Review of Object Manipulation Approaches: GGCNN and GR-ConvNet

Bernard Renardi
Faculty of Science and Engineering
Rijksuniversiteit Groningen
Groningen, The Netherlands
b.jo@student.rug.nl

Duygu Bayram
Faculty of Arts
Rijksuniversiteit Groningen
Groningen, The Netherlands
d.bayram.1@student.rug.nl

***Abstract*—This paper presents a review of two different approaches of Convolutional Neural Network (CNN) algorithm for object manipulation. This paper presents the evaluation results of Grasping Convolutional Neural Network (GG-CNN) and Generative Residual Convolutional Neural Network (GR-ConvNet) by literature and experiments review through several scenarios, which can be measured by $success\_rate$ and $object\_removed\_rate$. In literature review, we can pinpoint a different network adaptation of five identical residual block implemented in between convolutional block of GR-ConvNet, enabling accurate grasping and reduce inaccuracies due to vanishing gradient problem. By experiment, we achieve the highest $success\_rate$ of 89.2% from isolated scenario in GGCNN and $object\_removed\_rate$ from packed scenario in GR-ConvNet with a percentage of 98.0%.**

***Index Terms*—CNN, GGCNN, GR-ConvNet**

I. Introduction

Traditionally robotic manipulation has mostly considered repetitive tasks performed in tightly controlled spaces. However, recently there has been a lot of interest for deploying robots to domains that require more flexibility. Due to these challenges, recent research in grasp synthesis has overwhelmingly favored data-driven approaches to plan grasps directly from sensor data, outperforming manually designed policies. This limits the flexibility of the system, as in some cases it is easier to approach different objects in the scene from different directions.

Most recently, deep learning techniques have enabled some of the biggest advancements in grasp synthesis for unknown items. These approaches allow learning of features that correspond to good quality grasps that exceed the capabilities of humandesigned features by typically use adapted versions of Convolutional Neural Network (CNN) architectures designed for object recognition. From several publications, there are two leading grasping approaches based on their CNN architectures and their inputs, namely Grasping Convolutional Neural Network (GG-CNN) which is proposed by [1] and Generative Residual Convolutional Neural Network (GR-ConvNet) by [2]. GGCNN predicts the quality and pose of grasps at every pixel from a depth image as the input to overcomes limitations of common deep-learning grasping techniques by avoiding discrete sampling of grasp candidates then resulting shorter computation times. On the other hand, GR-ConvNet that generates antipodal grasps for every pixel in the input image not by calculating the best grasp probability by order, but directly inferring multiple grasp rectangles even for multiple objects from the output of GR-ConvNet at once based on inference module, thereby decreasing the overall computational time.

The main aim of this review is to present an experimental approach of evaluating latest approaches of object grasping to make a good reference for researchers and students to have a clear image of both methods. Our contributions are outlined as follows:

- This review helps researchers and students to have a good understanding about GGCNN and GR-ConvNet papers by checking the provided code and explaining how these systems work.
- We evaluate and analyze their performance in isolated, packed, and pile scenarios by measuring $success\_rate$(1).

$$success\_rate = \frac{\#successful\_grasps}{\#attempts} \tag{1}$$

II. Related Work

Object grasping requires the understanding of which parts of an object would lead to successful grasping and how the robot needs to move to perform this action. This comes with several complex sub-tasks, with object detection being an important one. However, it is difficult for computational models to generalize what is learned during the training to unseen objects, making it harder to create robust and fast object grasping systems. In order to improve performance in this area, it is important to optimize the variety of input we actually need. Some features to consider could be color, shape, texture, depth, and others. As such, there are several models approaching this problem differently.

Analytical approaches make use of mathematical modeling and geometry, and use physics to estimate grasping methods [3] [4]. However, they run into the issue of generalization as these are hard to transfer to real-world environments [3] in terms of physical and mathematical modeling. Empirical methods, in a traditional sense, exploit experiences to train the model to detect good grasping points, based on shapes [5], object classes [6] or object parts [7]. Despite improvements,

these models run into generalization issues as well, as they cannot perform well on unseen objects.

To tackle this issue, engineers have turned to deep learning methods [8] [9], often CNNs. This results in an issue with speed as the computing time for grasping action can be long due to the large amount of parameters [10] in deep learning models. A way of handling this is to avoid trying to account for all possible grasping instances, and instead preprocessing or pruning some alternatives [8] or running prediction tasks on sets of alternatives at once [10]. Other CNN-based approaches to grasping have been adopted, including increasing data size [10], shape completion via 3D CNN [11], adding tactile features [12], grasp prediction [13], and use of semantic object parts [14]. Some recent work has experimented with closed-loop approaches as opposed to open-loop grasping [15].

We investigate two models, GR-ConvNet [2] and GGCNN [1], with a main difference of input types. While both models take depth input, GR-ConvNet additionally accepts RGB input. We then contrast the performance of these models on three different object scene scenarios.

III. Methodology

For the following experiments, we utilize the PyBullet as the simulation environment. We also use Ubuntu 18.04 for our operating system. Having all these requirement fulfilled, we can finally setup our virtual machine (VM) on the computer.

