university of groningen / faculty of science and engineering

Learning to grasp objects in highly cluttered environments using Deep Convolutional Neural Networks

Bachelor's Project Thesis

Jeroen Oude Vrielink, s3416402, j.oude.vrielink@student.rug.nl,
Supervisor: Dr. Hamidreza Kasaei, hamidreza.kasaei@rug.nl,
Department of Artificial Intelligence

Abstract: This paper evaluates the performance of the Generative Residual Convolutional Neural Network (GR-ConvNet), proposed by Kumra *et al.* (2020), as a grasp inference method. The performance is evaluated by testing the network's predictive accuracy on a subset of the Cornell grasping dataset, and by conducting several rounds of the isolated, pack, and pile grasping experiments, as proposed by Kasaei *et al.* (2021), using a UR5e robotic arm. For this purpose, a simulation was developed in PyBullet (Coumans *et al.*, 2020). The GR-ConvNet achieved an accuracy of 97.7% on the Cornell grasping dataset. A grasping success rate of 91.9%, 73.2%, and 55.2% was achieved, and a target success rate of 91.9%, 73.2%, and 55.2% was achieved for the isolated, pack, and pile experiments respectively.

1 Introduction

Humans have to ability to quickly grasp and manipulate a wide variety of objects, regardless of whether they are complexly shaped, are only partly visible, or are situated in various clutter and occlusion environments. In recent years, there has been an increasing interest in the deployment of robots in human-centric environments. Whereas industrial settings mostly require repetitive grasping tasks, human-centric environments demand a more human-style grasping strategy that can adapt to various grasping scenarios, as previously described. Much research in this field has been done, and numerous methodologies have proven to be successful in a variety of grasping tasks, as demonstrated by Kasaei and Kasaei (2021); Morrison, Corke, and Leitner (2018); Lenz, Lee, and Saxena (2015), among others.

One of the state-of-the-art methods has been developed by Kumra, Joshi, and Sahin (2020). Their object agnostic approach proposes a Generative Residual Convolutional Neural Network (GR-ConvNet) that can predict the grasping quality, angle, and opening width of the gripper for every pixel in an n-channel input image. From the resulting three output images, multiple antipodal grasps can be inferred.

This paper aims to evaluate the performance of the GR-ConvNet as a grasp inference method by testing its predictive accuracy on the publicly available Cornell grasping dataset, as well as subjecting a UR5e robotic arm to three different grasping experiments, using the GR-ConvNet as a grasp inference method. For this purpose, a robotic simulation has been developed in PyBullet (Coumans and Bai, 2016–2020). The three grasping experiments, as proposed by Kasaei and Kasaei (2021), consists of one isolated object scenario, and two clutter scenarios. In the latter two scenarios, multiple objects are closely packed together and situated in a pile.

Figure 1.1 gives an overview of the full grasping system. The inference module acquires RGB and depth images obtained from an RGB-D camera with a top-down view of the object scene. After pre-processing, the images are passed onto the GR-ConvNet, which generates the grasping quality, angle, and width for every input pixel. The three output images are concurrently used to infer one or multiple antipodal grasps. The robotic arm can thereafter be instructed to execute the inferred grasps.

The remainder of this paper is organized as fol-

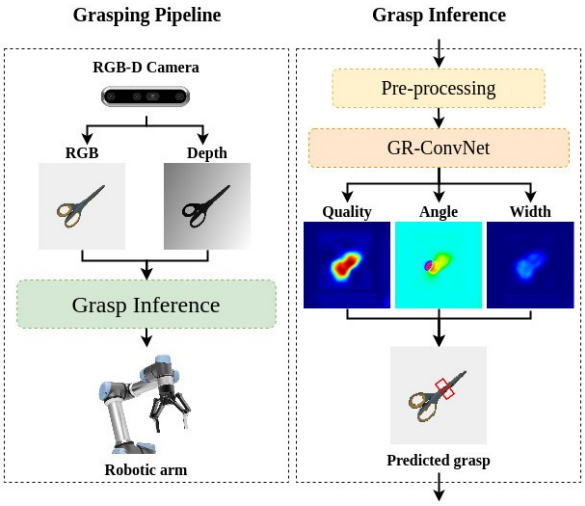

Figure 1.1: Grasping system overview - Left diagram illustrates the full grasping pipeline. Right diagram illustrates the grasp inference module.

lows: first, a brief overview will be given of the relevant research in the field of robotic grasping. Next, a formal problem formulation will be outlined, and the corresponding grasping methodology will be presented. In the following section, the evaluation of the methodology will be described in detail, and the results will be presented. The discussion will provide further interpretation of the results, which is followed by the conclusion.

2 Related work

Up until the beginning of the millennium, the field of robotic grasping has been mostly dominated by analytic grasp inference methods (Bohg, Morales, Asfour, and Kragic, 2014). Analytical approaches rely on kinematics and dynamics formulations in order to synthesize a grasp. For instance, Ding, Liu, and Wang (2000) designed an algorithm that can compute the desired pose for an n-finger gripper from an initial random grasp using a mathematical model of the target object. However, possible target objects for which this method works are limited to polyhedral shapes. This limitation highlights a weakness of analytic approaches. Namely, they often fail to find a mathematical formulation that generalizes to a variety of grasping tasks due to the mathematical complexity that arises from the many challenging constraints that need to be satisfied (Sahbani, El-Khoury, and Bidaud, 2012).

