GraspCaps: Capsule Networks Are All You Need for Grasping Familiar Objects

Tomas van der Velde, Hamidreza Kasaei

Abstract—As robots become more accessible outside of industrial settings, the need for reliable object grasping and manipulation grows significantly. In such dynamic environments it is expected that the robot is capable of reliably grasping and manipulating novel objects in different situations. In this work we present GraspCaps: a novel architecture based on Capsule Networks for generating per-point grasp configurations for familiar objects. In our work, the activation vector of each capsule in the deepest capsule layer corresponds to one specific class of object. This way, the network is able to extract a rich feature vector of the objects present in the point cloud input, which is then used for generating per-point grasp vectors. This approach should allow the network to learn specific grasping strategies for each of the different object categories. Along with GraspCaps we present a method for generating a large object grasping dataset using simulated annealing. The obtained dataset is then used to train the GraspCaps network. We performed an extensive set of experiments to assess the performance of the proposed approach regarding familiar object recognition accuracy and grasp success rate on challenging real and simulated scenarios.

I. INTRODUCTION

As robots become more and more accessible outside of industrial settings (e.g., homes, hospitals, shopping malls), the need for reliable object grasping and manipulation grows significantly. In such dynamic environments, it is expected that the robot is capable of reliably grasping and manipulating novel objects in different situations (see Fig. 1).

A large body of recent efforts for object grasping has focused on solving 4DoF \((x, y, z, \text{yaw})\) object-agnostic grasping, where the gripper is forced to approach objects from above. The major drawbacks of these approaches are that (1) they do not take into account the semantic function or label of the object, and (2) they inevitably limit the interaction possibilities with the object. For instance, they are not able to distinguish objects from each other and are not able to grasp a horizontally placed plate. These limitations motivate the study of “learning to grasp familiar objects”, where the robot is able to recognize the label of the object and its gripper is free to approach objects from any arbitrary direction it can reach.

Towards this end, we formulate object grasping as a supervised learning problem to grasp familiar objects. The primary assumption is that new objects which are geometrically similar to known objects can be grasped in similar ways using object-aware grasping [1], [2]. Object-aware grasping allows the network to specifically generate grasps based on the features and geometry of the specific object it is attempting to grasp, as opposed to object agnostic grasping, which generates grasp vectors based on the input to the network without any deeper knowledge on the features of the object it is attempting to grasp. Since capsule networks are able to extract intrinsic parameters of the input data, we’ve designed a novel architecture called GraspCaps, that receives a point cloud of the object as an input and generates per-point grasp configurations and a single semantic category label as outputs. Our approach takes the activation of one capsule in the capsule network and processes that activation further to produce a semantic category label and point-wise grasp vectors.

We perform extensive sets of experiments in both simulated and real-robot settings to validate the performance of the proposed approach. The contributions of this thesis can be summarized as follows:

- We propose a novel architecture for object-aware grasping that receives a point cloud of an object as input and produces a semantic category label and point-wise grasp synthesis as outputs. To the best of our knowledge, this is the first grasping model based on the capsule network.
- We propose an algorithm for generating 6D grasp vectors from point clouds and create a synthetic grasp dataset consisting of 4,576 samples with corresponding object labels and target grasp vectors.
- We perform extensive sets of experiments in both simulated and real-robot settings to validate the performance of the proposed approach.

II. RELATED WORK

Current research into processing point clouds can be split up in two categories: (i) approaches that first transform the point cloud into a different data structure [3], [4], [5], (ii)
approaches that directly process the point cloud [6], [7], [8], [9], [10]. Our method falls into the second category.

Processing the point set directly has a number of advantages, since no overhead is added by transforming the point set, and there is no chance of any information loss in the conversion. However, point sets are by definition unordered, which makes extracting local structures and identifying similar regions non-trivial. PointNet [6] was one of the first architectures to effectively use point set data for training a neural network in an object recognition task. By design the PointNet architecture is mostly invariant to point order, which benefits point sets since extracting a natural order from these sets is non-trivial. However, this does limit the performance of PointNet as it cannot recognize local structures in point sets. In prior research the importance of order in data for the performance of neural networks has been illustrated [11], hence order should not be fully disregarded. PointNet++ [7] improves upon PointNet by recognizing local structures in the data. Our network architecture is based in part on the architecture used by [12], which makes the insight to split up the PointNet architecture into several distinct modules.

