university of groningen / faculty of science and engineering

Learning to Grasp 3D Objects Using Deep Convolutional Neural Networks

Bachelor's Project Thesis

Roberto A. Navarro San Martin, r.a.navarro.san.martin@student.rug.nl
Supervisors: Dr. Hamidreza Kasaei, hamidreza.kasaei@rug.nl

Abstract: In this paper the performance of the auto-encoder Generative Grasp Convolutional Neural Network (GGCNN) architecture proposed by Morrison et al. (2018) is evaluated in object classification and 3D object grasping tasks. The GGCNN is trained using the Cornell dataset. The output of the encoder part of the network is used as object representation for the object classification task. The full architecture is used for the grasping task to identify the most suitable grasp given the orthographic image of an object constructed using a Global Orthographic Object Descriptor (GOOD) from the point cloud. The ModelNet10 and Restaurant-Object datasets were used to study the impacts of the K value(for the Knn algorithm) and bin parameter of the network on classification accuracy; resulting in no significant difference in performance across 19 unique configurations for the ModelNet10 dataset and 18 different configurations for the Restaurant-Object data set. For the object grasping task, a simulation was developed in PyBullet, where a gripper executes the best grasp candidate and the success or failure of the grasp is recorded. A success rate of 83.3% was achieved for total objects grasped, while a total grasping success rate of 47% for all attempted grasps.

1 Introduction

Up until recent years robots have primarily been automated to work in assembly jobs with finite-state domains and little to no uncertainty. Advances in the fields of artificial intelligence (AI) and computer vision have enabled researchers to explore the application of robots outside of these environments in tasks ranging from autonomous driving to assistive robotics. Now more than ever, their introduction into the unstructured, dynamic environments of the real world creates a necessity for consistency and reliability.

A robot must be capable of interpreting the dynamic environments it occupies, whether the changes come from the environment, noise, errors, or inaccuracies in perception and control (Morrison et al.,2018). In addition, a robot must be able to interact and change the state of the world it resides in, which has led to the creation of an amplitude of techniques in Robotic Grasping (Morrison et al., 2018; Nguyen et al., 2016; Song et al., 2015; Zeng et al., 2017; Kokic et al. 2017; Kasaei et al., 2017; Papazov et al., 2012). Object classification, grasp detection and grasp planning are generally the three encapsulating steps required to perform the most basic of grasp tasks. CNNs have proven to be a popular tool that can out-perform traditional computer vision methods, to the extent that every step previously mentioned currently implements their use in state-of-the-art applications.

The Generative Grasp CNN proposed by Morrison et al. (2018) has enabled researchers to avoid the need to classify and segment objects for grasp tasks by directly working with the depth images of the objects. This is known as an object-independent grasp synthesis approach, producing outstanding results for known, familiar and unknown 3D objects in both isolated and clustered environments.

New developments in the field such as OrthographicNet, developed by Kasaei et al. (2019) uses orthographic projection for object recognition in open-ended domains. By generating 3 views of a 3D object (top, side, front view) and feeding each view to a CNN to obtain a view-wise feature for each projection, a global deep representation is constructed

by merging the features.

In this paper we propose an exploratory topic of feeding the 3D Orthographic projection views proposed on OrthographicNet to depth-image trained GGCNNs in order to extrapolate the best grasping point for a 3D object. The goal of this paper is to evaluate the performance of a GGCNN trained on 2D images, for 3D object classification and grasping tasks.

2 Related Work

2.1 Object Classification

Now more than ever, with advances in 3D sensing technologies, object classification has become a focal topic in computer vision. With the introduction of Neural Networks in computer vision, there has been an increased number of tools that have implemented Neural Networks, however these come with their own set of limitations.

One of these tools would be the Recursive Neural Networks and Convolutional Neural Network model proposed by Socher et al. (2012). Socher introduced a combinatorial model for deep learning and classifying 3D objects. The single CNN layer extracts low level features from RGB and depth images, creating one representation for each. The representations generated by the CNN are then pooled and passed to a set of RNN with randomized weights, which recursively map features into a lower dimensional space vectors. The vectors are then concatenated to generate a final feature vector that is given to a soft-max classifier. This method outperformed two layer CNNs, while producing state-of-the-art performance.

PointGrid, a deep auto-encoder CNN proposed by Le and Duan (2018) combines point cloud and volumetric grids to better learn high-level features. The general network architecture of PointGrid is defined by two convolutional layers followed by a max pooling layer, where after every two convolutional layers, the spatial dimension is halved while the number of convolutional filters is doubled. In order to perform high level reasoning they use two fully connected layers, each with a ReLU activation and drop out layer. Similarly to Socher et al. (2012) , the output of the network is then passed to a soft-max classifier to calculate the probability of each category the network was trained for.

Descriptors are another tool used for object classification, as indicated by Kasaei et al. (2016b), they can generally be either local or global. Local descriptors encode a small area of an object around a point, while global descriptors generally encode the entire object. Local descriptors are generally better at handling cluttered and occluded images, where comparison is done by analyzing local features. Global descriptors tend to be less computationally expensive, yet are not capable of comparing local features as effectively as local descriptors. Descriptors are a powerful tool in open-ended learning environments as they allow for unknown categories to be learned, unlike CNNs which can only classify based on predefined categories. Frome et al. (2004) helped lay the framework for recognizing objects by developing a 3D generalization of 2D shape object descriptors. Additionally they proposed the harmonic shape context descriptor, which at the time outperformed other state-of-the-art descriptors, such as Spin Images (Johnson and Hebert, 1999), for cluttered scenes.

In more recent years, Kasaei et al. (2016b) proposed GOOD, a global orthographic object descriptor for orthographic images based on 3D point cloud data. By creating a local reference frame using a Principal Component Analysis (See Section 3.1) and performing a vector concatenation, similarly to Socher et al. (2012), of the orthographic projections, the GOOD descriptor can be calculated. Section 3.2 describes the methodology for calculating GOOD.

