks naturally suits for grasping rectangle detection task so there is no need for modifications in the network architecture.

IV. Method

In this section we describe our method and begin with a brief overview. In our approach, we assume the robot has a parallel gripper and a transformation from image coordinates of the camera to the world coordinates exist. In the first step, a camera image is obtained and feed into a novel network which we call GRPN. GRPN does a forward pass to compute possible grasping rectangles and their goodness in terms of class probability. In the second step, the predicted grasping pose with the highest probability is sent to the motion planner of the robot. The overall diagram of the approach is given in Fig. 2.

A. Grasping Rectangle Proposal Network GRPN

We use the Rotated Region Proposals Network (RRPN) [19] as the backbone of our approach. RRPN is originally developed from a two-stage object detection network Faster-RCNN [15].

The regular Faster-RCNN is designed to predict only horizontally aligned bounding boxes. The assumption of having vertically or horizontally straight objects mostly holds in computer vision benchmarks. However, for real world scenarios such as detecting buildings from a flying drone or predicting grasping rectangles for arbitrary placed objects, the assumption of horizontally aligned objects no longer holds. Therefore, it is required to propose rotated bounding-boxes. Jianqi et al. [19] have developed the RRPN network to overcome this issue. In their approach, the goal is to predict textboxes in the wild. Therefore, on top of the regular Faster-RCNN, they added an angle component in RPN to be able to generate rotated bounding boxes. Thus, RRPN can be used in more general object detection scenarios including rotated objects.

The RRPN architecture uses a 16 layer VGG-16 network pretrained on ImageNet as the backbone. Since the network naturally outputs rotated bounding boxes with class probabilities, there is no need to modify the overall architecture. As a result, our employed network is simpler to train and much more memory efficient compared to the network proposed in [2].

In our method, we have adapted RRPN to our grasping rectangle detection task by means of transfer learning. We call this adapted network the Grasping Rectangle Proposal Network (GRPN). Although textboxes and grasping rectangles are two very different domains, our experiments show that the previously encoded knowledge in the network is very useful when adapting it to the grasping rectangle proposal domain.

B. Implementation

We have implemented our method for evaluating it in a real-world scenario. For this, we have used the 2-arm Yumi platform with Kinect camera (shown in Fig. 1 top left). In order to simplify the actual implementation, we have implemented vertical pinch grasps with the robot. During execution, the RGB data coming from the overhead Kinect sensor is fed into the GRPN and the top-scoring grasping rectangle is used as the target grasping location. Using the center point and angle of the predicted rectangle, the robot arm is positioned above the grasp location. At the last step,

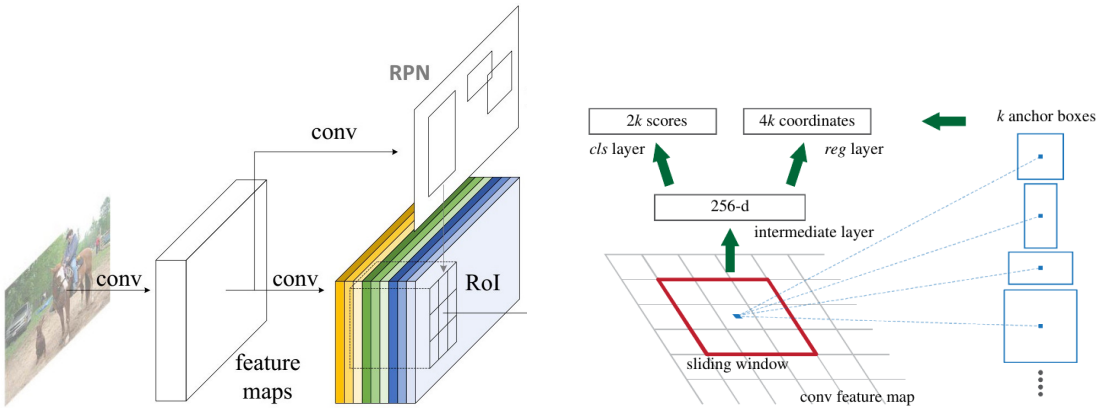

Fig. 3: Architecture of a two-stage object detection network [15], [16]. Left: Additional RPN layer on top of a CNN backbone for proposing object regions in an image. Right: The inner structure of the RPN. A sliding window and associated anchor boxes are used on the convolutional feature map to propose bounding boxes and associated class probabilities.

grasping motion is planned and executed to lift the object from the workspace. For our implementation, we have used MoveIt! [20] library for motion planning and execution.

