Partial Caging: A Clearance-Based Definition and Deep Learning

Anastasiia Varava*[1], Michael C. Welle*[1], Jeffrey Mahler[2], Ken Goldberg[2],
Danica Kragic[1] and Florian T. Pokorny[1]

***Abstract*— Caging grasps limit the mobility of an object to a bounded component of configuration space. We introduce a notion of partial cage quality based on maximal clearance of an escaping path. As this is a computationally demanding task even in a two-dimensional scenario, we propose a deep learning approach. We design two convolutional neural networks and construct a pipeline for real-time partial cage quality estimation directly from 2D images of object models and planar caging tools. One neural network, CageMaskNN, is used to identify caging tool locations that can support partial cages, while a second network that we call CageClearanceNN is trained to predict the quality of those configurations. A dataset of 3811 images of objects and more than 19 million caging tool configurations is used to train and evaluate these networks on previously unseen objects and caging tool configurations. Furthermore, the networks are trained jointly on configurations for both 3 and 4 caging tool configurations whose shape varies along a 1-parameter family of increasing elongation. In experiments, we study how the networks' performance depends on the size of the training dataset, as well as how to efficiently deal with unevenly distributed training data. In further analysis, we show that the evaluation pipeline can approximately identify connected regions of successful caging tool placements and we evaluate the continuity of the cage quality score evaluation along caging tool trajectories. Experiments show that evaluation of a given configuration on a GeForce GTX 1080 GPU takes less than 6 ms.**

I. Introduction

A rigid object is *caged* if it cannot escape arbitrarily far from its initial position. This notion provides one of the classical paradigms for reasoning about robotic grasping besides form and force closure grasps [1], [2]. While form and force-closure are concepts that can be analyzed in terms of local geometry and forces, the analysis of caging configurations requires knowledge about a whole connected component of the free configuration space and is hence a challenging geometric problem that has been studied analytically in 2D and 3D. However, since global geometric properties of configuration space may also be estimated more robustly, caging may hold promise particularly as a noise-tolerant approach to grasping and manipulation. Caging grasps may furthermore also be viewed as an intermediate step towards acquiring a form closure grasp [2].

Provably-correct analytical methods have an obvious advantage, as they provide theoretical guarantees. However, they are often computationally expensive, while complete caging guarantees may furthermore not be necessary in many practical applications.

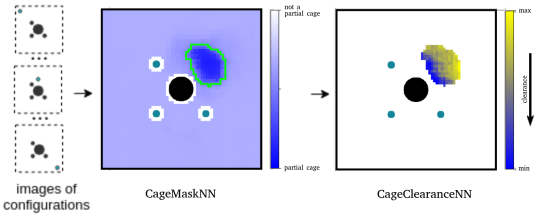

Fig. 1: Given an image of an object (depicted in black) and 3 or 4 caging tools (depicted in green), CageMaskNN determines whether a configuration belongs to the "partial cage" subset. If it does, CageClearanceNN, evaluate its quality according to the clearance measure learned by the network. On the figure, the blue region corresponds to successful placements of the fourth finger according to CageMaskNN, and their quality predicted by CageClearanceNN.

The notion of *partial caging* was first introduced by Makapunyo et al. [3], where the authors define a partial caging configuration as a non-caging formation of fingers that only allows rare escape motions. They introduced a quality measure for such configurations based on the complexity and length of paths constructed by a sampling-based motion planner, thus generalizing the binary notion of caging to a property parameterized by cage quality. We propose an alternative quality measure, which is based on the maximum clearance along any possible escaping path. This value is directly related to the maximum width of narrow passages separating the object from the rest of the free space. Intuitively, the quality of a partial cage depends on the width of a "gate" through which the object can escape. While we focus on the quality function definition, evaluation and learning in this work, we observe that this function changes continuously with respect to the caging tools' positions, and may hence in future be considered as a cost function for cage acquisition. We propose a Gate-Based Clearance Estimation Algorithm that evaluates partial caging configurations in a sampling-based manner.

One challenge with both our work and [3] is that a single configuration requires multiple runs of a motion planner and – in the case of RRT – potentially millions of tree expansion steps each, due to the non-deterministic nature of these algorithms. This increases the computation time of the evaluation process which can be critical for real-time applications, such as scenarios where cage quality needs to be estimated and optimized iterative to guide a caging tool from a partial towards a final cage. We significantly speed up the evaluation procedure for partial caging configurations by designing a deep learning-based pipeline that identifies

*These authors contributed equally.