2.2 Object Representation

Object representations can be generally classified based on the type of data used, which can either be point clouds, meshes, or volumetric grids (Le and Duan, 2018). As mentioned by Qi et al. (2016) neither meshes nor point clouds provide highly regular input data formats, which leads to researchers transforming such data to 3D voxel grids (Maturana and Scherer, 2015; Le and Duan, 2018) or a collection of images (Kasaei, 2019). However said transformation can increase the size of data unnecessarily and conceal natural in-variances.

Point clouds and volumetric grids have been the favored feature extraction format. Point clouds are flexible and simple, they avoid the combinatorial

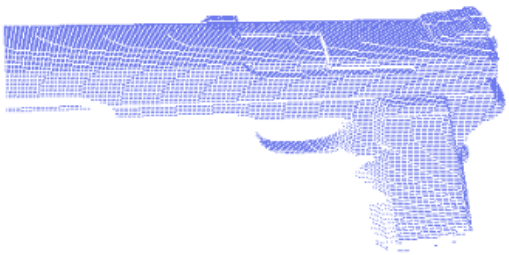

Figure 2.1: The partial point cloud representation of a pistol.

irregularities of meshes (Qi et al., 2016). However, point clouds still have limitations such as their unorganized structure and being permutation invariant (Qi, Su, Mo, and Guibas, 2016; Le and Duan, 2018). An example of a point cloud can be seen in Figure 2.1. As indicated by Le et al. (2018) volumetric grids provide highly regular data, unlike the previous representations, yet suffer from the computational and memory requirements due to the 3D grid resolution required to produce finer geometry details.

Tools such as OrthographicNet (Kasaei, 2019) generates three orthographic views of an object by using 3D point clouds, which contain the 3D coordinates of the point and the RGB data. In this paper, the object representation replicates the methodology of OrthographicNet, by generating orthographic projections. For an in-depth look at the orthographic projection methodology read Section 3.1.

2.3 Object Grasping

As mentioned by Bohg et al. (2013), identifying the best grasp from incredibly large sets of candidate grasps is a challenge that has generated a magnitude of methods. These methods can be divided into analytic methods and data-driven methods. Analytic methods generally assume that the exact location of object and contact points is known (Mahler et al., 2017). These methods provide the advantage of making grasp analysis more practical. However, it comes at the cost of increased inconsistencies and ambiguities (Bohg et al., 2013). In recent years data-driven grasp methods have become popular, which emphasize and rely on perceptual data such as effective object classification and pose estimation. This has occurred due to the increased use of CNNs in robotics and computer vision. Furthermore, it is widely accepted to group the data-driven methodologies based on the object information that is available (Kasaei et al. , 2017; Bohg et al., 2013; Papazov et al., 2012)whether methods attempt to generate grasp candidates for unknown objects, familiar objects or known objects.

In earlier years, Miller et al. (2003) investigated the effectiveness of grasping known objects by generating grasps based on the shape primitives (cylinders, boxes, cones and spheres). In this method, the grasp candidate domain is constrained by the primitives, where each primitive was given a strategy for grasping. Due to the nature of the approach, there was no consideration for grasp affordances, nor object recognition, leading to sub-optimal results in objects such as flasks. Additionally, this approach was not robust enough to effectively work in dynamic, complex environments in a timely manner.

Recent research has been focused on identifying grasp candidates by using affordance detection, yielding a large number of successful methods (Zeng et al., 2017; Song et al., 2015 Kokic et al., 2017; Kasaei et al., 2017; Ardon et al., 2019). Methodology proposed by Morrison et al. (2018) combined affordance detection with semantic object representation, by using Markov logic networks to learn the semantic relationships between object attributes, grasp affordances and locations. By using Gibbs sampling, they sample their grasping candidates based on the probability distribution of the candidate set and select the grasp with the highest probability given an affordance.

Kokic et al. (2017) addressed task-specific grasping. By combining grasp affordance detection using an auto-encoder CNN with contact constraints they are able to evaluate affordances in both full and partial view synthetic data. To describe semantic relationships between tasks and object classes they generate affordance labels, either cut, poke, pound, pour or support, to categorize object parts in order to identify if an object can be used to perform a task. In parallel they classify the object as a whole and identify the orientation. Once both object clas-

sification and affordance detection has finished they produce a binary output map that represents the graspable and non-graspable parts of the object. The binary output map, in addition to the classification and orientation of the object, are given to an optimization based grasp planner to generate a path perform the grasp.

Morrison et al. (2018) generated grasps on a real time manner using an object independent auto-encoder CNN. By using pixel-wise quality and pose prediction, they were able to avoid discrete sampling on grasp candidates and reduce computational time significantly, while reducing the size of the CNN and maintain performance similar to other CNN. Using their proposed method, they were capable of producing high-accuracy grasping results in static, dynamic environments for both isolated and cluttered objects; particularly a 81% accuracy in the dynamic cluttered environment. Additionally, due to the object independent nature of the GGCNN, it simply requires depth images to effectively grasp any object, forgoing the need to classify objects.

3 Methodology

As previously mentioned, the research goal of this paper is to evaluate the effectiveness of GGCNNs in object classification and grasping tasks that use orthographic projections. This section outlines the methods by which the research goal is studied, with a focus on orthographic projection, GOOD and the GGCNN. First a concise section regarding the method to generate the orthographic projections, and GOOD; both replicating the methodology of Kasaei et al. (2016a,2016b) and Kasaei (2019). Followed by an in-depth section focused on describing the architecture and other important features of the GGCNN and its use in the described tasks. Lastly, an explanation of the grasping pipeline used in this paper.

3.1 Orthographic Projection

Orthographic projection refers to the representation of a 3D object in 2D. Following Kasaei's (2016a, 2019) methodology, based on Principal Component Analysis (PCA), the point cloud of an object is represented as a set of points $P = p_i\{1, ..., n\}$, where each point p_i contains the x,y,z coordinates of the point. Kasaei (2019) creates a global object reference frame to represent the bounding box of the object, where the three principal axes of an object's point cloud are constructed by computing the center of the object using Equation 3.1.

$$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{3.1}$$

Once the center of the object has been calculated, a normalized co-variance matrix of the object is generated. This matrix, denoted as Σ, will be used in order to identify the eigen-values of the global object reference frame through eigen-value decomposition. Σ can be found by using the result from 3.1, where in order to find the variance, the geometric center is subtracted from the point and multiplied by its transposed value, as seen in Equation 3.2 .