Empirical approaches aim to avoid the mathematical complexity analytical approaches face by using classification and learning methods. Some of these approaches include learning from human demonstration. A human teacher shows how a specific grasping task should be performed, after which the robot is able to perform the grasp by itself (Kasaei, Shafii, Lopes, and Tomé, 2019; Shafii, Kasaei, and Lopes, 2016; Fischer, van der Smagt, and Hirzinger, 1998; Ekvall and Kragic, 2004). However, these approaches are highly task-specific, and often fail to generalize to unseen objects, in a similar fashion to analytical approaches (Sahbani et al., 2012).

Ideally, grasp affordances should be predicted based on object agnostic features such that any given object can be grasped, regardless of its shape. This approach has become somewhat realizable with the rise of deep learning. Deep learning has demonstrated that it can *learn* useful features from large-scale datasets through its hidden layer architecture and has proven to be successful in numerous fields ranging from advanced text generation (Brown et al., 2020) to gene-based protein structure prediction (Senior, Evans, Jumper, et al., 2020).

Likewise, deep learning approaches have proven to be successful for grasp inference, as demonstrated by Lenz et al. (2015); Morrison et al. (2018); Kasaei and Kasaei (2021) among others. Research in this field can be categorized according to numerous characteristics, some of which will be discussed in more detail.

Uni- vs. Multi-modal data: Some approaches rely on a single input stream for grasp prediction. For instance, Johns, Leutenegger, and Davison (2016) trained a Convolutional Neural Network that can predict the quality score of every grasp pose across the image using a single depth image as input. The best grasp is then selected by smoothing the predicted pose using a grasp uncertainty function. Others take a multi-modal approach and rely on multiple input streams for grasp prediction. Yan, Khansari, Hsu, Gong, Bai, Pirk, and Lee (2019) use an object detection network to obtain the RGB, depth, and segmentation image of a given target object. With the use of a Point Cloud Prediction network, the three images are converted to a 3D point cloud, which is then fed to a critic network

to predict a grasp. Kumra and Kanan (2017) rely on RGB-D images to infer grasp poses. Their deep CNN architecture makes use of residual blocks, as proposed by He, Zhang, Ren, and Sun (2016). Their research demonstrates the advantages of using a deeper architecture along with residual blocks for better and faster feature learning. Likewise, Chu, Xu, and Vela (2018) use RGB-D images as input for grasp inference. Their research focuses on predicting multiple grasp poses for multiple objects in a single image.

Open- vs. Closed-loop control: A robot that is able to adapt to a dynamic environment has to continually observe its surroundings. In a grasping scenario, this would imply that a given target object is continuously assessed such that the predicted grasp can be adapted to changes in the object's pose. This is referred to as closed-loop control. Systems that predict grasp poses from a single image are referred to as open-loop control. The latter control method often results from computationally expensive grasp synthesis methods. If computation times take too long, objects cannot be continually assessed (Kasaei, Luo, Sasso, and Kasaei, 2021). For instance, Lenz et al. (2015) developed a two-step cascaded system with two deep networks. In this system, the most promising grasp predictions from the first network are re-evaluated by the second. Although they obtained good results, the computation time took $13.5s$ per generated grasp. Morrison et al. (2018) developed a Generative Grasping Convolutional Neural Network suitable for closed-loop grasping. The network predicts the grasping quality, angle, and width for every pixel from an input depth image. This one-to-one mapping allows for computation times of $19ms$ per generated grasp.

Uni- vs. Multi-directional grasping: Much of the aforementioned research is limited to 4 Degrees of Freedom (DoF) grasping which forces objects to be grasped from above. However, some objects can better be approached from a different angle due to their shape (e.g. bottles, mugs, cans). Additionally, many of these grasping methods use a single top-down view of the object scene, thus relying on a favorable camera position. Kasaei and Kasaei (2021) developed a multi-view deep learning approach that can predict grasp poses in highly cluttered environments. They use a partial point cloud to generate multiple depth views of the object scene. The best view is chosen by a view selection function and fed to a neural network that predicts the grasping quality, angle, and width on a pixel-wise basis. By considering multiple views of the object scene their method allows for multi-directional grasping.

The grasping methodology in this paper will be limited to 4-DoF, top-down, open-loop grasping, relying on multi-modal RGB-D input data obtained from a single top-down view.

3 Problem formulation

The grasping problem that needs to be solved can be defined as (i) predicting an antipodal grasp from an n-channel input image of the object scene (ii) converting the predicted grasp expressed in image space to robot space (iii) instructing the robot to execute the predicted grasp.

A grasp will be formally defined according to the representation proposed by Morrison et al. (2018). A grasp G_r in the robot's frame of reference is defined by equation 3.1:

$$G_r = (\mathbf{P}, \Theta_r, W_r, Q) \tag{3.1}$$

$\mathbf{P}$ denotes the (x, y, z) position of the end effector's tip, Θ_r denotes the rotation of the gripper around the z-axis, W_r denotes the required opening width of the gripper, and Q denotes the grasp quality score.