[1]Anastasiia Varava, Michael Welle, Danica Kragic and Florian Pokorny are with the Division of Robotics, Perception and Learning, School of Electrical Engineering and Computer Science, KTH Royal Institute of technology varava,mwelle,dani,fpokorny@kth.se

[2]Jeffrey Mahler and Ken Goldberg are with the AUTOLab, Department of Electrical Engineering and Computer Science, University of California, Berkeley mahler, goldberg@berkeley.edu

partial caging configurations and approximates the partial caging evaluation function. For this purpose, we create a dataset of 3811 two-dimensional object shapes and 19055000 caging tool configurations and use it to train and evaluate our pipeline.

Apart from evaluating given partial caging configurations, we also use the proposed quality measure to choose potentially successful placements of one out of 3 or 4 caging tools, assuming the positions of the remaining tools are fixed. In Fig. 1, we represent the output as a heat map, where for every possible translational placement of a caging tool along a grid the resulting partial caging quality value is computed. Another application of the pipeline is the evaluation and scoring of caging configurations along a given reference trajectory.

The contributions of this paper can be summarized as follows: *i)* a partial caging quality measure based on the width of the gates formed by obstacles in the collision space of the object; *ii)* a deep neural network, CageMaskNN, that classifies caging tool positions as partial cages; *iii)* a deep neural network, CageClearanceNN, that approximates the partial caging quality measure; *iv)* a 2D partial caging dataset consisting of 3811 object shapes and 19055000 caging tool configurations, for 3 and 4 caging tools and where the shape of each caging tool varies along a 1-parameter family of elongation; each configuration is evaluated with respect to the proposed partial caging quality measure. The dataset will be made freely available on the first authors' website upon publication[3].

II. Related Work

One direction of caging research is devoted to point-wise caging, where a set of points (typically two or three) represents fingertips, and an object is usually represented as a polygon or a polyhedron. Rimon and Blake in their early work [4] proposed an algorithm to compute a set of configurations for a two-fingered hand to cage planar non-convex objects. Later, Pipattanasomporn and Sudsang [5] proposed an algorithm reporting all two-finger caging sets for a given concave polygon. Vahedi and van der Stappen in [6] described an algorithm that returns all caging placements of a third finger when a polygonal object and a placement of two other fingers are provided. Later, Rodriguez et al. [2] considered caging as a prerequisite for a form closure grasp by introducing a notion of a pregrasping cage. Starting from a pregrasping cage, a manipulator can move to a form closure grasp without breaking the cage, hence guaranteeing that the object cannot escape during this process.

One can derive sufficient caging conditions for caging tools of more complex shapes by considering more complex geometric and topological representations. For example, an approach towards caging 3D objects with 'holes' was proposed by some of the authors in [7], [8], [9]. Another shape feature was later proposed in [10], where we presented a method to cage objects with narrow parts. Makita et al. [11], [12] have proposed sufficient conditions for caging objects corresponding to certain geometric primitives.

Finally, research has studied the connectivity of the free space of the object by explicitly approximating it. For instance, Zhang et al. [13] use approximate cell decomposition to check whether pairs of configurations are disconnected in the free space. Another approach was proposed by Wan and Fukui [14], who studied cell-based approximations of the configuration space based on sampling. McCarthy et al. [15] proposed to randomly sample the configuration space and reconstruct its approximation as a simplicial complex. Mahler et al. [16], [17] extend this approach to verifying and generating *energy-bounded* cages – configurations where physical forces and obstacles complement each other in restricting the mobility of the object. These methods work with polygonal objects and caging tools of arbitrary shape, and therefore are applicable to a much broader set of scenarios. However, these methods are computationally expensive, as discretizing and approximating a three-dimensional configuration space is not an easy task.

To enable a robot to quickly evaluate the quality of a particular configuration and to decide how to place its fingers, we design, train and evaluate a neural network that approximates our caging evaluation function. This approach is inspired by recent success in using deep neural networks in grasping applications, where a robot policy to plan grasps is learned on images of target objects by training on large datasets of images, grasps, and success labels. Many experiments suggest that these methods can generalize to a wide variety of objects with no prior knowledge of the object's exact shape, pose, mass properties, or frictional properties [18], [19], [20]. Labels may be curated from human labelers [21], [22], [23], collected from attempts on a physical robot [24], [25], or generated from analysis of models based on physics and geometry [26], [27], [28], [29]. We explore the latter approach, developing a data-driven partial caging evaluation framework. Our pipeline takes images of an object and caging tools as input and outputs *(i)* whether a configuration is a partial cage and *(ii)* for each partial caging configuration, a real number corresponding to a predicted clearance is then used to rank the partial caging configuration.