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(p_i - c)(p_i - c)^T \tag{3.2}$$

Eigen-value decomposition is done using the equation ΣV= EV, where V and E are a set of three eigen-vectors and three eigen-values respectively. It is important to note that these eigen-values are ordered in decreasing order. Due to the fact the eigen-values have a corresponding eigen-vector, the eigen-vector $\vec{v_1}$ is always the largest vector of Σ, which indicates the direction with the largest variance. $\vec{v_2}$ is the second largest eigen-vector, and is orthogonal to $\vec{v_1}$, this allows us to define the X and Y axes using the $\vec{v_1}$ and $\vec{v_2}$ respectively. The Z axis is defined by the cross-product of $\vec{v_1}$ and $\vec{v_2}$.

Kasaei (2019) generated only the front, top and right-side primary orthographic views of the object and mirrored them for the rear, bottom and left-side views respectively. For each of the primary views a squared projection plane centered on the geometric center is generated; each view is perpendicular to one axis. The largest dimension of the axis aligned bounding box of an object determines the dimensions of the projections, which are always a square. The bounding box dimensions can be easily found by calculating the distance in coordinates between the minimum and maximum points of the largest side of the object. In order to identify the

sign of the axis, Kasaei (2019) calculates Pearson's correlation coefficients of the X and Y axes by projecting on the Z plane. To calculate the Pearson Correlation, each point in the projection plane can be denoted as $p = (\alpha, \beta) \in R^2$. The value range for this correlation is [-1,1], where a -1 is a strong negative relationship and a 1 is a strong positive relationship; it is this value that indicates the sign of the axis. Pearson's correlation can be seen in Equation 3.3:

$$r_{axis} = \frac{\sum \alpha_i \beta_i - n\overline{\alpha}\overline{\beta}}{(n-1)s_\alpha s_\beta} \tag{3.3}$$

Where r is the coefficient for the axis being projected on the Z plane, α and β indicate the perpendicular distance to the horizontal and vertical axis respectively. Furthermore, n is the number of points in the projection plane, $\overline{\alpha}$ and $\overline{\beta}$ are the mean values of their respective distances. Both s_α and s_β indicate the standard deviation of the points in the axis being projected on the Z plane, Equation 3.4 demonstrates how to calculate the s_α value; the s_β can be calculated analogously.

$$s_\alpha = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\alpha - \overline{\alpha})^2} \tag{3.4}$$

Once the correlation values for r_x and r_y are calculated, the sign value is found by their multiplication. If the product is a negative value, the three projections are mirrored, otherwise they remain the same.

3.2 Global Orthographic Object Descriptor

As described by Kasaei et al. (2016a, 2016b) GOOD is scale and pose invariant object descriptor. GOOD uses a unique and repeatable local reference frame (LRF), created by performing PCA (See Section 3.1). GOOD is computed by the concatenation of the three orthographic projections of the object in their respective orthogonal plane; as indicated in the previous sub-section. Each projection created by the orthographic projection can be described by a distribution matrix.

A distribution matrix, $\mathbf{M}$ can be calculated by counting the number of points that fall into a bin. A projection is divided into $n \times n$ bins; where n is the parameter that indicates the number of bins specified to compute GOOD. Using the same mathematical definitions for projections in Section 3.1, each projection is selected to generate a distribution matrix $\mathbf{M}_{n\times n}$, which is then normalized to achieve invariance. The matrix is converted into a vector $\mathbf{m}_{1\times n^2}$. Since three projections are generated for a single object, three vectors are produced. The vectors are finally concatenated.

The vector is concatenated in a specific order, the first vector must be the most informative and therefore it must have the highest entropy (See Section 3.2.1 for a detailed view at entropy). The two remaining vectors must be ordered based on the variance of the vectors in order to select which one appears as the second vector on the descriptor. Variance is defined using Equation 3.5 obtained from Kasaei et al. (2016b), this is an appropriate measure as it indicates the spread of the points in relation to each other; if many points are closer to each other it is likely the image is not very informative.

$$\sigma^2(m) = \sum_{i=1}^{n}(i - \mu_m)^2 m_i \tag{3.5}$$

As seen in the equation, i indicates the $\mathbf{m}_{1\times n^2}$ vector and m_i indicates the distribution at bin i. μ_m is the weighted average of the values of i, which correspond to the center of the object in the projection; this value can be calculated using Equation 3.6.

$$\mu_m = \sum_{i=1}^{n^2} i m_i \tag{3.6}$$

3.2.1 Calculating Entropy of an Image

The entropy of an image allows us to identify the image with most states, which serves as an indicator for the image with most information. The first step towards calculating entropy is converting the RGB image into a single-band grey-scale image. By having a grey-scale image, it becomes easier to find the marginal distribution of an image. The marginal distribution, in this context, is the probability distribution of the grey-scale levels in the image; the occurrences of each grey-scale level are counted and divided by the total number of levels. The probabilities that are equal to zero,

meaning no occurrences happen, are filtered out. The entropy is then calculated using Equation 3.7:

$$-\sum_{i=0}^{n-1} p_i log_b p_i \tag{3.7}$$

where p_i indicates the probability of a pixel p having a gray level i and b indicates the base used for the logarithmic function, in this case $b = 2$.

3.3 Generative Grasp Convolutional Neural Network

As previously mentioned the GGCNN created by Morrison et al. (2018) is a real-time object independent auto-encoder network that uses pixel-wise quality and pose prediction to generate a grasp map. Additionally, due to the network's single pass nature it has significantly less parameters than other classification methods, with 62,420 parameters. The network was trained using a subset of the Cornell dataset, created by Morrison et al. (2018), which contains 885 images of real objects that have a total of 8019 human labelled grasps. The data is augmented by generating images with random rotations, crops and zooms, producing a total of 8840 RGB-D images. The network is trained with 80% of the RGB-D images, using the RMSprop optimizer, and evaluated with the remaining 20% of the RGB-D images. The following subsection will dive deeper into how the GGCNN is used for the object classification task, and how it is used for the object grasping task, how it represents a grasp and its components, in addition to how the grasp map is represented. Immediately after, a subsection dedicated to the network architecture and its topology.