The n-channel input image $\mathbf{I}$ can be defined as the set of real numbers $\mathbb{R}$ with height h and width h, as formally defined by equation 3.2:

$$\mathbf{I} = \mathbb{R}^{n \times h \times w} \tag{3.2}$$

The predicted grasp will first be expressed in terms of the input image space. This grasp, denoted as G_i, is formally defined by equation 3.3:

$$G_i = (x, y, \Theta_i, W_i, Q) \tag{3.3}$$

The pair (x, y) specifies the center point of the grasp in image coordinates, Θ_i denotes the rotation of the grasp in the camera's frame of reference, W_i the width of the grasp in image coordinates, and Q denotes the same grasp quality score as referred to by equation 3.1.

The grasp quality score Q, the angular rotation Θ_i, and the grasping width W_i are generated for all pixels in the input image. The sets of these values

will be denoted as $\mathbf{Q}$, $\mathbf{\Theta_i}$, and $\mathbf{W_i}$ respectively. The grasp quality score is expressed as a value in the range $[0, 1]$. A score approaching 1 indicates higher confidence in the success of the predicted grasp. The angular rotation is expressed as a value in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The grasping width is expressed as a measure of uniform depth and takes a value in the range $[0, W_{max}]$, where W_{max} denotes the maximum opening width of the gripper expressed as the number of pixels.

A grasp G_i can then be inferred from the sets $\mathbf{Q}$, $\mathbf{\Theta_i}$, and $\mathbf{W_i}$. A predicted grasp expressed in image space will first have to be converted to robot space in order for the robot to execute it. This can be done using a cascaded transformation as described by equation 3.4:

$$G_r = T_{rc}(T_{ci}(G_i)) \tag{3.4}$$

T_{ci} refers to the transformation that converts the grasp G_i from image space into three-dimensional camera space using the intrinsic parameters of the camera. T_{rc} transforms the cameras space into robot space using the position of the camera and the robot's base position. The obtained grasp G_r can then be executed by the robot.

4 Method

The grasp prediction process consists of three steps. First, the n-channel input data is normalized, cropped and resized to dimensions $n \times 224 \times 224$. The input data can consist of an RGB image, a depth image, or both. Grasp prediction is therefore not limited to one type of input modality. If a depth image is provided as input it will be inpainted according to the method proposed by Xue, Zhang, and Cai (2017) to obtain a better depth representation.

Next, the processed input is fed to the GR-ConvNet. The network generates four images that predict the grasp quality score, the grasping width, and the grasping angle in the form $\cos 2\Theta$ and $\sin 2\Theta$ for each of the input pixels.

In the final step, the GR-ConvNet's output is used to infer a grasp. The two images for the grasping angle are combined to obtain a single grasping angle image. This is due to the rotation of a grasp being uniform around $\pm\frac{\pi}{2}$. After smoothing the images using a Gaussian function, the three images are combined to infer one or multiple grasps.

4.1 Model architecture

Figure 4.1 shows the proposed GR-ConvNet architecture. The n-channel input is first passed through three convolutional blocks. Each convolutional block consists of a two-dimensional convolutional layer with batch normalization and a Rectified Linear Unit (ReLu) activation function. The convolutional layer is used for feature extraction. The batch normalization deals with the internal covariate shift by standardizing the input data from every mini-batch. This allows for higher learning rates and in some cases acts as a regularizer that helps prevent overfitting (Ioffe and Szegedy, 2015).

The convolutional blocks are followed by five residual blocks. Each residual block consists of a two-dimensional convolutional layer with batch normalization and ReLU activation function, followed by another two-dimensional convolutional layer with batch normalization. Additionally, each residual block has a skipping connection that adds the incoming data to the output of the block. This architecture allows for the training of deeper networks, and consequently, more complex functions can be learned (He et al., 2016).

Next are three transposed convolutional blocks. Each of these blocks consists of a transposed two-dimensional convolutional layer with batch normalization and ReLU activation function. The convolutional blocks have reduced the original input to dimensions 56×56. This size can be difficult to interpret. The transposed convolutional layers allow for upsampling such that each of the output images has dimensions 224×224.

The network has a total of 1,900,900 parameters. Due to its relatively lightweight nature, the network could be used for closed-loop control at a rate of up to 50 Hz (Kumra et al., 2020).

4.2 Training methodology

The training data consists of a set $\mathbf{I} = \{I^1 \dots I^n\}$ that contains n input images of various object scenes, and a set $\mathbf{G_i} = \{g_1^1 \dots g_{m_1}^1 \dots g_1^n \dots g_{m_n}^n\}$ that contains a number of successful grasps in images space for each of the n input images. The network aims to learn the function $\gamma(\mathbf{I}) \rightarrow \mathbf{G_i}$ that

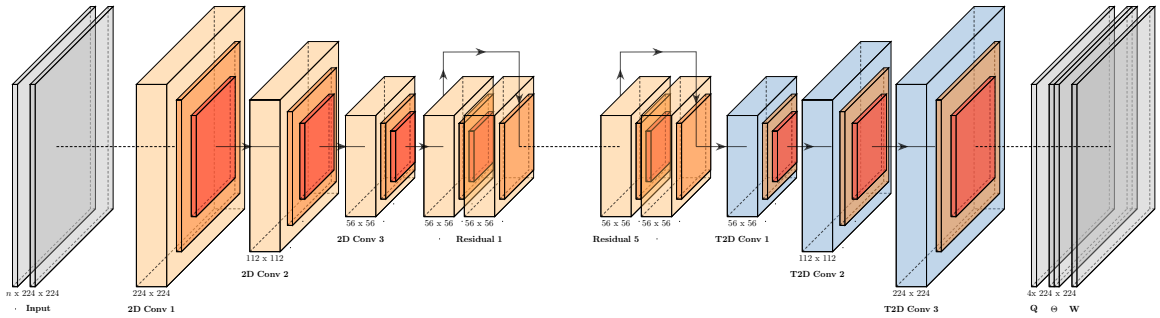

Figure 4.1: GR-ConvNet architecture - Yellow indicates the convolutional layers, orange the batch normalization, red the ReLU activation function, blue the transposed convolutional layers, and gray the input/output images.