Generative approaches to training dataset collection for grasping typically fall into one of three categories: methods based on probabilistic mechanical wrench space analysis [29], methods based on dynamic simulation [26], [28], and methods based on geometric heuristics [27]. Our work is related to methods based on grasp analysis, but we derive a partial caging evaluation function based on caging conditions rather than using mechanical wrench space analysis.

III. Partial Caging and Clearance

A. Partial Caging

In this section, we discuss the notion of partial caging. Let $\mathcal{C}$ be the configuration space of the object, $\mathcal{C}_{col} \subset \mathcal{C}$ be its subset containing configurations in collision, and let $\mathcal{C}_{free} = \mathcal{C} - \mathcal{C}_{col}$ be the free space of the object. Let us

[3] https://people.kth.se/~mwelle/

assume $\mathcal{C}_{col}$ is bounded. Recall the traditional definition of caging:

Definition 1: A configuration $c \in \mathcal{C}_{free}$ is a cage if it is located in a bounded connected component of $\mathcal{C}_{free}$.

In practical applications, it may be beneficial to identify not just cages, but also configurations which are in some sense 'close' to a cage, i.e., configurations from which it is difficult but not necessarily impossible to escape. Such partial caging can be formulated in a number of ways: for example, one could assume that an object is partially caged if its mobility is bounded by physical forces, or it is almost fully surrounded by collision space but still can escape through narrow openings.

We introduce the maximal clearance of an escaping path as a quality measure. Intuitively, we are interested in partial caging configurations where an object can move within a connected component, but can only escape from it through a narrow passage. The 'width' of this narrow passage then determines the quality of a configuration.

Let us now provide the necessary definitions. Since by our assumption the collision space of the object is bounded, there exists a ball $B_R \subset \mathcal{C}$ of a finite radius containing it. Let us define the escape region $X_{esc} \subset \mathcal{C}$ as the complement of this ball: $X_{esc} = \mathcal{C} - B_R$.

Definition 2: A collision-free path $p : [0,1] \rightarrow \mathcal{C}_{free}$ from a configuration c to X_{esc} is called an escaping path. The space of all possible escaping paths is denoted by $\mathcal{EP}(\mathcal{C}_{free}, c)$.

Let $cl : \mathcal{EP}(\mathcal{C}_{free}, c) \rightarrow \mathbb{R}_+$ be a cost function defined as the minimum distance from the object along the path p to the caging tools: $cl(p) = \min_{c \in p}(\text{dist}(o_c, \mathbf{g}))$ where o_c is the object placed in the configuration c and $\mathbf{g}$ denotes the caging tools. We define the caging evaluation function as follows:

$$Q_{cl}(c) = \begin{cases} \min_{p \in \mathcal{EP}(\mathcal{C}_{free}, c)} cl(p), \ \mathcal{EP}(\mathcal{C}_{free}, c) \neq \emptyset \\ 0, \mathcal{EP}(\mathcal{C}_{free}, c) = \emptyset. \end{cases} \tag{1}$$

B. The set $\mathcal{C}_{cage}$

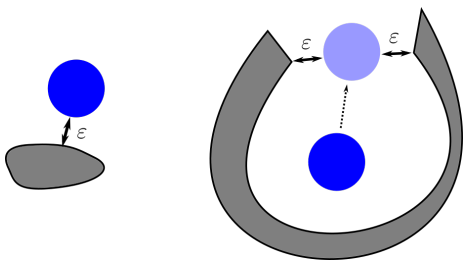

Fig. 2: On the left, an object (blue) can easily escape from the caging tool (grey); on the right, the object is partially surrounded by the caging tool and escaping is therefore harder. Both escaping paths will have the same clearance ε.