3.3.1 Object Classification

As previously stated, the GGCNN is an autoencoder network, meaning that it has one encoder and one decoder part. By using the output of the encoder part of the network, we obtain a feature vector which is used as an object representation. This representation is produced by feeding a training object instance into the network; in other words the network can be seen as a function that receives a depth image and produces a feature vector. For the object classification segment of this paper, two datasets will be used to benchmark the performance of the network for object recognition; ModelNet10 (Wu et al., 2015) and restaurant object (Kasaei et al., 2015). These datasets are divided into training and data, for a more in-depth look at the datasets, refer to sub-sections 4.1 and 4.2.

The feature vectors produced from the training split of each category in the dataset can then be stored in memory, along with their labels, to serve as "training" instances. Once all the feature vectors from the training partition have been produced, we can use them to classify the objects from the evaluation partition. A evaluation instance would be given to the network, which produces its corresponding feature vector, to then compare said vector with every other feature vector in memory (obtained from the training data). In order to carry out the comparison between two feature vectors the chi-squared distance function is used, as seen in Equation 3.8; where x and y are vectors, x_i and y_i are the elements i of their respective vector, and n is the length of the vector.

$$d(x, y) = \sum_{i=0}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)} \tag{3.8}$$

The k closest number of neighbours are compared, the label that appears the most frequently is assigned as the classification label of the evaluation instance, if there is an equal number of labels for different categories, the one with the closest distance is chosen. Each newly assigned label is then compared to the test instance's ground truth, and the accuracy of each set is calculated. Figure 3.1 visualizes how the algorithm works.

3.3.2 Grasp Representation

In order to represent a grasp we must be able to represent its components. The GGCNN, like other CNNs, defines its grasps perpendicular to a 2D axis. Morrison et al. (2018) define a grasp with the Equation 3.9. Here p indicates the pose of the gripper as the Cartesian coordinates (x,y) of the gripper's centre relative to an image, ϕ indicates the rotation of the gripper on the depth axis, w indicates the necessary width of the gripper and lastly the scalar quality heuristic q that represents the success probability of the grasp. Therefore a grasp point is selected based on the q parameter as it indicates the best grasp.

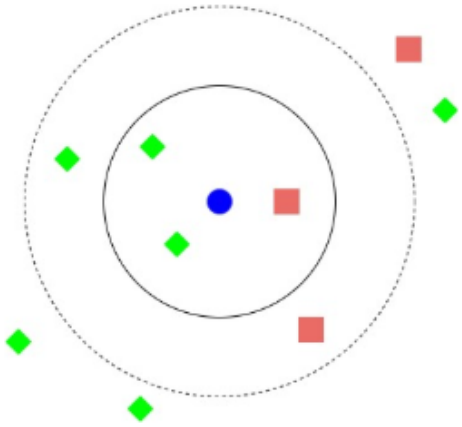

Figure 3.1: The blue filled circle indicates a feature vector in the coordinate space that is to be classified, while the green diamonds and red squares indicate classified objects from different categories. The blue circle is compared to the k closest instances, represented by the encapsulating circles.

$$g = (p, \phi, w, q) \tag{3.9}$$

Since the objective is to detect grasps on an orthographic depth image, each grasp is generated with coordinates relative to the projection i.e on the XoZ plane, where the X axis remains as the horizontal axis while the Z axis serves as the vertical axis and the Y axis as the depth. Morrison et al. (2018) redefine a grasp g in an image I as:

$$\widetilde{g} = (s, \widetilde{\phi}, \widetilde{w}, q) \tag{3.10}$$

For orthographic images the same definition holds, where s is a 2D point that represents the pixel coordinates; it is important to note that these coordinates can be represented in different axes, as previously described, as they are dependent on the axes of the selected orthographic projection. Additionally, $\widetilde{\phi}$ is the rotation of the gripper and $\widetilde{w}$ is the grasp width in terms of image coordinates.

A grasp map, which is the collection of grasps available in the orthographic image space, is then defined by Morrison et al. (2018) as:

$$\mathbf{G} = (\Phi, W, Q) \tag{3.11}$$

where Φ, W, Q are all elements of I and contain the values $\widetilde{\phi}$,$\widetilde{w}$,q for every pixel in an orthographic image I, in turn, they too are images since they contain a value for every pixel; therefore the output of the network is a set of orthographic images.

The grasp map is represented by three images of 300x300 pixels as seen in Equation 3.11. The two previously mentioned Q and W represent the quality of a grasp at each pixel and the gripper width necessary to grasp the object at each point respectively. Q is a scalar value that indicates the quality of the grasp point, where a 0 is of poor quality, while a 1 is of excellent quality. Meanwhile W is a value in pixels in the range of [0,150], indicating that a grasp width cannot be larger than half of the image I. The third image Φ represents the angle required to execute a grasp at each point in the image. Φ has a value range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$, due to the two point gripper used in their experiments; similarly in this paper a two finger gripper is used. Figure 3.2 demonstrates the output of the network for a sample object.

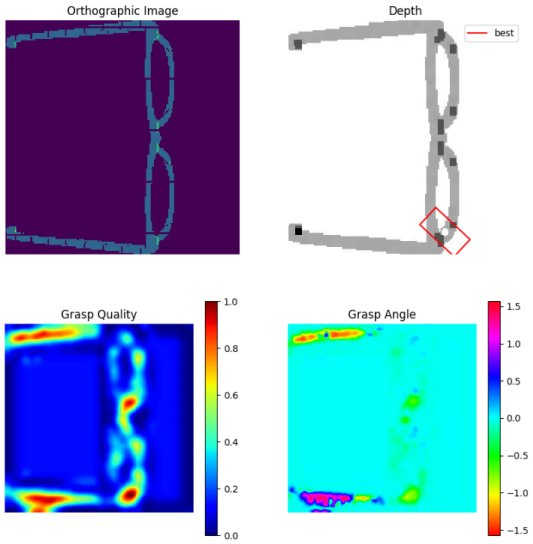

Figure 3.2: Output of the GGCNN for a pair of glasses for which the best grasp is calculated; Orthographic Image being the input of the network.