V. Experimental Results

Our method is evaluated both on a benchmark dataset and a real robot scenario. For testing the grasping location detection performance of GRPN with respect to the state-of-the-art methods, we have used the Grasping Rectangles dataset [4].

A. Cornell Grasping Rectangles Dataset

Cornell Grasping Rectangles Dataset [4] consists of 885 images of single objects on a white background. Each object is annotated with positive and negative grasping rectangles. Positive rectangles indicate favorable grasping locations while the negative rectangles indicate the opposite. Fig. 4 shows example images from the dataset. For this dataset, it is assumed that the robot has parallel grippers. For each grasping rectangle, the angle of the longer edges determine the vertical orientation of the end effector (perpendicular to the workspace) while the shorter edges indicate the gripper locations.

In order to fully evaluate the performance of our approach, we have randomly shuffled the dataset three times into 70% training and 30% test samples which results in 620 training images and 265 test images at each shuffle. The reported results are obtained by averaging over the performance on these three shuffles. Moreover, we have used data augmentation strategy to increase the training data and evaluated its effect on the performance.

During training, for the networks without data augmentation we have used SGD with learning rate:0.01, step size:7000, gamma:0.1 and momentum:0.9. Networks without data augmentation are trained for a maximum of 15000 iterations. Snapshots of the network were taken at every 5000 iterations. We limited the maximum number of iterations to 15000 to avoid overfitting to the small training set. For the network that uses augmented data, we have used SGD with

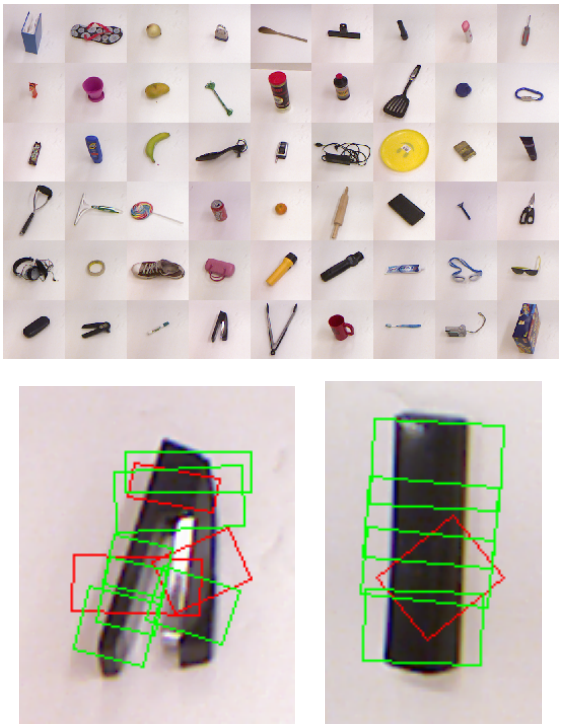

Fig. 4: Example images from Cornell Grasping Rectangles Dataset [4]. Top: Different objects in the dataset. Bottom: Example annotations. Red color represents negative grasping rectangles while green represents positive rectangles.

parameters learning rate:0.01, step size:10000, gamma:0.1 and momentum:0.9. That network was trained for 25000 iterations. For the classification phase of the detection, we only used positive rectangles as positive samples. Thus, all the networks are trained for binary detection of positive rectangles and background.

1) Data Augmentation for Increasing Training Samples: Data augmentation is a common approach for increasing the number of training images by applying various im-

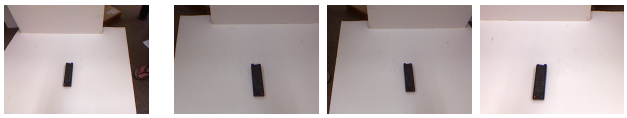

Fig. 5: Example augmentation set. Leftmost image is the original data while the remaining images show three augmented examples with random horizontal flip, brightness change and zoom.

age transformations. The most common transformations are random crop, rotation, translation, horizontal/vertical flip, illumination change and noise addition. While applying data augmentation, the crucial thing is to use representative augmentations. As an example, while augmenting a car image, it is not favorable to use vertical flips since observing an upside down car is a very rare case.

In our approach, we have applied a combination of augmentations which involved random horizontal flip, random illumination change and random zooming at a rate within 10-50% of the original image size. Redmon et al. [13] similarly applied data augmentation to this dataset but at a very high rate (3000 augmentations per original image). In our case, we only had 3 augmentations per original image. An example augmentation result is shown in Fig. 5.

The obtained results with GRPN are given in Table I. Here GRPN+ denotes the network trained with augmented training data while the numbers represent the number of training iterations. We have used the evaluation metrics given in [4] which are point and rectangle metrics. Some example results obtained with GRPN is given in Fig. 6.