maps the input images onto the successful grasps. This can be done by minimizing the negative log-likelihood that the function produces the successful grasp given the input image, as described by equation 4.1. The negative log function ensures that small input values result in large output values, and large input values result in small output values. The likelihood is averaged over the m grasps for each of the n input images.

$$-\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i}\log\gamma(g_j^i|I^i) \tag{4.1}$$

Kumra et al. (2020) report using different optimizer for training which include standard back-propagation, Adam (Kingma and Ba, 2017), and mini-batch Stochastic Gradient Descent (Nitanda, 2014). The learning rate was set to 10^{-3}, and a mini-batch size of 8 was used.

The loss function of choice is the smooth L1 loss, and is formally defined by equation 4.2.

$$\mathscr{L}(G_y,\hat{G_y}) = \sum_{k=1}^{4} z_k \tag{4.2}$$

G_y refers to the unprocessed output of the network, $\hat{G_y}$ to the ground truth output, and z to the loss value of the k-th output image, which is computed according to equation 4.3.

$$z_k = \begin{cases} 0.5(G_{y_k} - \hat{G_{y_k}})^2, & if\,|G_{y_k} - \hat{G_{y_k}}| < 1 \\ |G_{y_k} - \hat{G_{y_k}}| - 0.5 & otherwise \end{cases} \tag{4.3}$$

The loss function uses a squared term if the absolute error falls below 1 and an absolute term otherwise. This can help against exploding gradients as errors greater than 1 are not squared.

4.3 Grasp inference

The two angle output images can be combined to form a single grasping angle set $\mathbf{\Theta_i}$ using equation 4.4.

$$\mathbf{\Theta_i} = \frac{1}{2}\arctan\frac{\sin 2\Theta}{\cos 2\Theta} \tag{4.4}$$

A grasp G_i can then be inferred using the sets $\mathbf{Q}$, $\mathbf{\Theta_i}$, and $\mathbf{W_i}$. The best grasp, denoted as G_i^*, can be inferred by finding the argument for which the set $\mathbf{Q}$ has a maximum grasp quality score as formally described by equation 4.5.

$$(x,y) = \underset{a\in\mathbb{N}^2}{\operatorname{argmax}}\,\mathbf{Q}(a) \tag{4.5}$$

Argument a consists of two natural numbers which serve as the center point of grasp G_i^*. These two numbers can be used to obtain the angular rotation and the grasping width from the sets $\mathbf{\Theta_i}$, and $\mathbf{W_i}$ respectively to obtain a full grasp G_i^*.

A predicted grasp expressed in image space will first have to be converted into three-dimensional camera space. The depth of grasp G_i can be inferred using the depth input image. This value is determined by taking the minimum depth value of the surrounding neighbors at the image coordinates $(x,y) \in G_i$ that are within a radius of Δ. Both the center point of G_i and its depth value will have to be converted to a metric unit. The center point can

be converted using a conversion rate, and the depth value with the use of equation 4.6.

$$f(p_d) = \frac{m_f \times m_n}{m_f - (m_f - m_n) \times p_d} \tag{4.6}$$

p_d denotes the depth value, m_n the near plane distance, and m_f the far plane distance of the camera's projection matrix. Grasp G_i and its depth value can then be rotated into the camera space using transformation matrix 4.7. The Cartesian coordinates (x, y, z) refer to the camera's position, and θ refers to the necessary rotation around the z-axis.

$$M(x, y, z, \theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

To transform the camera space into the robot space the same transformation matrix can be used. However, this time the coordinates (x, y, z) refers to the robot's base position.

This process can be up-scaled to obtain multiple grasps from the sets $\mathbf{Q}$, $\mathbf{\Theta_i}$, and $\mathbf{W_i}$. Instead of only finding the argument for which $\mathbf{Q}$ has an absolute maximum, other arguments can be found where $\mathbf{Q}$ has local maxima. These grasps can then all be converted to robot space to obtain a set of grasps $\mathbf{G_r}$, as described by equation 4.8.

$$\mathbf{G_r} \leftarrow (\mathbf{Q}, \mathbf{\Theta_i}, \mathbf{W_i}) \in \mathbb{R}^{3 \times h \times w} \tag{4.8}$$

Figure 4.2 shows examples of the grasp quality, angle, and width output images. Additionally, the figure shows one or multiple inferred grasps, represented as rectangles, for object scenes containing a single or multiple objects.