3.3.3 Network Architecture

Morrison, Corke, and Leitner (2018) indicate that initially the image is pre-processed in order to be fed into the network. The image is cropped to and scaled to a 300×300 pixel resolution in order to fit it to the input of the network, followed by in-painting invalid depth values. The image is fed to the GGCNN, which is a fully convolutional network with 6 convolution layers. The GGCNN's first layer is 9×9 layer with 32 filters with a stride of 3. The second layer is a 5×5 layer with 16 filters and a decreased stride of 2, followed by a third 3x3 layer with 8 filters with a stride of 2. It is at the forth layer where transposed convolution begins, with the identical dimensions, filters and stride sizes of the third layer. The fifth layer is a 3×3 transposed convolutional layer, with 16 filters of stride 2. The sixth and last transposed convolutional layer has the identical properties as the first layer.

As input it takes an Indepth image I of 300×300 pixels, and produces four 300×300 images: Q, Φ and W. Φ is a special case as it generates two images that represent the unit vector components of 2Φ, one for $sin(2\Phi)$ and $cos(2\Phi)$ each. The output of the network is a grasp map G, where Q is filtered using a Gaussian kernel, which increases the robustness of grasps by making G's local-maxima converge to regions with more quality grasps. The best grasp pose is then obtained by identifying the pixel with the highest value in Q.

3.4 Grasping pipeline

The execution of a grasp requires a multitude of steps, first the image from the simulation environment is taken from the camera, which has multiple buffers. The depth buffer of the camera is used to produce and store a depth image, which is then pre-processed by in-painting the depth values in the image in order to only have an image of the object. Next the image is passed to the GOOD service, which calculates the GOOD of an image, and returns the most informative projection in the form of a 300×300 in-painted depth image. The projection is then fed to the GGCNN network service in order to calculate an n number of grasps, ordered by success probability, the network then returns all the values for each grasp. The output of the network is then given to the gripper to execute.

The gripper always approaches the object in an orthogonal direction to the projection, meaning the gripper always makes a direct approach towards the desired grasp point. This indicates that, for example in a projection of the XoY plane the gripper approaches the plane directly perpendicular using the Z axis, in other words from the 'top'. Prior to the approach, the depth value of the point is found by reanalyzing the image and selecting the closest depth at the desired coordinate point. Once the depth has been identified the gripper approaches the object in a straight line, performs a stop, closes the two gripper fingers, and moves in an upward motion. If the object is not grasped, or if it falls from the grasp, it is considered a failed grasp attempt. Rightfully, if the gripper still has the object in its grasp after the upward motion is completed it is considered a successful grasp.

4 Experiments

There were a multitude of experiments ran, in particular to evaluate the GGCNN's performance and identify the best configurations in a object classification setting. In order to do so, two datasets were used to measure the accuracy of classification; ModelNet10 and the restaurant object dataset. The aforementioned datasets are significantly different in terms of size, which led to different experimental set-ups for each one. However the two parameters that were studied are the same, the number of neighbours used to classify the object (using the K nearest neighbours algorithm) and the number of bins used to divide the projections.

4.1 ModelNet10 Dataset

The ModelNet10 dataset contains 10 categories, with a total of 4899 object instances. The categories of the dataset are bathtub, bed, chair, desk, dresser, monitor, night stand, sofa, table, and toilet. Due to the relatively large size of the dataset, the dataset was split into mutually exclusive subcategories for training and testing. Approximately 80% of the dataset was used for training, while the remaining 20% was used for evaluation.

As previously mentioned, the goal of the experiments was to identify the best configurations for object classification. The K parameter was modi-

fied in order to inspect the effect on performance, starting with a value of 1 up to 11 using only odd values; this resulted in 6 configurations. It is important to note that the number of bins in these experiments remained constant; being 150 bins. In order to effectively measure the impact of every configuration each one was run three times with different training and testing data partitions, resulting in a total 18 evaluation runs. The average results of these experiments can be seen in Figure 5.1.

A similar experiment set-up was conducted for the bin parameter. The first experiment had configuration of 30 bins, each succeeding configuration after had an increment of 10 bins with 150 being the maximum bin value used for an evaluation run. Every configuration had a K value of 3. Similarly, every configuration was run 3 times with different training and testing data partitions, resulting in 39 evaluation runs. The average results of these experiments can be seen in Figure 5.2.

4.2 Restaurant Object Dataset

Similarly to the ModelNet-10 dataset, the restaurant object dataset has 10 categories; bottle, bowl, flask, fork, knife, mug, plate, spoon, teapot and vase. This dataset is significantly smaller with a total of 306 object instances, leading to a different, yet equally valid experimental set up. By performing a 10-fold cross-validation on the dataset where 80% of the dataset was used for training, while the remaining 20% was used for evaluation, with each cross validation having different training and evaluation data. The difference with this experimental set-up is the use of the 10-fold cross validation, meaning that rather than running 3 different instances of the same configuration on different data, the same configuration is run only once iterating over different mutually exclusive partitions of training and testing data. It is important to note the configuration parameters were not modified between folds in the cross validation.

Subsequently, a similar experiment set-up for the K parameters was performed. Each configuration was run once, each one with a bin parameter of 150 bins, with K being 1 in the first run, with an increment of 2 each consequent run up until 9. The reason for this dataset not performing an 11 nearest neighbour classification is due to limitations with one of the categories in the dataset; the fork category only contains 11 instances. The total number of configurations ran were 5, in which each run had 10 cross validations, resulting in 50 runs. The average results of these experiments can be seen in Figure 5.3.