Later research showed successful results working with point sets by transforming the point set to be processed by a convolutional neural network. PointCNN [8] preprocesses the input data by applying a $\chi$-transform on the point set. DGCNN [10] and Point-GNN [9] employ layer architectures that transform the point set into a graph representation and apply convolution to the resulting graph edges.

Several approaches have been successful in processing point sets using a CNN by first transforming the point set into a more regular data structure, such as a 3D voxel grid [5], top-down view [3], [13], [14], or multi-view 2D images [4]. The resulting data structures can be processed with existing deep neural network architectures. These conversions come with significant limitations however, as there is a sizable loss in information when converting the point cloud to a different structure, whether that be in the form of losing natural point densities when converting to a voxel grid, or the loss of spatial relations between points when converting to a top-down image. Additionally, the generated voxel grids might be more voluminous than the original point set, as it is likely that many of the voxels remain empty [6]. Due to these considerations we decided to base the GraspCaps architecture in part on the architecture proposed by [10], as it is able to process the point cloud directly, and the graph representation obtained during execution is able to retain the spatial relation between points present in the input point cloud.

In the field of grasp generation, S$^4$G [15] extended the PointNet architecture to generate 6D grasps based on the input point set. Grasp pose detection (GPD) [16] was developed to generate and evaluate the fitness of grasps. It takes a point cloud as its input and generates a number of grasps which are then filtered on fitness. The network then classifies the grasp candidate as either successful, or unsuccessful. PointNetGPD [17] builds upon the idea of GPD and expands on it by employing the PointNet architecture to evaluate the fitness of grasps. GraspNet [18] uses a variable autoencoder to generate a set of grasps for an object using a point cloud. It evaluates the fitness of the generated grasps using an encoder-decoder network. It is able to generate grasps with a success rate comparable to PointNetGPD.

In contrast to GPD-based approaches, our method generates point-wise grasp configurations and is able to classify familiar objects.

### III. NETWORK ARCHITECTURE

As shown in Fig. 2, the GraspCaps architecture consists of three main modules: the feature extraction module, the capsule module, and the grasp synthesis module. The Feature Extraction Module takes a normalized $1024 \times 3$ dimensional point cloud as its input, and processes the point cloud using the feature extraction module from the Dynamic graph CNN (DGCNN) [10] network to generate a $1024 \times 1$ dimensional
feature vector. Specifically, it processes a point cloud by use of four EdgeConv layers. An EdgeConv layer first transforms the point cloud into \( N \) undirected local graphs of \( k \) nodes, where \( N \) is the number of points in the input point cloud, and \( k \) is an integer representing the number of closest neighbouring points that are used for constructing the local graph. The network then applies a shared multi-layer perceptron (MLP) on the edges of the graphs. This process is repeated for each of the four EdgeConv layers in the feature extraction module. The output of all five EdgeConv layers is concatenated and processed using a shared MLP and pooled to a \( 1024 \times 1 \) feature vector using an adaptive max-pool operation.

The **Capsule Module** of our network consists of a single fully connected layer, a primary capsule layer, and a secondary capsule layer. The fully connected layer consists of 256 neurons using sigmoidal activation that transform the feature vector for use in the primary capsule layer. The primary capsule layer consists of 16 capsules, each containing 4 neurons. The secondary capsule layer consists of 8 capsules each containing 30 neurons, with each capsule corresponding to one specific shape class of object. The primary and secondary capsule layers are connected through routing by agreement [19]. Capsules in deeper layers encode more meaningful features. Our network is trained such that each capsule in the deepest layer corresponds to one class of object. To determine the class of the object, the Euclidean norms of the activations in the secondary capsule layer are ranked. The capsule with the highest norm is then chosen as the winner and its activation is used for further processing in the next module.