5 Experimental results

To evaluate the proposed method by Kumra et al. (2020) the accuracy of the network is tested on a portion of the Cornell grasping dataset. Additionally, several rounds of three different grasping experiments are performed in a simulated robotic environment. Due to time and resource restrictions, a pre-trained GR-ConvNet model was used that comes from Kumra's open-source project[1]. A demonstrative video of the grasping experiments can be found online at `https://www.youtube.com/watch?v=fXpZMnZUZoA`

[1]Open source project from Kumra et al. (2020) available at: `https://github.com/skumra/robotic-grasping`

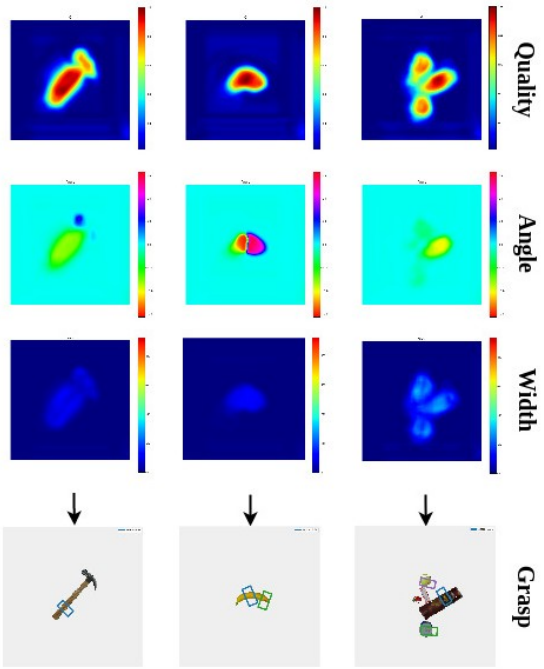

Figure 4.2: Network output - Figure shows the grasp quality, angle, and width output images with their corresponding inferred grasps represented as rectangles.

5.1 Network evaluation

The performance of the network is measured according to the rectangle metric as proposed by Jiang et al. (2011). According to this metric a grasp is considered to be valid if it satisfies the following two requirements:

i) The intersection over union (IoU) between the predicted grasp rectangle and the ground truth grasp rectangle is more than 25%.
ii) The orientation difference between the predicted grasp rectangle and the ground truth grasp rectangle is less than 30°.

Accuracy is then calculated by dividing the valid predictions by the total number of predictions made.

The network has been trained on the extended version of the publicly available Cornell grasping dataset. This dataset contains 1035 RGB-D images of 240 different objects. The images have been annotated with a total of 5110 valid grasps and 2909 invalid grasps. These ground truth annotations consist of several grasp rectangles per image. To increase the training data, Kumra et al. (2020) augmented the data by randomly cropping, rotating, and zooming in on the images. This yielded a total of 51,000 grasp examples. For training, only valid

grasps were considered.

Due to the Cornell grasping dataset being offline, a different version of the dataset was used[2]. This version contains 885 RGB-D images, yet still has 5110 valid annotated grasps. The RGB-D images are labeled with serial numbers and are sorted in increasing order. The evaluation set is generated by taking the last 10% of the RGB-D images and their corresponding annotations.

The network is then evaluated by generating a grasp rectangle for each image in the validation set. The predicted grasp rectangle is then compared to each of the ground truth annotations of the corresponding image. If at least one of the comparisons satisfies the two aforementioned requirements, the predicted grasp is considered valid. The overall accuracy is then calculated. This evaluation process yielded an accuracy of 97.7%.

5.2 Grasp evaluation

A robotic simulation in PyBullet (Coumans and Bai, 2016–2020) has been developed to test the grasping performance. PyBullet is a Python module for robotics simulation and machine learning. It wraps the C++ based Bullet physics engine, therewith combining Pythons ease of programming and the computational efficiency of C++. The code for the simulation is available online at `https://github.com/JeroenOudeVrielink/ur5-robotic-grasping`

The experimental setup consists of a UR5e robotic arm equipped with a two-fingered Robitq 2F-140 gripper placed on top of a desk. Various objects can be placed on the desk for the robot to grasp. The camera that obtains the RGB-D image is situated above the object scene. It's line of sight is perpendicular to the table in order to create a top-down view of the scene. Figure 5.1 displays the experimental setup.

The experiments, as proposed by Kasaei and Kasaei (2021), consist of three different grasping scenarios. These scenarios include objects in isolation, objects closely packed together, and objects in a pile. The goal of all three experiments is to grasp the target objects, lift them off the desk, and place them in the target tray. The robot knows the pose of the target tray in advance, whilst it has to detect the poses of the target objects.

The grasping performance is evaluated by measuring the success rate of objects grasped and placed inside the target tray. The success rate is defined as the number of successes divided by the number of attempts (see equation 5.1). A grasp is considered successful if both gripper fingers make contact with the object after it has been lifted off the table. An object has been successfully placed inside the target tray if the object's base position is within the target tray's bounding box. Additionally, for the packed and pile scenario, the percentage of objects that have been successfully cleared from the table is recorded.

$$success\ rate = \frac{number\ of\ successes}{number\ of\ attempts} \tag{5.1}$$

The set of target objects[3] contains 15 different objects with varying shape, size, and weight, and is a subset of the YCB object dataset (Calli, Walsman, Singh, Srinivasa, Abbeel, and Dollar, 2015). A full list of the objects can be found in appendix A.