Observe that a low value of clearance measure on arbitrary configurations of $\mathcal{C}_{free}$ does not guarantee that a configuration is a sufficiently "good" partial cage. For example, consider only one convex caging tool located close to the object as in Fig. 2 (left). In this case, the object can easily escape. However, the clearance of this escaping path will be low, because the object is initial located very close to the caging tool. The same clearance value can be achieved in a much better partial caging configuration, see Fig. 2 (right). Here, the object is almost completely surrounded by a caging tool, and it can escape through a narrow gate. Clearly, the second situation is much preferable from the caging point of view. Therefore, we would like to be able to distinguish between these two scenarios.

Assume that caging tools are placed such that the object can escape. We increase the size of the caging tools by an offset, and eventually, for a sufficiently large offset, the object collides with the enlarged caging tools; let us assume that the size of the offset at this moment is $\varepsilon_{col} > 0$. We are interested in those configurations for which there exists an intermediate size of the offset $0 < \varepsilon_{closed} < \varepsilon_{col}$, such that the object is caged by the enlarged caging tools, but is not in collision. This is not always possible, as in certain situations the object may never become caged before colliding with enlarged caging tools. The left part of Fig. 3 illustrates this situation.

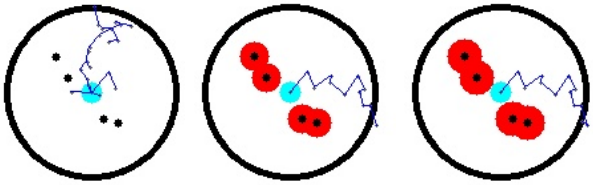

Fig. 3: The light blue disc depicts the object. The black discs depict caging tools, their offset is depicted in red. The RRT is depicted in purple. From left to right, three consecutive enlargements of the caging tools are depicted. The object can always escape until its initial configuration stops being collision-free.

Let us formally describe this situation. Let $\mathcal{C}^{\varepsilon}_{free}$ be the free space of the object induced by $\varepsilon-$offset of caging tools. As we increase the size of the offset, we get a nested family of spaces $\mathcal{C}^{\varepsilon_{col}}_{free} \subset ... \subset \mathcal{C}^{\varepsilon}_{free} \subset ... \subset \mathcal{C}^{0}_{free}$, where ε_{col} is the smallest size of the offset causing a collision between the object and the enlarged caging tools. There are two possible scenarios: in the first one, there is a value $0 < \varepsilon_{closed} < \varepsilon_{col}$ such that when the offset size reaches it the object is caged by the enlarged caging tools. This situation is favorable for our application, as in this case the object has some freedom to move within a partial cage, but cannot escape arbitrarily far as its mobility is limited by a narrow gate (see Fig. 4). We denote the set of all configurations falling into this category as the *caging subset* $\mathcal{C}_{cage}$. These configurations are promising partial cage candidates, and our primary interest is to identify these configurations. In the second scenario, for any ε between 0 and ε_{col}, the object is not caged in the respective free space $\mathcal{C}^{\varepsilon}_{free}$, as shown in Fig. 3.

We define the notion of *partial caging* as follows:

Definition 3: Any configuration $c \in \mathcal{C}_{cage}$ of the object is called a partial cage of clearance $Q_{cl}(c)$.

Note that the case where $\mathcal{EP}(\mathcal{C}_{cage}, c) = \emptyset$ corresponds to the case of a complete (i.e., classical) cage. Thus, partial caging is a generalization of complete caging.

Based on this theoretical framework, we propose a partial caging evaluation process that consists of two stages. First, we determine whether a given configuration belongs to the caging subset $\mathcal{C}_{cage}$. If it does, we further evaluate its clearance with respect to our clearance measure Q_{cl}, where, intuitively, configurations with smaller clearance are considered more preferable for grasping and manipulation.

C. Gate-Based Clearance Estimation Algorithm

Input: object O, caging tools G, ε_{max}
Output: clearance of an escaping path ε_{cl}
$\varepsilon_{min} \leftarrow 0$;
while *Can-Escape(O, G, ε_0)* **do**
  $\varepsilon_{cl} \leftarrow (\varepsilon_{min} + \varepsilon_{max})/2$;
  if *Can-Escape(O, G, ε_{cl})* **then**
    $\varepsilon_{min} \leftarrow \varepsilon_{cl}$
  end
  else
    $\varepsilon_{max} \leftarrow \varepsilon_{cl}$
  end
end
return ε_{min};

Algorithm 1: Gate-Based Clearance Estimation

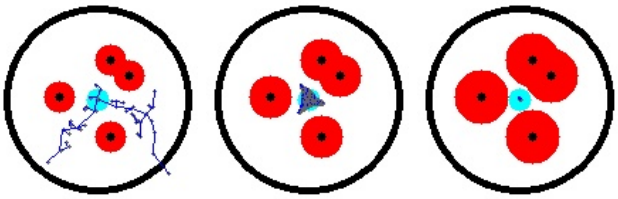

Fig. 4: From left to right: the object can escape only in the first case, and becomes completely caged when we enlarge the caging tools.