The **Grasping Module** consists of four fully-connected heads. Their architecture is based on the architecture of the reconstruction network as described by [19], each containing three fully connected layers. The first two layers consist of 512 and 1024 neurons respectively, using leaky ReLU activation. The third layer size is dependent on the output shape of the network and uses linear activation. The reconstruction head has an output shape of \( 1024 \times 3 \) to generate point locations in 3D space. The grasp head has an output shape of \( 1024 \times 4 \) to generate quaternion rotation vectors for each point. The quality and width heads both have an output shape of \( 1024 \times 1 \), as they only need to output a scalar per point. The output of the grasping head is normalized to obtain a unit quaternion vector.

All parameters were chosen by performing extensive parameter sweeps.

### A. Loss functions

As the network performs several tasks at once, a custom loss function was written. The loss function can be broken down into three main components: Margin loss \( \ell_{\text{margin}} \) [19], Reconstruction loss \( \ell_{\text{recon}} \) [19], and Grasping loss \( \ell_{\text{grasp}} \).

Margin loss is used to train the capsule activation to be representative of the class it encapsulates:

\[
\ell_{\text{margin}}(v_i) = T_i \max(0, m^+ - \|v_i\|)^2 + \lambda(1 - T_i) \max(0, \|v_i\| - m^-)^2
\]  

where \( L \) is the loss, and \( T_i \) is the existence of the object the capsule corresponds to in the input data, which is set to 1 if the object is present, and 0 if the object is absent. The \( \lambda \) used is a constant scaling factor that is set to 0.5. It is used to scale down the shrinking of the activity vectors in the event that the output of the capsule is incorrect, which stops the initial learning from shrinking the activation vectors [19]. \( m^+ \) and \( m^- \) are constants set to 0.9 and 0.1 respectively. \( v_i \) is the activation vector of capsule \( i \). By calculating the Euclidean norm, \( \|v_i\| \), we obtain a scalar that can be treated as the probability that the corresponding class is present in the input.

Reconstruction loss is used as a regularizer that ensures that the capsule learns information relevant its corresponding object class. In our architecture it is defined as the mean squared error loss between the input point set and the reconstructed point set. It is scaled down by a factor \( \beta \) such that it doesn’t overtake the other losses and functions only as a regularizer.

The grasping loss is based on the loss function defined in [5], which is composed of separate loss functions for rotation, grasp quality, and the grasp width. The first alteration that was made from the loss function is the inclusion of \( T_i \), which behaves identical to the \( T_i \) described in the margin loss above. Including this factor in the grasping loss ensures that the grasping network does not train on capsule activations that do not correspond to the correct object. The second alteration from the loss function is the definition of the quality loss \( \ell_{\text{quality}}(q_i, \hat{q}_i) \), which we extend to allow any real-valued ground truth value \( q_i \in [0, 1] \) instead of the binary ground truth \( q_i \in \{0, 1\} \) used by [5]. Using this alteration the network is able to use \( q_i \) like a fitness value, and distinguish usable but imperfect grasps from the best-fitting grasps. Bad or colliding grasps are heavily penalized during data generation, so their values are relatively small, which reduces the risk of the network training on bad grasps. The grasping loss is defined as:

\[
\ell_{\text{grasp}}(g_i, \hat{g}_i) = T_i[\ell_{\text{quality}}(q_i, \hat{q}_i) + q_i(\ell_{\text{rotation}}(r_i, \hat{r}_i) + \alpha\ell_{\text{width}}(w_i, \hat{w}_i))]
\]

where both \( \ell_{\text{quality}} \) and \( \ell_{\text{width}} \) are defined as the mean squared error loss. The rotation loss is defined in Equation (3). \( \alpha \) is a constant set to 0.001. As the end-effector of the robotic arm we are using is a symmetrical two-fingered gripper, a grasp vector rotated by 180° around the grippers wrist results in effectively the same grasp [5]. Therefore, we define the rotational loss function to consider both grasps correct:

\[
\ell_{\text{rotation}}(r, \hat{r}) = \min(\ell_{\text{quat}}(r, \hat{r}), \ell_{\text{quat}}(r\pi, \hat{r}))
\]