The grasping procedure is identical for all three experiments. The robot is moved out of the camera's sight and an RGD-D image of the object scene is obtained. Three grasps are inferred and are sorted according to their quality score. If a grasp has a quality score that falls below the threshold of 0.6, it is discarded. The grasps are then executed in decreasing order by bringing the robot to the desired pose, and closing the gripper until both fingers squeeze the object with enough force. The objects are then lifted off the desk and brought to the know target location. If the robot has failed to place the object inside the target tray, all objects are restored to their pre-attempt poses such that the next grasp can be executed.

5.2.1 Isolated scenario

In the isolated object scenario, a single object is placed on the table with a random position and orientation (see Figure 5.1). The position is generated by selecting a random x- and y-coordinate that are both within ± 10 cm of the camera's (x, y) position. The orientation is generated by randomly rotating the object around its z-axis. A set of grasps is inferred and the best one, in terms of grasp quality, is finally executed. Only if the robot has failed to place the object in the target tray the next grasp is executed.

The experiment was run 100 times for each object in the target set. Figure 5.2 shows the grasping and target success rate for each individual object. An overall grasping of 94.8% was achieved, and a target success rate of 91.9% (see table 5.1).

5.2.2 Pack and pile scenarios

Figure 5.3 shows an example of a pack and pile scenario. For both scenarios, five objects are randomly selected from the target set. The pack scenario is created by first placing one object in the center of the desk. The

[2]The dataset was taken from Kaggle, a platform for open-source datasets, and is available at: `https://www.kaggle.com/oneoneliu/cornell-grasp`

[3]Object models are available at: `https://github.com/eleramp/pybullet-object-models`

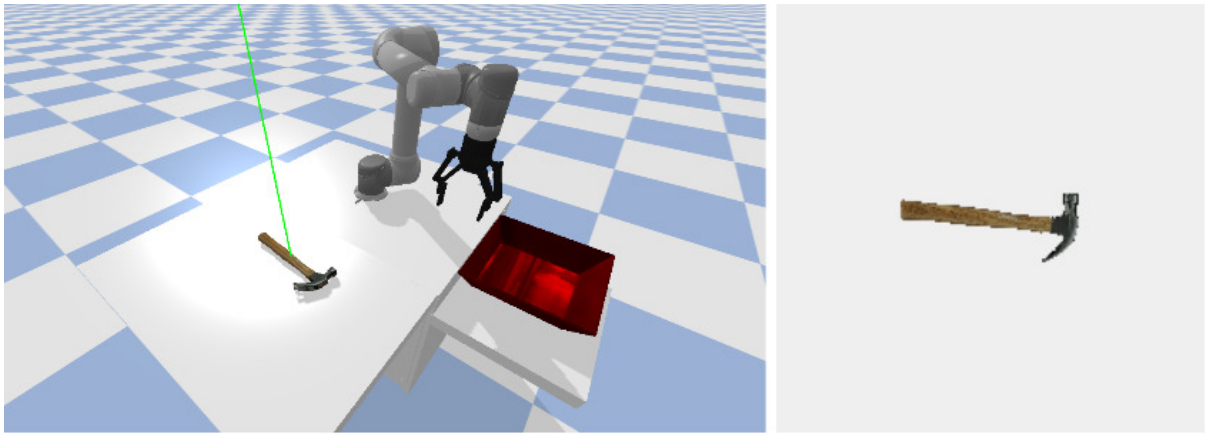

Figure 5.1: Experimental setup - The left image shows the UR5e robotic arm and a target object placed on the desk. The target tray is highlighted red. The green line denotes the camera's line of sight. The right image shows the corresponding view of the camera.

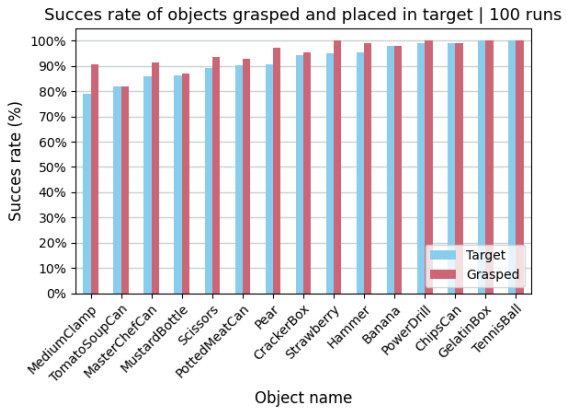

Figure 5.2: Experimental results isolated scenario: - The figure display the grasping and target success rate for each individual object. The experiment was conducted 100 times.

Figure 5.3: Pack and pile scenarios - Left image shows an example of a pack scenario. Right image shows an example of a pile scenario.

four remaining objects are then placed on the desk and moved toward the center objects until almost colliding with another object. All objects have been randomly rotated around their z-axis.

The pile scenario is created by randomly placing the target objects in a box. Once all objects are stable the box is removed. It should be noted that the objects *hammer* and *cracker box* have been excluded from the pile scenario. To create a pile of objects the box had to be small enough and the hammer did not fit. The cracker box does not fit in the gripper when toppled over.

After having created either one of the scenarios, a set of grasps is generated and executed. If an object has been successfully placed inside the tray, a new set of grasps is generated. The experiment continues until either all objects have been cleared from the desk, or three failed target attempts occur consecutively.