A. Installation and Setup

All required steps have been explained systematically on the provided classroom Github[1]. There are several parts for this step including:

- Clone and Compile ROS packages
- Add the folder as workspace
- Set up all dependency of packages
- Compile all packages

B. GR-ConvNet Simulation

The first simulation is to run perform a simulation with GR-ConvNet as the network by running the $simulation.py$ script. The simulation has some setup parameters we need to set including:

$scenario$	: isolated / packed / pile
$network$	: GR_ConvNet
$runs$	: 10
$save-network-output$	: True

By the end of the simulation, we will obtain summary of 10 results for each grasping scenario in GR-ConvNet. For a represenation of the scenarios, please see Fig 1, as taken from the assignment sheet.

[1]https://github.com/SeyedHamidreza/cognitive_robotics_manipulation

Fig. 1: Example visuals of isolated, packed, and pile scenes (left to right).

C. GGCNN Simulation

In order to implement GGCNN network in the $simulation.py$, two files must be modified to enable GGCNN as the option. In $grasp\_generator.py$, we add more option for the network in $predict$ function, while in $simulation.py$, we add pretrained model path for GGCNN network available on Github[2]. We need to set parameter $scenario$ to GGCNN and run the experiment as much as the previous simulation. In the end of this experiment, we will also obtain summary of 10 results for each grasping scenario in GGCNN.

IV. Results and Discussion

After obtaining the results from both experiments, we conduct analysis into the two separate sections to compare both approaches: (1) Literature Review and (2) Experiments Review.

A. Literature Review

Grasp Representation: both of the approaches have similar representation used to define the problem of robotic grasping as predicting grasps for unknown objects from an image of the scene and executing it on a robot known as a pose.

From reference [2], the grasp pose in robot frame can be denoted as

$$G_r = (\mathbf{P}, \Theta_r, W_r, Q) \tag{2}$$

where $\mathbf{P} = (x, y, z)$ is gripper's center position, Θ_r is tools rotation around the z-axis, W_r is the required width for the tool in the range of [0, W_{max}], and Q is the grasp quality score. W_{max} is the maximum width of the antipodal gripper.

Network Architecture: GGCNN takes an inpainted depth image as the input with depth. Then, the data is fed into 3 convolutional layers with the size of 9x9, 5x5, and 3x3 respectively based on the model shown in this code[3] from line 5, 16-18. As described in [1], the network then produces three different outputs representing the unit vector components of Q, Θ and W in a form of 300×300 pixel image. The final model of GG-CNN contains 62,420 parameters in total.

On the other hand, GR-ConvNet takes a 224 × 224 n-channel input image with RGB and depth information is fed into 3 convolutional layers with the size of 9x9, 4x4, and 4x4 respectively plus additional 5 identical 3x3 residual block

[2]https://github.com/dougsm/ggcnn

[3]https://github.com/dougsm/ggcnn/blob/master/models/ggcnn.py

modules between the last convolutional layer and its transpose layer based on its network model shown in this code[4] from line 11, 14, 17, 20-24. This network also produces three different outputs and has a total of 1,900,900 parameters.

Training Dataset: To train GGCNN network, the Cornell Grasping Dataset is adopted to the learning setup. The Cornell Grasping Dataset itself contains 885 RGB-D images of real objects, with 5,110 human-labelled positive, 2,909 negative grasps, and 1,710 augmented images.

For GR-ConvNet, the network is trained by Cornell Grasping Dataset, with additional 51,000 grasp examples from augmentation. This network also use Jacquard Grasping Dataset as training dataset, consists of 54,000 RGB-D images and annotations of successful grasping positions based on grasp attempts performed in a simulated environment. In total, it has 1.1 millions grasp examples with no augmentation on this dataset.

Overall Review: In theory, despite their kernel sizes, implementing residual block on a network will have some advantages while working on a large number of convolutional layers. A CNN with large number of layers can somehow loose their training ability due to vanishing gradient problem, which is caused by involving gradient based learning so the network actually stops learning when the gradient is too small. Not only residual block can tackle this problem, but also will not increase the error percentage due to its identity mapping.

Another point is the number of input data. We know that GGCNN utilizes only depth image data to produce the prediction output, while GR-ConvNet also includes depth and RGB data of the image in the form of tensors to its network. Intuitively, by feeding more data to the network to learn, we can expect a higher accuracy of its prediction output which makes GR-ConvNet theoretically outperforms GGCNN.

Fig. 2: Examples of what the model sees during grasping for scenarios isolated, pile, packed top to bottom, and GGCNN and GR-ConvNet, on the left and right respectively.

B. Experiments Review

We implemented the GGCNN and GR-ConvNet models to compare their results in term of *Success Rate* and *Object Removed Rate*. The summary of the results can be seen in TableI. For a visual representation of what the model sees during grasping, refer to Fig 2.

We found that the models showed different performance rates in packed and isolated scenarios. While GR-ConvNet was a better fit for the packed scenario with a 65.8% Grasping Success Rate (46.5% for GGCNN), GGCNN outperformed GR-ConvNet in grasping isolated objects with a success rate of 89.2%, compared to GR-ConvNet's 85.9%.