The same number of bin configurations as the ModelNet-10 setup were evaluated; starting with 30 bins, each configuration having an increment of 10 bins, until a maximum of 150 bins. In total there were 13 configurations for this parameter, in which each run had 10 cross validations, resulting in 130 runs. The average results of these experiments can be seen in Figure 5.4.

4.3 Grasping Simulation

The simulation was developed using PyBullet, which is an API in python for the C++ based Bullet Physics Engine. It is a modified version of the simulation created by Li et al. (2020). The simulation runs in real time and grasps are performed on objects of 10 different categories using a two contact point Kuwait gripper. The 10 object categories used for measuring performance are as follows: axes, bottles, cups, glasses, guns, hammers, knives, mugs, lamps and plane-models.

Each category has 3 different object instances. Only one instance of an object exists on the simulation at any time, and each object is simply loaded in a location centered on the camera near the origin point of the world; as the simulation uses the world coordinates for moving and placing the gripper. For each object instance 3 grasps are calculated. Each of the calculated grasps can only be attempted three times, starting with the highest success probability grasp, followed by the second and third highest success probability grasps; if a grasp succeeds in less than three trials, the remaining grasps are not attempted. If the gripper does not succeed at the third trial, the grasp is considered a failure. Each grasp attempt is run completely independent of any other grasp trial, in order to ensure the experimental set-up is consistent for all runs. Additionally, it is important to note that a *bin* value of 150 was used for the GOOD descriptor. The value was selected to be half the resolution of the image, as any lower value would make grasping very inaccurate. The code for this can be found in the following repository: `https://github.com/RobertoNavrro/`

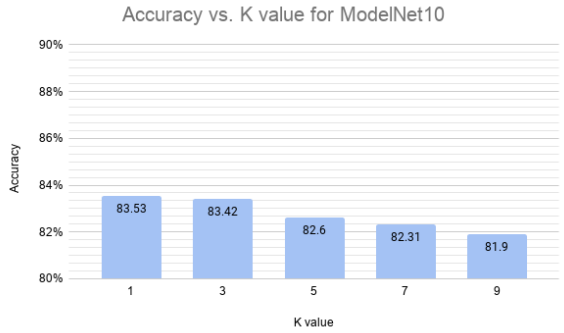

Figure 5.1: The average accuracy performance for every K value for the ModelNet10 dataset.

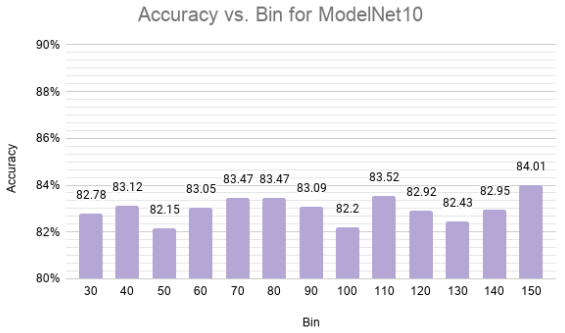

Figure 5.2: The plotted accuracy performance for every bin value for the ModelNet10 dataset.

`Orthographic_Object_Grasping_RUG.git`

5 Results

5.1 ModelNet10 Dataset

As previously indicated there were a total of 57 evaluations runs on the ModelNet10 dataset. In order to measure if the configuration had a significant impact on the GGCNN's performance, a series of statistical test are carried out, which allows us to identify if some configurations have a significantly different classification performance by analyzing the mean accuracy. To do so, the mean values of each configuration with the same independent variable were compared; all configurations with $bin = 150$ and $k = 1, 3, .., 11$ with each other, and all configurations with $k = 3$ and $bin = 30, 40, .., 150$ with each other.

K parameter: A visualization of the results can be seen in Figure 5.1. Using ANOVA we obtain a $F = 1.941$, furthermore it yielded a $p = 0.161$.

bin parameter: By performing the ANOVA test, we obtain a $F = 1.223$ and a $p = 0.32$. These results are further visualized in Figure 5.2.

5.2 Restaurant Object Dataset

The restaurant object dataset is significantly smaller, due to this a 10-cross-fold validation was ran for each configuration. The total number of validations for this dataset is therefore 180 runs, 130 for the *bin* parameter, and 50 for the K parameter.

Unlike the data for the ModelNet10, the results violate the normality assumption of ANOVA, leading us to use a non-parametric alternative; the Kruskal-Wallis test. Similarly to the set-up proposed in the ModelNet10 subsection, the aim is to compare the means of each configuration with the same independent variable as indicated previously.

K parameter: Using the Kruskal-Wallis test, we obtain a $p = 0.6842$. The results can be seen in Figure 5.3.

bin parameter: The results can be seen in Figure 5.4. Once again by running the Kruskal-Wallis test, we obtain a $p = 1$.

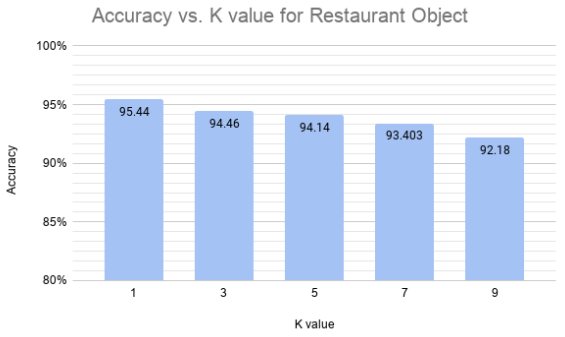

Figure 5.3: The plotted accuracy performance for every K value for the restaurant object dataset.

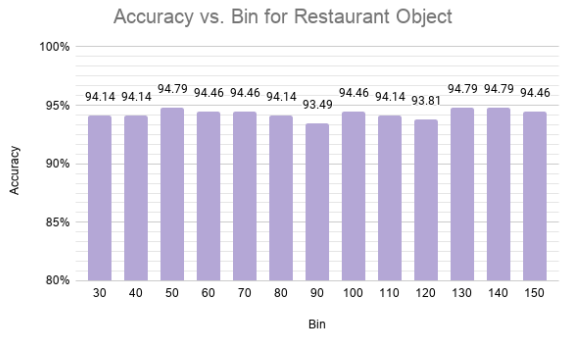

Figure 5.4: The average accuracy performance for every bin value for the restaurant object dataset.