Both experiments have been conducted 100 times. An overview of the results can be found in table 5.1. For the pack scenario a grasping success rate of 77.4% was achieved, and a target success rate of 73.2%. Additionally, 447 out of a total of 500 objects have been successfully placed inside the target tray resulting in 89.4% of the objects cleared from the desk. In the pile scenario a grasping success rate of 59.2%, and a target success rate of 55.2% were obtained. In this scenario, 342 of the 500 objects were successfully placed inside the target tray resulting in 66.4% of the objects cleared from the desk.

Scenario	Grasping succ. rate (%)	Target succ. rate (%)	Cleared (%)
Isolated	94.8 (1513/1596)	91.9 (1466/1596)	-
Pack	77.4 (473/611)	73.2 (447/611)	89.4 (447/500)
Pile	59.2 (367/620)	55.2 (342/620)	68.4 (342/500)

Table 5.1: Experimental results - The table displays the grasping success rate, target success rate, and the percentage of objects cleared for each of the three scenarios.

6 Discussion

This section will provide further insight into the results obtained in the network and grasping evaluations. Additionally, the failure cases and the limitations of the used approach will be discussed.

6.1 Network

With an accuracy of 97.7% the GR-ConvNet shows good predictive performance on the validation set. However, there are two noteworthy points about the evaluation procedure. First, the version of the Cornell dataset used to evaluate the network contains fewer example images than the version used to train the network. Second, the network has been evaluated with a script that comes from Kumra's open source project, and it is assumed that this script generates a validation set that approximates the distribution of the train/test split that was used to train the network. For a more thorough network evaluation see therefore Kumra et al. (2020).

6.2 Grasping

With an overall grasping and target success rate of 94.8% and 91.9%, the grasping methodology has produced a reliable grasp in the isolated object scenario. The results do indicate performance differences between the different target objects. Some objects can be grasped flawlessly or with near perfection, whilst for other objects, the grasping and target success rate falls below 90%.

For the pack and pile scenarios, the methodology produced a less reliable grasp. The pack scenario shows almost a 20% decrease in both grasp and target success rate in comparison to the isolated scenario. Still, almost 90% of the objects can be cleared from the desk. Reliability decreases even further in the pile scenario, with a grasping success rate of a little under 60%, and a target success rate that approaches 50%. Additionally, a little under 70% of the objects can be cleared from the desk.

6.2.1 Failure analysis

Only a very small percentage of the failures in the isolated scenario can be attributed to invalid grasp predictions. Most of the failures are caused by predicted grasps that have their center points at the edge of an object, as can be seen in figure 6.1. This can cause round objects to be squeezed out of the gripper as it closes. Additionally, the transformation of the predicted grasp to three-dimensional space relies on a look-up table method without any form of camera calibration, causing a slight translation error. This amplifies the aforementioned problem and can cause objects that require a high level of precision to be missed or grasped improperly. Moreover, the inference method relies on local peaks in the grasp quality image, thus making it not always possible to infer more than one grasp for smaller objects.

Some of the failures in the packed and pile scenarios can be attributed to the previously described problems, but the bulk of the failures is caused by inaccurate predictions and the gripper colliding with nearby objects. As the objects move closer together it becomes harder to distinguish them from one another. This can cause a predicted grasp to have its center point in between two objects, or an imperfect orientation. The gripper colliding with nearby objects can prevent the robot from reaching the target pose, or being unable to close the gripper as the gripper fingers are blocked. It was observed that the pile scenario is more prone to these problems due to its highly cluttered nature. In addition, as objects are placed into the box to create the pile scenario they can take complex poses, adding extra complexity to the grasping problem. This explains the performance difference between the pack and pile scenario. Figure 6.1 illustrates some of the discussed issues.

6.2.2 Limitations and further work

It should be noted a number of changes have been made to the experimental setup in order to improve grasping performance. First, the objects *chips can*, *mustard bottle*, and *tomato soup can*, proved to be difficult to grasp when standing upright. It was therefore chosen to place the objects on their sides in all three grasping scenarios, as can be seen in Figure 6.2. Ideally, these objects would be approached from the side rather than from above. A grasping method that utilizes more than 4-DoF is therefore needed such as the method proposed by Kasaei and Kasaei (2021).

Figure 6.1: Predictions of failed attempts - Left image shows a predicted grasp placed at the edge of a round object. The Center image shows a packed scenario where all three predicted grasps caused a collision with surrounding objects. The right image shows a highly cluttered pile scenario, illustrating the difficulty of distinguishing objects from one another.

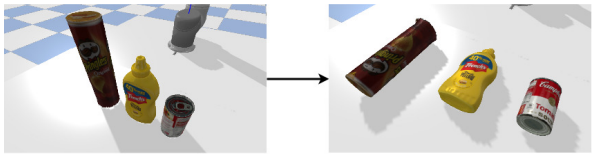

Figure 6.2: Challenging objects - The *chips can*, *mustard bottle*, and *tomato soup can* were placed on their sides.

Additionally, it was observed that objects are prone to falling out of the gripper when moving them to the target tray, even if grasped properly. To counter this problem, both gripper fingers and all target objects were modified to have increased friction.

Moreover, the original target set included a deformable object that behaved strangely when grasped due to the limited physics of the simulation. It was therefore chosen to exclude this object from the target set. The *strawberry* was also slightly deformable but was modified to be rigid. To deal with deformable objects more advanced grasping methods are required (Seita, Florence, Tompson, Coumans, Sindhwani, Goldberg, and Zeng, 2021).