where \( \ell_{\text{quat}}(r, \hat{r}) = 1 - |r \cdot \hat{r}| \) is defined to use the inner product to calculate the distance between the ground truth rotation \( r \) and the generated rotation \( \hat{r} \). The loss function that is used for training the network incorporates all of these loss functions as follows:
where \( \mathbf{t}_i \) is the output of the network, consisting of the capsule output \( \mathbf{v}_i \), the reconstruction of the input point set \( \hat{\mathbf{p}}_i \), and the generated grasp \( \hat{\mathbf{g}}_i \). \( \hat{\mathbf{t}}_i \) is the target vector, composed of the input point set \( \hat{\mathbf{p}}_i \), and the target grasp \( \hat{\mathbf{g}}_i \). \( \beta \) is a scaling constant set to 0.0005.

IV. GENERATING GRASP SYNTHESIS

Due to the niche our network exists in (simultaneous recognition and grasping on a point cloud), no existing benchmark datasets were found that met our requirements, so instead we created our own. In particular, to generate a large dataset with valid grasp configurations as targets, an algorithm was needed to create valid grasps based on point cloud data.

In this work, the gripper is simulated as a combination of three separate boxes: two for the fingers, and one for the base (see Fig. 3). Initially, we randomly place an object inside the workspace of the robot and capture the object’s point cloud. The point cloud of the object is then converted to a watertight mesh for more efficient collision checking. The original point cloud is retained to calculate the fitness of grasps. The algorithm then initializes the grasp in a random rotation, either with the gripper facing down towards the object or facing the object from the side. After initializing, the orientation and opening width of the gripper are iteratively updated to converge on a well-fitting grasp. A new state is accepted if the grasp has a higher fitness than the current state, or by random chance using simulated annealing [20].

In simulated annealing, the algorithm starts out with a high temperature. This temperature corresponds with a high chance of accepting new states, even if they have a lower fitness than the current state. The temperature of the system linearly decreases during execution and worse states are less and less likely to be accepted. In the final iterations simulated annealing behaves almost identical to stochastic gradient descent. The high temperature at the start of the process allows the algorithm to escape local maxima and converge to the global optimal solution, while the low temperature at the end of the process ensures that the algorithm converges. In terms of grasp generation it allows the system to escape usable but imperfect grasps and reach better grasps. The only state transition that is not able to occur even when accepting random state transitions is to go from a non-colliding state to a colliding state. We have also considered bounds on the rotation to ensure the algorithm does not generate grasps that are difficult for the robotic arm to reach.

The fitness of each state is determined by five factors. The first factor considers what percentage of points is located between the two fingers of the gripper. This should increase the likelihood that the algorithm converges on grasping a large part of the object. The second factor determines the minimum distance between a finger and the points between the fingers. It does this for each finger. The third factor looks at how closely the normals of the fingers of the gripper overlap with the normals of the object. If the normals are completely opposed to each other the gripper is perpendicular to the object, and the grasp should have a high chance of success. The fourth factor is defined as the distance between the two fingers. Since the grasp should be as tight as possible, a small penalty is added to the fitness for the distance between the two fingers. The last factor adds a base fitness based on the distance of the grasping point to the center of the object. The closer the point is to the center of the object, the higher its score. Examples of generated grasp synthesis for different objects are shown in Fig. 3.

V. EXPERIMENTAL RESULTS

A. Grasp dataset

As shown in Fig. 4 we used 90 simulated household objects to generate a large synthetic grasp dataset [21]. Furthermore, to address the familiar object recognition task, we grouped all objects into 8 distinct object categories based on shape similarities, e.g. milk cartons are grouped with juice cartons and other boxes. The obtained synthetic dataset is composed of 4,576 samples (572 per class). All samples are made by placing an object with random orientation and location in Gazebo [22] and taking a single sensor measurement using a simulated Microsoft Kinect camera to obtain a point cloud.

In order to train the grasp-generation section of the network efficiently, 120 grasp points are generated per object centered on a random points in the point cloud. Each
generated grasp consists of a rotation in quaternion format, a quality measurement, and the opening width of the gripper. By effectively generating two datasets, we are able to first train the classification section of the network, after which we train the grasping section. This resulted in better performance than training the network on a single one-sample-one-grasp dataset in an end-to-end manner.