5.3 Grasping

As previously mentioned, for each object in a category three grasps are calculated and executed. There will be two measures of performance that will be carried out in order to study the performance of the GGCNN on orthographic images; total accuracy and object accuracy. Total accuracy refers to the total number of successful grasps from the total attempted grasps, while object accuracy refers to the number of successfully grasped objects from the total object set. A total of 90 grasps were attempted.

As seen in Figure 5.5 the overall success of object grasping percentage was relatively high with an 83.3%, while the total grasping percentage was significantly worse with a 47% success rate.

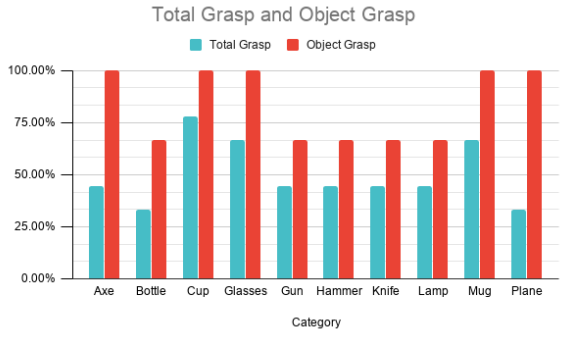

Figure 5.5: Results for the total and object grasping experiments performed per category.

6 Discussion

The following section delves deeper into the results obtained from the experiments. First, an analysis focused on the object classification task, which can be divided by datasets, followed by an examination of the results obtained in the grasping task.

6.1 Object Classification

An impact study of the bin and k parameters for the ModelNet10 and restaurant object datasets are examined to identify their importance on network performance.

6.1.1 ModelNet10 Dataset

The null hypothesis for the ANOVA test is $\mathbf{H} : \mu_1 = \mu_2 = ... = \mu_n$ In order to reject the null hypothesis, our test should produce a $p \leq 0.05$.

K parameter: As previously mentioned, the ANOVA test produced $F = 1.941$, indicating that the variability of performance within the same configurations is slightly high, this could be attributed to the quality of neighbours available at testing time. The p-value obtained was $p = 0.161$, indicating that the performance differences are not significantly different; the null hypothesis cannot be rejected.

bin parameter: As described in the results section, the ANOVA test produced an $F = 1.223$, which indicates that the variability of performance within equal *bin* parameter runs is relatively low. Furthermore, the obtained $p = 0.32$ clearly indicates that modifying the bin parameter does not yield any significant performance difference. This indicates that the alternative hypothesis is rejected, and the null hypothesis is true.

Figure 6.1 is the confusion matrix of the configuration with the best performance. This figure gives a clear example of the performance of the GGCNN on the ModelNet dataset, as it demonstrates that the 14.66% misclassification can primarily be attributed to the similarity between the categories table-desk, dresser-nightstand and sofa-bathtub.

6.1.2 Restaurant Object Dataset

The null hypothesis for the Kruskal Wallis test is the same as for ANOVA, $\mathbf{H} : \mu_1 = \mu_2 = ... = \mu_n$. In

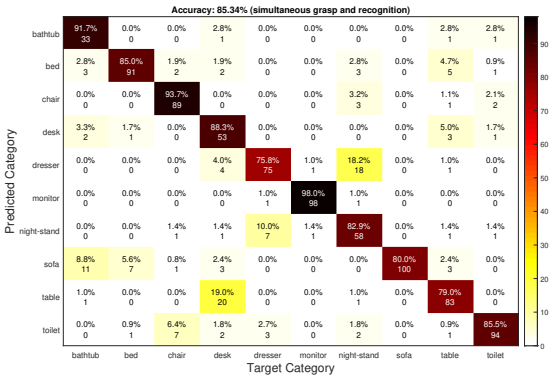

Figure 6.1: The confusion matrix of the best configuration run for the ModelNet10 dataset, where $K = 3$ and $bin = 150$.

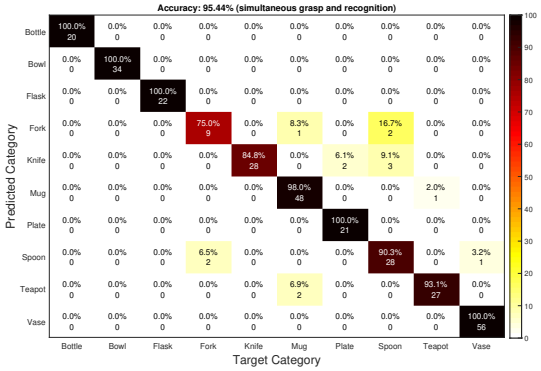

Figure 6.2: The confusion matrix of the best configuration run for the Restaurant Object dataset, where $K = 1$ and $bin = 150$

order to reject the null hypothesis, our test should produce a $p \leq 0.05$.

K parameter: The Kruskal Wallis test produced a $p = 0.6842$, which clearly indicates that modifying the K parameter has no impact on the performance of the network.

bin parameter: As previously described, the Kruskal Wallis test produced a $p = 1$, which indicates that modifying the *bin* parameter yields no significant difference in the performance of the network.

Once again, by inspecting the confusion matrix of the best configuration run for the Restaurant Object dataset in Figure 6.2 it can be identified that the 4.66% misclassification occurs primarily due to three categories. These are the fork, spoon and knife categories. These objects are quite similar to each other, which explains the erroneous classifications.

6.2 Object Grasping

The results can be explained primarily two factors, the orthographic projection method and some of the simulation's inaccuracies. Object categories knife, bottle, gun and hammer all had an instance that could not be successfully grasped, leading to a negatively skewed success rate, primarily due to their thin depth at the grasp location, not allowing the gripper to successfully pull the object. Additionally, in some occasions the grasp moved to a slightly inaccurate position causing the grasp to closely miss the desired location, this was due to the need to go from a resized orthographic image of the lower resolution to a real world coordinate system. Additionally, this conversion also required the grasping pipeline to approximate the closest depth value in that location, adding another potential inaccuracy.