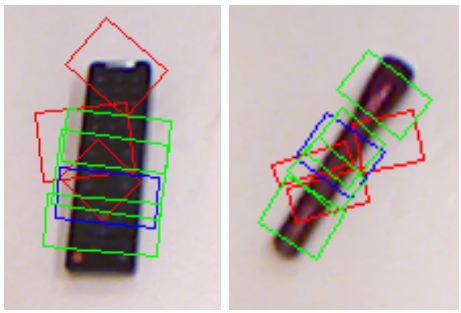

Fig. 6: Example outputs of GRPN. Red rectangles represent negative grasping locations, green rectangles represent positive grasping locations and blue rectangle represents the top-scoring proposed rectangle.

2) Point Metric: The point metric compares the distance between the center locations of the ground truth positive rectangles and the proposed rectangle. If the distance is less than a certain threshold, it is accepted as correct proposal. However in [4], distance threshold used is not explicitly given. Because of that [13], [2] skipped this metric for evaluation. Instead of avoiding this metric, we first determined the approximate metric size of each image pixel using the actual object sizes in the dataset. Then we have used two different distance thresholds to get a better idea about the method's performance. According to [21], humans have about 1.5 cm positional error while grasping. Thus, we first used a 1.5 cm error threshold. If we look at the results at Table I for this threshold, GRPN+ achieves the best performance with 85.9% average accuracy. This strong performance is probably due to the zoomed samples in the augmented data that help the network to generalize better. If we relax the threshold to 3 cm, the top performance is achieved by GRPN15000 with 97% accuracy. The minimum accuracy under 3 cm is above 92% for all GRPN models. As mentioned before, the latest state-of-the-art approaches either don't use this metric or do not mention the used distance threshold but our method achieves a strong performance for this metric even with a very low error threshold.

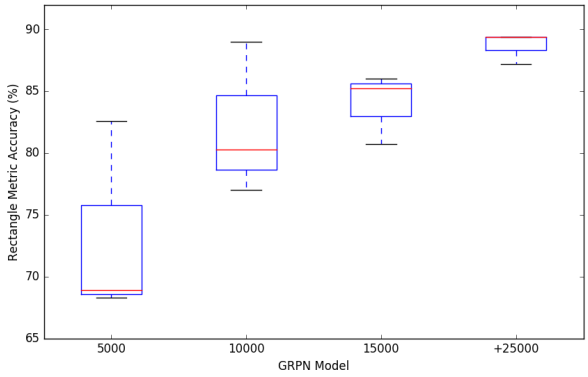

Fig. 7: Comparison of GRPN Models' rectangle metric performance over each data shuffle. x-axis represents the different GRPN models while y-axis represents the rectangle metric accuracy for different data shuffles.

3) Rectangle Metric: The rectangle metric is more complete compared to point metric, because it also considers the angle and size of the proposed rectangle. In this metric, the ground truth positive rectangles and the proposed rectangle is compared using Intersection over Union (IoU). During our evaluations we have used the same thresholds given in [4] for angular difference (30°), and IoU (0.25). As seen from the results in Table I, GRPN+ achieves the top accuracy (88.7%) in this metric among the models that use RGB modality while Kumra et al. [2] achieve slightly better accuracy (89.2%) with their multimodal model. It is also important to mention that GRPN+'s performance is consistent between different data shuffles as seen in Fig. 7.

Thus, despite having a simpler architecture compared to [2] and not being as engineered as Redmon's [13] top scoring method, our method achieves the state-of-the-art performance in RGB modality in a consistent manner.

We also observed that, more training data and more iterations generally improves the network's success rate and consistency as seen in Fig. 7. Here the red lines represent the median of the data, while the head and tail lines represent the maximum and minimum values obtained. The top and bottom edges of the box represent 3^{rd} and 1^{st} quartiles respectively.

TABLE I: Cornell Grasping Rectangles Dataset Results

Approach	Point (%) <1.5 cm	Point (%) <3 cm	Rectangle (%)
Lenz et al. [4]	88.4		73.9
Redmon et al. [13] (RGB-D)	N\A		88.0
Kumra et al. [2] (RGB)	N\A		88.4
Kumra et al. [2] (RGB-D)	N\A		**89.2**
GRPN 5000 (RGB)	73.7	92.9	73.3
GRPN 10000 (RGB)	79.8	94.7	82.1
GRPN 15000 (RGB)	83	**97**	84
GRPN+ 25000 (RGB)	85.9	96	88.7

It is observed that when GRPN is trained for only 5000 iterations without data augmentation there is a large performance variance among different shuffles. On the other hand, if we look at the performance of GRPN+, it can consistently achieve high accuracy among different data shuffles. This result also validates the importance of data augmentation when it is correctly used.