In this section, we propose one possible approach to estimate $Q_{cl}(c)$ – the Gate-Based Clearance Estimation Algorithm. Instead of finding a path with maximum clearance directly, we gradually inflate the caging tools by a distance offset until the object becomes completely caged. For this, we first approximate the object and the caging tools as union of discs, see Fig. 4. This makes enlarging the caging tools an easy task – we simply increase the radii of the discs in the caging tools' approximation by a given value. To estimate $Q_{cl}(c)$, we approximate objects and caging tools as unions of discs. The procedure is then described in Alg. 1.

We perform bisection search to find the offset value at which an object becomes completely caged. For this, we consider offset values between 0 and the diameter of the workspace. We run RRT at every iteration of the bisection search in order to check whether a given value of the offset makes the object caged. In the experiments, we choose a threshold of 4 million iterations [1] and assume that the object is fully caged if RRT does not produce an escaping path at this offset value. Note that this procedure, due to the approximation with RRT up to a maximal number of iterations, does not guarantee that an object is fully caged; however, since no rigorous bound on the number of iterations made by RRT is known, we choose a threshold that performs well in practice since errors due to this RRT-based approximation become insignificant for sufficiently large maximal numbers of RRT sampling iterations. In Alg. 1, Can-Escape(O, G, ε_{cl}) returns $True$ if the object can escape and is in a collision-free configuration.

[1] Our experimental evaluation for our test dataset suggested that if after 4 million iterations RRT had not found an escaping path, then the object was caged with overwhelming likelihood. We thus considered RRT with this setting to provide a sufficiently good approximation for training the neural network.

IV. Learning Q_{cl}

To estimate the quality of a configuration, we need to run RRT several times to determine the critical value of the offset at which the object becomes caged. The more difficult it is to find a path, the longer we should run each iteration. In real-time applications, a robot has to make decisions immediately and has to be able to evaluate caging configurations within milliseconds. Thus, the main obstacle on the way towards using the caging evaluation function defined above in real time is the computation time needed to evaluate a single partial caging configuration. To address this problem, we design and train two convolutional neural networks. The first acts as a binary classifier that identifies configurations that belong to $\mathcal{C}_{cage}$ defined in the previous section. The second one approximates the caging evaluation function Q_{cl} to further classify configurations. The network takes images of an object and caging tools as input, and returns an approximation of the partial caging evaluation function from Def. 3. The process is visualized in Fig. 1. Our goal is to estimate Q_{cl} given $O \subset \mathbb{R}^2$ – an object in a fixed position, and $G = \{g_1, g_2, .., g_n\}$ – a set of caging tools. We assume that caging tools are usually disconnected, while objects always have a single connected component. In our current implementation, we consider $n \in \{3, 4\}$, and multiple caging tool shapes.

A. Dataset Generation

We create a dataset of object models consisting of two-dimensional slices of objects' three-dimensional mesh representations created for the Dex-Net 2.0 framework [29]. We further approximate each model as a union of discs.

Fig. 5 shows four objects in three configurations. The dataset is composed of 3811 objects, 124435 partial caging configurations, and 18935565 configurations that do not belong to the caging subset $\mathcal{C}_{cage}$.

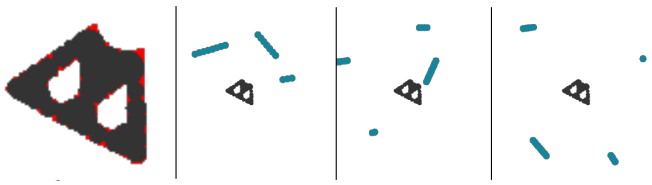

Fig. 5: First column: polygonal representations of objects (red) and in black their approximation by a union of discs of various sizes closely matching the polygonal shape; second and third column: configurations that do not belong to $\mathcal{C}_{cage}$; last column: a partial caging configuration.