7 Conclusions

In this section the main conclusions regarding the classification and grasping tasks are discussed, followed by a look into future potential research topics and the limitations of this study for each task.

7.1 Object Classification

The GGCNN provides an acceptable performance for 3D object classification tasks, by achieving an 85.34% accuracy on the ModelNet10 dataset and a 95.44% accuracy on the Restaurant Object dataset. The difference in performance between datasets can be explained by the data itself. The distance between objects within a category in the ModelNet10 is greater and the objects have much higher variability, while the objects for the Restaurant Object dataset are much similar within categories.

The obtained results indicate that the GGCNN could potentially be used in real-world environments where there might be notable memory restrictions, however if such restrictions do not exists there are other neural network tools that provide

better performance. Using the ModelNet10 dataset as a benchmark, other networks such as OrthographicNet, RotationNet, VRN-Ensemble and SPNet provide a higher accuracy by a margin of atleast 4%.

For future research it might be of interest to benchmark the performance of the network using the ModelNet40 dataset, and it would be fruitful to examine and study how the parameters for training the GGCNN can be refined to selectively improve classification performance. Furthermore, a study regarding the impact of pooling and distance functions in a similar classification task using the GGCNN could be insightful and hopefully generate better results.

7.1.1 Limitations

The primary limitation with the object classification tasks is the number of datasets used for bench-marking performance. In order to validate the GGCNN for object classification tasks in the real world, more datasets would be required. Additionally, due to the scope of this research, classification performance was not evaluated in a real world environment. It is crucial to test the GGCNN for object classification tasks in a real world setting to further validate the results obtained.

7.2 Object Grasping

As seen in the results for the grasping, the GGCNN achieves an object grasping success rate of 83%, however, the total grasping success rate indicates that the GGCNN might not be an ideal choice for orthographic images with a 47% total grasping success rate, a real world approach would be necessary to further evaluate the networks performance. Additionally due to the low object instance count, more trials with additional object instances and different object categories would be fruitful.

Furthermore, as a result of the limitations posed by the simulation environment and approximation methodology required, a study regarding the impact the of the bins in performance could indicate whether a higher bin count could provide better performance, albeit a bin count closer to the image resolution would defeat the purpose of separating the image in bins.

7.2.1 Limitations

The object grasping experimental set-up has a notable number of limitations, primarily due to the simulation environment. As previously mentioned, PyBullet is a python wrapper for the Bullet physics engine, and it is not fully complete as it lacks certain functionality from the Bullet Engine. The number of instances for each category could be bigger, this is due to the unavailability of working object models, specially since the models themselves simply provide the visual geometry of the object. The collision geometry of the object is obtained by performing Hierarchical Approximate Convex Decomposition, which produces a somewhat crude approximation of the object's hit-box, which inherently adds inaccuracies to the grasp execution and results. Furthermore, because of the down-sampling created by bins in the GOOD, some accuracy is again lost. Testing the performance of the GGCNN with orthographic images in a real world environment, where less inaccuracies can be present, could help produce more decisive results.

References

P. Ardón, È. Pairet, R. Petrick, S. Ramamoorthy, and K. Lohan. Learning grasp affordance reasoning through semantic relations. *IEEE Robotics and Automation Letters*, PP:1–1, 08 2019. doi: 10.1109/LRA.2019.2933815.

J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis - A survey. *CoRR*, abs/1309.2660, 2013. URL `http://arxiv.org/abs/1309.2660`.

A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *European conference on computer vision*, pages 224–237. Springer, 2004.

A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

H. Kasaei. OrthographicNet: A deep learning approach for 3D object recognition in open-ended domains. *arXiv preprint arXiv:1902.03057*, 2019.

H. Kasaei, M. Oliveira, G. H. Lim, L. S. Lopes, and A. M. Tomé. Interactive open-ended learning for 3D object recognition: An approach and experiments. *Journal of Intelligent & Robotic Systems*, 80(3-4): 537–553, 2015.

H. Kasaei, N. Shafii, L. S. Lopes, and A. M. Tomé. Object learning and grasping capabilities for robotic home assistants. pages 279–293, 11 2017. ISBN 978-3-319-68791-9. doi: 10.1007/978-3-319-68792-6_23.

S H. Kasaei, L. S. Lopes, A. M. Tomé, and M. Oliveira. An orthographic descriptor for 3D object learning and recognition. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4158–4163. IEEE, 2016a.

S H. Kasaei, A. M. Tomé, L. S. Lopes, and M. Oliveira. GOOD: a global orthographic object descriptor for 3D object recognition and manipulation. *Pattern Recognition Letters*, 83:312–320, 2016b.

M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic. Affordance detection for task-specific grasping using deep learning. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 91–98, 2017.

T. Le and Y. Duan. Pointgrid: A deep network for 3D shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018.

Y. Li, L. Schomaker, and S H. Kasaei. Learning to grasp 3D objects using deep residual u-nets. In *2020 IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020.

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. *CoRR*, abs/1703.09312, 2017. URL `http://arxiv.org/abs/1703.09312`.

D. Maturana and S. Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1824–1829. IEEE, 2003.

D. Morrison, P. Corke, and J. Leitner. Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Detecting object affordances with convolutional neural networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770, 2016.

C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka. Rigid 3D geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research*, 31(4):538–553, 2012.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL `http://arxiv.org/abs/1612.00593`.

R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3D object classification. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 656–664, Red Hook, NY, USA, 2012. Curran Associates Inc.

D. Song, C. H. Ek, K. Huebner, and D. Kragic. Task-based robot grasp planning using probabilistic inference. *IEEE Transactions on Robotics*, 31(3):546–561, 2015.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D shapenets: A deep representation for volumetric shapes. pages 1912–1920, 2015.

A. Zeng, S. Song, K. Yu, E. Donlon, F.s R. Hogan, M. Bauzá, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I.Taylor, W. Liu, T. A. Funkhouser, and A. Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *CoRR*, abs/1710.01330, 2017. URL `http://arxiv.org/abs/1710.01330`.