To further improve grasping performance the grasp quality threshold and minimum distance between local peaks in the quality image can be tuned. Increasing the minimum distance can prevent the three predicted grasps from all be placed in a highly cluttered region. However, the minimum distance will be scenario-specific as too large values will prevent multiple grasp prediction for objects in isolation, which is not ideal for an all-purpose grasping system.

To deal with translation inaccuracy of the predicted grasp to robot space a camera calibration method could be implemented that corrects for lens distortion. Furthermore, some form of explicit collision checking method could be implemented, such as proposed by Li, Schomaker, and Kasaei (2020), to improve performance in the pack and pile scenario.

7 Conclusions

In this paper, the performance of the GR-ConvNet as a grasp inference method was evaluated by testing its predictive performance on a portion of the Cornell grasping dataset, and several rounds of the isolated, pack, and pile grasping experiments were conducted in a robotic simulation. The GR-ConvNet showed good predictive performance on the validation set, and a reliable grasp was observed for the isolated scenario. Grasping reliability decreased by a substantial portion in the pile scenario. Nevertheless, a large percentage of the objects could be cleared from the table. The pile scenario proved to be most difficult due to its highly cluttered nature, and grasping performance was observed to be the worst out of the three scenarios.

References

Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014. doi: 10.1109/TRO.2013.2289018.

Tom Brown et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics Automation Magazine*, 22(3):36–52, 2015. doi: 10.1109/MRA.2015.2448951.

Fu-Jen Chu, Ruinian Xu, and Patricio A. Vela. Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters*, 3(4):3355–3362, 2018. doi: 10.1109/LRA.2018.2852777.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2020.

Dan Ding, Yun Liu, and Shuguo Wang. Computing 3-D optimal form-closure grasps. volume 4, pages 3573 – 3578 vol.4, 02 2000. ISBN 0-7803-5886-4. doi: 10.1109/ROBOT.2000.845288.

S. Ekvall and D. Kragic. Interactive grasp learning based on human demonstration. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 3519–3524 Vol.4, 2004. doi: 10.1109/ROBOT.2004.1308798.

M. Fischer, P. van der Smagt, and G. Hirzinger. Learning techniques in a dataglove based telemanipulation system for the dlr hand. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 2, pages 1603–1608 vol.2, 1998. doi: 10.1109/ROBOT.1998.677377.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Yun Jiang et al. Efficient grasping from RGBD images: Learning using a new rectangle representation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3304–3311, 2011. doi: 10.1109/ICRA.2011.5980145.

Edward Johns, Stefan Leutenegger, and Andrew J. Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4461–4468, 2016. doi: 10.1109/IROS.2016.7759657.

Hamidreza Kasaei and Mohammadreza Kasaei. MVgrasp: Real-time multi-view 3D object grasping in highly cluttered environments. *arXiv preprint arXiv:2103.10997*, 2021.

Hamidreza Kasaei, Sha Luo, Remo Sasso, and Mohammadreza Kasaei. Simultaneous multi-view object recognition and grasping in open-ended domains. *arXiv preprint arXiv:2106.01866*, 2021.

S Hamidreza Kasaei, Nima Shafii, Luís Seabra Lopes, and Ana Maria Tomé. Interactive open-ended object, affordance and grasp learning for robotic manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3747–3753. IEEE, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

Sulabh Kumra and Christopher Kanan. Robotic grasp detection using deep convolutional neural networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 769–776, 2017. doi: 10.1109/IROS.2017.8202237.

Sulabh Kumra, Shirin Joshi, and Ferat Sahin. Antipodal robotic grasping using generative residual convolutional neural network. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9626–9633, 2020. doi: 10.1109/IROS45743.2020.9340777.

Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. doi: 10.1177/0278364914549607.

Yikun Li, Lambert Schomaker, and S. Hamidreza Kasaei. Learning to grasp 3D objects using deep residual U-nets. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 781–787, 2020. doi: 10.1109/RO-MAN47096.2020.9223541.

Douglas Morrison, Peter Corke, and Jürgen Leitner. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*, 2018.

A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2011.07.016.

Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

A.W. Senior, R. Evans, J. Jumper, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577:706–710, 2020. ISSN 7792. doi: 10.1038/s41586-019-1923-7.

Nima Shafii, S Hamidreza Kasaei, and Luís Seabra Lopes. Learning to grasp familiar objects using object view recognition and template matching. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2895–2900. IEEE, 2016.

Hongyang Xue, Shengming Zhang, and Deng Cai. Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *IEEE Transactions on Image Processing*, 26(9):4311–4320, 2017. doi: 10.1109/TIP.2017.2718183.

Xinchen Yan, Mohi Khansari, Jasmine Hsu, Yuanzheng Gong, Yunfei Bai, Sören Pirk, and Honglak Lee. Data-efficient learning for sim-to-real robotic grasping using deep point cloud prediction networks. *arXiv preprint arXiv:1906.08989*, 2019.

A Appendix

The set of target objects includes the objects: banana, chips can, cracker box, gelatin box, hammer, master chef can, clamp (medium size), mustard bottle, pear, potted meat can, power drill, scissors, strawberry, tennis ball, and tomato soup can.