B. Architecture of Convolutional Neural Networks

We propose a multi-resolution architecture that takes the input image as 64x64x2, 32x32x2, and 16x16x2 tensors. This architecture is inspired by inception blocks [30]. We utilize the same architecture for both CageMaskNN and CageClearanceNN. The network CageMaskNN determines whether a certain configuration belongs to $\mathcal{C}_{cage}$, while CageClearanceNN predicts the clearance Q_{cl} value for a given input configuration.

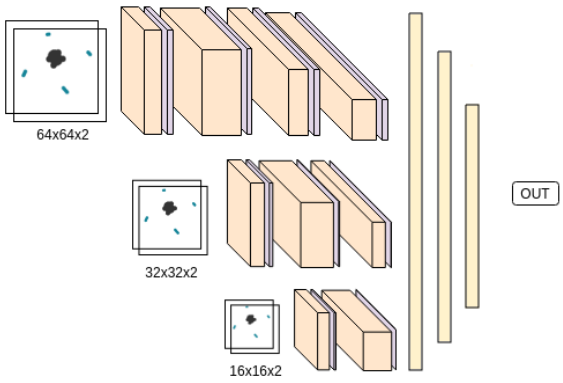

Fig. 6: As caging depends on global geometric properties of objects, a CNN architecture with multi-resolution input was designed to capture these features efficiently.

The architecture for the networks is shown in Fig. 6. Both networks take images of an object and caging tools on uniform background with position and orientation belonging to the same coordinate frame constituting a two-channel image (64x64x2) as input. CageMaskNN performs binary classification of configurations by returning 0 in case a configuration belongs to $\mathcal{C}_{cage}$, and 1 otherwise. CageClearanceNN uses clearance Q_{cl} values as labels and outputs a real value – the predicted clearance of a partial cage. The networks are trained using Tensorflow's [31] implementation of the Adam algorithm [32]. The loss is defined as the mean-squared-error (MSE) of the prediction and true label. The batch size was chosen to be 100 in order to compromise between learning speed and gradient decent accuracy.

V. Training and Evaluation of the Networks

A. CageMaskNN

We generate 4 datasets containing 5%, 10%, 15%, and 20% caging configurations in $\mathcal{C}_{cage}$ respectively. This is achieved by oversampling as well as by performing rotational augmentation with 90, 180 and 270 degrees of the existing caging configurations.

The evaluation is performed on a test set consisting of 50% caging examples from $\mathcal{C}_{cage}$. In Fig. 7, we show the ROC-curve and PR-curve. The F1-score and accuracy are visualized as well. All four versions of the network where trained with 3048 objects with 2000 configuration each, using a batch size of 100 and 250000 iterations. To avoid overfitting, a validation set of 381 objects is evaluated after every 100^{th} iteration. The final scoring is done on a test set consisting of 381 previously unseen objects. The mean squared error (MSE) on the unseen test set was 0.0758, 0.0634, 0.0973 and 0.072 for the 5%, 10%, 15% and 20% version respectively, indicating that CageMaskNN is able to generalize to novel objects and configurations from our test set.

Training set	AUC	AP
1 object	0.92	0.88
1 object, no validation	0.91	0.88
10 objects	0.91	0.88
10 objects, no validation	0.93	0.85
100 objects	0.97	0.92
1000 objects	1.00	1.00

TABLE I: AUC - Area under ROC curve and AP - average precision for different training set constitutions, evaluated on the test set. In all training sets 10 % of configurations belong to in C_{cage}. Varying this distribution between 5 % and 20 % did not lead to any difference in AUC and AP values.

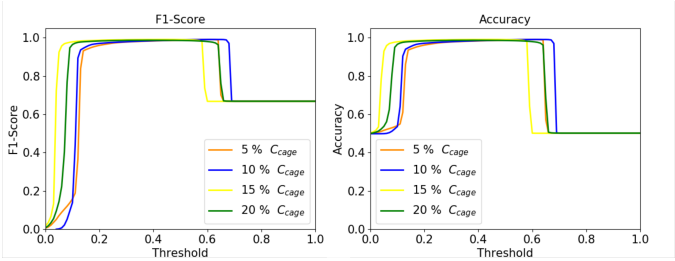

Fig. 7: F1-score and accuracy of the network depending on different thresholds

We observe that the network version with 10% cage relevant data performs slightly better than the other versions. Only the 5% setting however perform visibly worse.