In terms of computational performance, it is hard to directly compare these three methods since all of them are tested with different computer configurations. Nevertheless, Lenz et al. [4] mentions 13.5s per frame processing time while Redmon et al. [13] mentions 76ms per frame. Our method requires 0.5s per frame on average while running on a Nvidia GTX 1070 GPU. Thus, although our method is not the fastest, it still has acceptable performance for real-time deployments since the robot's grasping action takes on the order of several seconds.

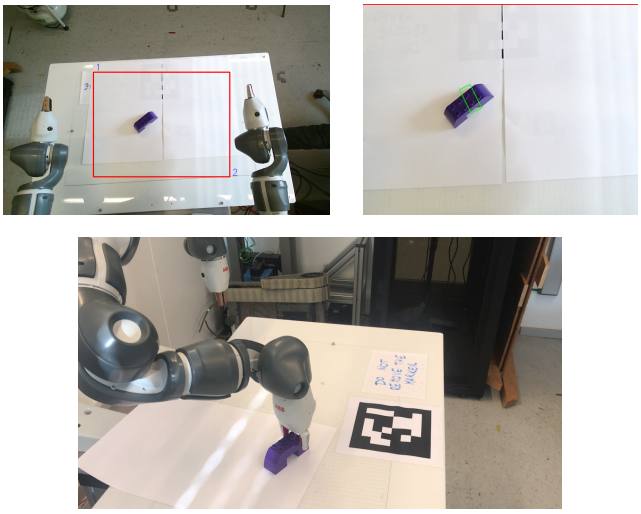

Fig. 8: GRPN based graspng with Yumi. Top Left: The workspace view of the robot. Top Right: Proposed grasping rectangle by GRPN. Bottom: Grasping action performed by Yumi.

B. Yumi Robot

In order to see the real-world performance of our method, we have tested our approach on a Yumi robot. During our tests, we have placed different objects (one at a time) which were not present in the training dataset, in the workspace of the robot. Our pipeline processed the image frame coming from overhead Kinect sensor, determined the grasping pose and performed the grasp. An example grasping execution for a Lego brick is given in Fig. 8. We have tested our approach with 5 different objects, which are pan, Lego brick, screwdriver, mug and a plastic cup. For all of the objects, the robot was able to place its grippers on the right position. It successfully lifted 3 of them while the plastic cup was missed because of the space between fingers. In the mug case, a successful grasp was performed but the payload of the arm was exceeded during lifting. Thus the robot stopped during motion execution. During these experiments, we also observed that the grasping rectangles predicted by the system were consistent for these objects in the different parts of the workspace.

VI. Conclusion

In this paper, we have proposed a robot grasp detection approach that can work with any kind of object that can be graspable with parallel grippers. For this task, we first adapted an object detection network to robot grasp detection domain using transfer learning. The resulting network, named GRPN, learned to predict grasping rectangles for any kind of object that can be graspable with parallel grippers. Since the object detection network naturally outputs bounding boxes, no change in the architecture is required for obtaining the right output. We have extensively evaluated our method both on a benchmark dataset and a real-robot system. Our obtained results show that GRPN either outperforms or is on-par with the previous state-of-the-art methods with a much simpler network architecture and less amount of data. We have also observed that, data augmentation methods, when correctly used, help to improve the system's performance and consistency. As future work, we want to improve GRPN by adding the depth information as a new modality and evaluating different training settings.

Acknowledgment

This work is supported by the European Unions Horizon2020 research and innovation program under grant agreement No. 644839 (CENTAURO) and SSF FACT projects.

REFERENCES

[1] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, 2018.

[2] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 769–776.

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

[4] 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.

[5] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 173–184.

[6] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[8] J. Jiang, "A literature survey on domain adaptation of statistical classifiers," *URL: http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey*, 2008.

[9] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[10] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012.

[11] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev, "On the efficient computation of independent contact regions for force closure grasps," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 586–591.

[12] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal, "Learning of grasp selection based on shape-templates," *Autonomous Robots*, vol. 36, no. 1-2, pp. 51–65, 2014.

[13] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1316–1322.

[14] D. Guo, F. Sun, T. Kong, and H. Liu, "Deep vision networks for real-time robotic grasp detection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 1, p. 1729881416682706, 2016.

[15] 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 (NIPS)*, 2015.

[16] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[17] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[18] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[19] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Transactions on Multimedia*, 2018.

[20] S. Chitta, I. Sucan, and S. Cousins, "Moveit!" *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[21] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.