As discussed, the main difference between the presented models is GR-ConvNet's inclusion of RGB data. We think that reducing information for grasping helps in isolated instances as there is only one object, resulting in better model performance. In the case of multiple objects, this reduction becomes a hinderance for GGCNN as there are many objects, and color potentially becomes important information to differentiate between objects and their grasping points for successful manipulation. We can further see that GGCNN has a noticeably increased difference in grasping attempts in the packed scenario (109 attempts compared to GR-ConvNet's 76 attempts). For isolated instances, it performs better, possibly because the additional color information serves as some noise for this task.

On the other hand, we come across interesting results when the robot is presented with a pile of objects. While the GGCNN has a higher success rate (60.3% as opposed to the 46.9% for GR-ConvNet), the rate of removed objects is lower in contrast (2 items less). It seems that GGCNN requires less attempts to successfully grasp an object, but is perhaps slower than GR-ConvNet, or grasps less objects at a time. On the other hand, GR-ConvNet attempts grasping more but succeeds in removing more objects at the same time. In this case, the color information and the open-loop approach seems to increase model speed, although the gap between the *Objects Removed Rate* is not as large as the *Grasping Success Rate* between these two models. As a result, we conclude that GGCNN might still be a better fit for this task. Furthermore,

[4]https://github.com/skumra/robotic-grasping/blob/master/inference/models/grconvnet4.py

Method	Scenario	Manipulation Success Rate	Grasping Success Rate	Object Removed Rate
GGCNN	packed	0.413 (45/109)	0.468 (51/109)	0.9 (45/50)
GR-ConvNet	packed	**0.645 (49/76)**	**0.658 (50/76)**	**0.98 (49/50)**
GGCNN	pile	**0.575 (42/73)**	**0.603 (44/73)**	0.84 (42/50)
GR-ConvNet	pile	0.449 (44/98)	0.469 (46/98)	**0.88 (44/50)**
GGCNN	isolated	**0.862 (144/167)**	**0.892 (149/167)**	-
GR-ConvNet	isolated	0.825 (146/177)	0.859 (152/177)	-

TABLE I: Summary of results for GGCNN and GR-ConvNet on 10 runs of 3 scenarios: packed, pile, isolated.

in object piles, depth could be serving as a more important feature as opposed to packed scenarios.

V. Conclusion

We present our literature and experiment review of two notable approaches of object manipulation based on CNN (GGCNN and GR-ConvNet) on several scenarios of random object in PyBullet, and reported the important portion of our findings. We show through grasping experiments that GR-ConvNet performs better in packed scenario with higher $success\_rate$ of manipulation and also $object\_removed\_rate$ compared to GGCNN. However, on the other hand, we also find that GGCNN outperforms GR-ConvNet in pile and isolated scenario. In general, the highest $success\_rate$ comes from isolated scenario in GGCNN with a percentage of 89.2% and $object\_removed\_rate$ comes from packed scenario in GR-ConvNet with a percentage of 98.0%.

Authors' Contributions

Bernard Renardi: GR-ConvNet
Duygu Bayram: GGCNN (TA helped both of us, thank you)
Both authors contributed equally to writing.

References

[1] Morrison, D., Corke, P., Leitner, J. (2018). Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach.

[2] Kumra, S., Joshi, S., Sahin, F. (2019). Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network.

[3] Antonio Bicchi and Vijay Kumar. Robotic Grasping and Contact: A Review . In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 348–353, 2000.

[4] Domenico Prattichizzo and Jeffrey C. Trinkle. Grasping. In Springer Handbook of Robotics, chapter 28, pages 671–700. Springer Berlin Heidelberg, 2008.

[5] Corey Goldfeder, Peter K Allen, Claire Lackner, and Raphael Pelossof. Grasp Planning via Decomposition Trees. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 4679–4684, 2007.

[6] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic Grasping of Novel Objects using Vision. The International Journal of Robotics Research (IJRR), 27 (2):157–173, 2008.

[7] Sahar El-Khoury and Anis Sahbani. Handling Objects By Their Handles. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2008.

[8] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. The International Journal of Robotics Research (IJRR), 34(4-5):705–724, 2015.

[9] Joseph Redmon and Anelia Angelova. Real-Time Grasp Detection Using Convolutional Neural Networks. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 1316–1322, 2015.

[10] Lerrel Pinto and Abhinav Gupta. Supersizing selfsupervision: Learning to grasp from 50k tries and 700 robot hours. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 3406–3413, 2016.

[11] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 2442–2447.

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

[13] 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," arXiv preprint arXiv:1703.09312, 2017.

[14] L. Antanas, P. Moreno, M. Neumann, R. P. de Figueiredo, K. Kersting, J. Santos-Victor, and L. De Raedt, "Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach," Autonomous Robots, vol. 43, no. 6, pp. 1393–1418, 2019.

[15] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Large-Scale Data Collection. In International Symposium on Experimental Robotics, pages 173–184, 2016.