B. Ablation Studies

Through a series of ablation experiments, we investigate the impact the capsule module has on familiar object recognition and grasping. As a baseline, we considered a network based on the GraspCaps architecture, but lacking the capsule layer present in GraspCaps. This allows the network to generate object agnostic grasps and allows us to observe whether capsule networks are a viable method of capturing the information needed to generate usable object aware grasps. The classification module is based on the architecture described in [10]. Both networks were first trained on the generated dataset of 8 classes for 300 epochs, after which they had their weights frozen in the feature extraction and classification modules and were trained on the 120 grasp per sample dataset for an additional 200 epochs. Networks trained on the real-life dataset were first pre-trained on the synthetic dataset for 300 epochs.

The network was trained with batches of 16 point clouds, and optimized using the ADAM optimizer [23] with a constant learning rate of 0.00001 \((10^{-5})\). Each of the objects’ point cloud is pre-processed by centering it around the point \(x = 0, y = 0\), and normalizing the data to the range \([-1,1]^3\). During training, the point cloud is augmented by adding random noise from a normal distribution in the range \([-0.005,0.005]\). This emulates the sensor noise often present in the real world and should let the network generalize better to unseen data.

The swift increase in grasp accuracy shows that both networks quickly converge to satisfactory performance. The most interesting part of these results is comparing the grasping accuracy on the synthetic dataset with the results obtained on the real-world dataset; in the synthetic dataset, both networks converge to approximately 94% accuracy, with GraspCaps outperforming the ablated network by a difference of less than 1%. When looking at the real-world dataset results however, the difference is much more pronounced. Again, GraspCaps achieves an accuracy of 94% while the ablated network converges to a significantly lower accuracy of 89%. This is a strong indication that the GraspCaps architecture has a higher generalization ability than the ablated network. The real dataset is more noisy than the synthetic dataset, so the results indicate that the GraspCaps architecture is more resistant to varying input data and noise than the ablated network, leading to superior performance in the real world.

C. Results in Simulation

To evaluate the performance of the proposed approach in a robotic grasping scenario, we integrated the GraspCaps architecture into our dual-arm robot setup in the Gazebo simulator. The robot will be required to identify the class of the object present, and generate a grasp configuration that can be used to successfully pick up the object. The robot will then pick up the object and attempt to place it into the basket. To execute the grasp, we select the arm that is closest to the object.

We increase the probability of correctly classifying the object in the input point set by pre-processing the input. We take the full point set of the object, which is often well over the required 1024 points, and take multiple permutations of 1024 points to give to the network. This decreases the chance of the network mis-classifying an object due to an unfortunate selection of points. The classification of the object is determined by a majority vote on the neural network output. In the event of a tie the algorithm is re-run until consensus is reached. The output of the network will be post-processed by smoothing the output of the fitness head of the network to identify regions with high fitness that are suitable for grasping. An example of the effect of the smoothing can be found in Fig. 5.

We found that, although both the GraspCaps network and the comparison network achieved an equal classification performance of 80%, the GraspCaps network outperformed the ablated comparison network in terms of grasping success with percentages of 71% for the GraspCaps network versus 53% for the ablated network.

D. Performance on novel objects

To test the generalizability of the network, we tested the network performance on a set of seven novel objects from the 3DGEMS dataset [24] shown in Fig. 6.

A number of the novel objects, such as the glass bottle and the tea box, can be easily sorted into the classes the network was trained on. Other objects however, such as the computer mouse and digital clock, have more abstract shapes and may be difficult to sort into the existing classes. For these objects specifically
it will be very interesting to see how the network will adapt and if it will be able to successfully grasp the objects.

Both the GraspCaps architecture and the ablated architecture will attempt to grasp each of the objects five times to get a clear indication of the generalization capabilities of both networks. The experimental setup and task will be equal to the setup in Gazebo described earlier this section. When comparing the results of the GraspCaps network shown in Table I with the results of the ablated network in Table II, it can be seen that the GraspCaps network outperforms the ablated network with an average of 71% grasping accuracy for the GraspCaps architecture compared to an average of 57% for the ablated network.

During the experiments on grasping the tissue and tea boxes, both networks generated grasp configurations that pierced the box with one of the gripper fingers in the majority of attempts, which caused the grasp to fail. The grasps that were generated on the boxes were in most cases correctly oriented, with the grippers parallel to the sides of the box. There is some sense to the generated grasps piercing the tissue box, as there is a significant opening in the center that could fit one of the fingers of the gripper, which the network could have noticed. An explanation for the tea box is harder to find. One possibility is that, since the tea box is rather large and has equally sized sides, both networks were unable to find grasp configurations that did not collide with the box. Adding large cubic objects to the dataset could potentially solve this problem. As can be seen in Tables I and II, all other objects had relatively high grasping success rates during evaluation of the network. This indicates that on average the network generalizes well for novel objects.

E. Results on robotic platform

In this round of experiments, we design a pick and place scenario to evaluate the performance of the proposed approach in real-world scenarios. We integrated the GraspCaps into our dual arm setup [25][1][26], and instructed the robot to perform the pick and place task. The GraspCaps network has been tested in a real world setting and has been found to perform well on objects that exist in its dataset, as well as on a number of novel objects. Three examples of grasping objects in various situations are shown in Fig. 7 Experimental results showed that the network was able to accurately pick up and place objects in the basket in both isolated scenario (Fig. 7 left and center) and pile of objects (Fig. 7 right), where a number of objects were placed on a table with random rotation and significant overlap, causing a number of objects to be partially or fully obscured from view. The network successfully picked up the objects in the pile with correct orientation of the gripper and was able to drop objects of multiple shapes into the basket without collision. A video demonstrating the performance of the GraspCaps network in both isolated object scenarios and a pile scenario has been attached to the paper as a supplementary material [1]. The video contains both successful and unsuccessful attempts, as well as novel objects that were not present in the training data.

VI. CONCLUSIONS

We have presented GraspCaps, a novel simultaneous recognition and grasping model based on capsule network. Along with GraspCaps we have presented a new dataset for simultaneous recognition and grasping on point cloud data and an algorithm for generating grasps using simulated annealing. GraspCaps has been found to perform well in both simulated and real-life settings on a variety of familiar and novel objects. For future research it would be interesting to extend the network with an additional head that generates an affordance mask, such that the network can be trained to grasp objects only at specific parts, for example, grasping the handles of a pan, or the handle of a knife.

1https://youtu.be/duuEDnk6HNw

TABLE I: Performance of the GraspCaps on novel objects.

| Object          | Classification | Frequency | Grasping success |
|-----------------|----------------|-----------|------------------|
| Bottle          | V Cylinder     | 5/5 (100%)| 4/5 (80%)        |
| V Cylinder      | 2/2 (40%)      |           |                  |
| Box             | 1/5 (20%)      |           |                  |
| Clock           | Hardware Box   | 2/5 (40%) | 5/5 (100%)       |
| V Object        | 2/2 (40%)      |           |                  |
| Mouse           | Box            | 5/5 (100%)| 5/5 (100%)       |
| Beer Mug        | V Object       | 5/5 (100%)| 4/5 (80%)        |
| V Cylinder      | 1/5 (20%)      |           |                  |
| Hardware        | 1/5 (20%)      |           |                  |
| Tissue box      | Box            | 5/5 (100%)| 1/5 (20%)        |
| Hardware Box    | 2/5 (40%)      |           |                  |
| Hammer          | H Cylinder     | 5/5 (100%)| 4/5 (80%)        |
| Average         | —              |           | 25/35 (71%)      |

TABLE II: Performance of ablated network on novel objects.

| Object          | Classification | Frequency | Grasping success |
|-----------------|----------------|-----------|------------------|
| Bottle          | V Cylinder     | 3/5 (60%) | 3/5 (60%)        |
| V Object        | 2/2 (40%)      |           |                  |
| Box             | 1/5 (20%)      |           |                  |
| Clock           | Hardware Box   | 3/5 (60%) | 5/5 (100%)       |
| V Object        | 1/5 (20%)      |           |                  |
| Mouse           | Box            | 2/5 (40%) | 5/5 (100%)       |
| V Cylinder      | 1/5 (20%)      |           |                  |
| Hardware        | 1/5 (20%)      |           |                  |
| Beer Mug        | V Cylinder     | 5/5 (100%)| 1/5 (20%)        |
| V Object        | 2/5 (40%)      |           |                  |
| Hardware        | 1/5 (20%)      |           |                  |
| Tissue box      | Box            | 5/5 (100%)| 1/5 (20%)        |
| Hardware Box    | 2/5 (40%)      |           |                  |
| Hammer          | H Cylinder     | 5/5 (100%)| 3/5 (60%)        |
| Average         | —              |           | 20/35 (57%)      |
REFERENCES

[1] S. H. Kasaei, N. Shafii, L. S. Lopes, and A. M. Tomé, “Interactive open-ended object, affordance and grasp learning for robotic manipulation,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 3747–3753.

[2] N. Shafii, S. H. Kasaei, and L. S. 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). IEEE, 2016, pp. 2895–2900.

[3] K. Santhakumar and H. Kasaei, “Lifelong 3d object recognition and grasp synthesis using dual memory recurrent self-organization networks,” Neural Networks, vol. 150, pp. 167–180, 2022.

[4] H. Kasaei and M. Kasaei, “Mvgrasp: Real-time multi-view 3d object grasping in highly cluttered environments,” arXiv preprint arXiv:2103.10997, 2021.

[5] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, “Volumetric grasping network: Real-time 6 dof grasp detection in clutter,” arXiv preprint arXiv:2101.01132, 2021.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” arXiv preprint arXiv:1706.02413, 2017.

[8] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, “Pointcnn: Convolution on χ-transformed points,” Advances in neural information processing systems, vol. 31, pp. 820–830, 2018.

[9] W. Shi and R. Rajkumar, “Point-gnn: Graph neural network for 3d object detection in a point cloud,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 1711–1719.

[10] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” Acm Transactions On Graphics (tog), vol. 38, no. 5, pp. 1–12, 2019.

[11] O. Vinyals, S. Bengio, and M. Kudlur, “Order matters: Sequence to sequence for sets,” arXiv preprint arXiv:1511.06391, 2015.

[12] A. Cheraghian and L. Petersson, “3d capsule: Extending the capsule architecture to classify 3d point clouds,” in 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2019, pp. 1194–1202.

[13] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[14] S. Kumra, S. Joshi, and F. Sahin, “Gr-convnet v2: A real-time multi-grasp detection network for robotic grasping,” Sensors, vol. 22, no. 16, p. 6208, 2022.

[15] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, “Volumetric grasping network: Real-time 6 dof grasp detection in clutter,” arXiv preprint arXiv:2101.01132, 2021.

[16] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp pose detection in point clouds,” The International Journal of Robotics Research, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[17] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, “Pointnetgpd: Detecting grasp configurations from point sets,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 3629–3635.

[18] A. Mousavian, C. Eppner, and D. Fox, “6-dof grasping net: Variational grasp generation for object manipulation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2901–2910.

[19] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” arXiv preprint arXiv:1710.09829, 2017.

[20] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, “Optimization by simulated annealing,” Science, vol. 220, no. 4598, pp. 671–680, 1983.

[21] H. Kasaei and S. Xiong, “Lifelong ensemble learning based on multiple representations for few-shot object recognition,” arXiv preprint arXiv:2205.01982, 2022.

[22] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2149–2154.

[23] A. Rasouli and J. K. Tsotsos, “The effect of color space selection on detectability and discriminability of colored objects,” arXiv preprint arXiv:1702.05421, 2017.

[24] H. Kasaei, M. Oliveira, G. H. Lim, L. S. Lopes, and A. M. Tomé, “Towards lifelong assistive robotics: A tight coupling between object perception and manipulation,” Neurocomputing, vol. 291, pp. 151–166, 2018.

[25] H. Kasaei, S. Luo, R. Sasso, and M. Kasaei, “Simultaneous multi-view object recognition and grasping in open-ended domains,” arXiv preprint arXiv:2106.01866, 2021.