Next, we investigate how the performance of the networks depends on the size of the training data. Fig. 8 demonstrates how the performance of the networks increases with the number of objects in the training dataset. As shown, the network performs relatively well even with a modest numbers of objects. One key factor is the validation set which lowers the generalisation error by choosing the best performance during the entire training run, thus reducing the risk of overfitting.

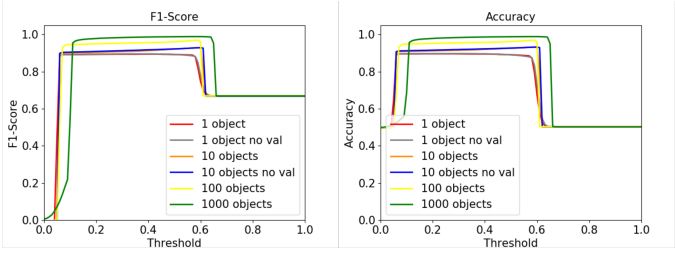

Fig. 8: F1-score and accuracy of the network trained with 1, 10, 100, and 1000 objects, respectively.

B. CageClearanceNN

The purpose of CageClearanceNN is to predict the value of the clearance measure Q_{cl} given a partial caging configuration. CageClearanceNN is trained on 3048 objects. The score is scaled with a factor of 0.1 as we found the network

to improve performance for smaller training input values. To explore how many objects are needed to reach a certain performance, several additional versions of the same network are trained. The result in Fig.9 shows a rapid performance increase in terms of MSE as we increase the number of training data objects to 1000, and a slight performance increase between 1000 and 3048 training objects. This indicates that further increases in performance may require a significant scaling-up of the training dataset.

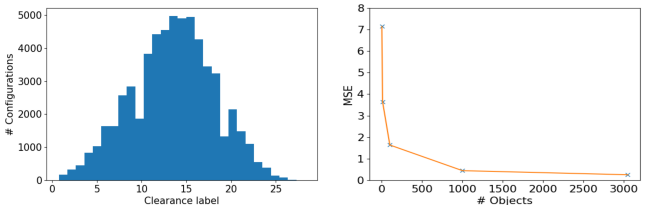

Fig. 9: left: test set label distribution, right: MSE performance for networks trained with 1, 10, 100, 1000 and 3048 objects.

We also investigated the MSE for specific clearance value intervals. Fig. 10 shows the MSE performance on the test set with respect to the clearance values scaled by 0.1. The network that was trained only on one object does not generalise over the entire clearance/label spectrum, but performance then increases with more objects. The outliers with large errors that are present in the network trained on 100 objects(bottom left) are significantly lower for the network trained on 1000 objects. On the right side, we can see the MSE for the final CageClearanceNN network. We observe that the errors are heavily concentrated along small values and relatively evenly distributed among clearance values.

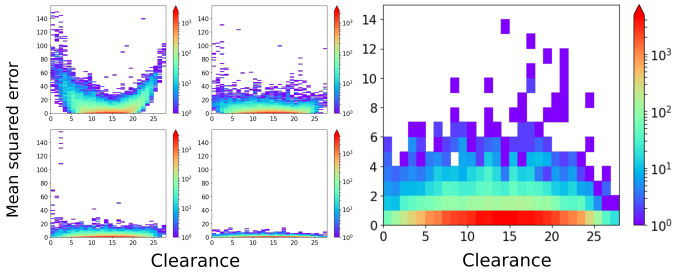

Fig. 10: MSE for each test case sorted for labels. Left: shows performance of 1, 10, 100, 1000 objects (top left, top right, bottom left, bottom right). Right: shows MSE of entire test set for the final CageClearanceNN. Note that the figure on the right is zoomed in as errors are significantly smaller (see the left y-axis of that figure)

VI. Caging Pipeline Evaluation

A. Caging tool placement

In this experiment, we consider the scenario where $n-1$ out of n caging tools are already placed in fixed locations, and our framework is used to evaluate a set of possible placements for the last tool to acquire a partial cage. We represent possible placements as cells of a two-dimensional grid and assume that the orientation of the caging tool is fixed. Fig. 11 illustrates this approach.

In the example a, we can see that placing the caging tool closer to the object results in better partial caging configurations. This result is consistent with our definition of the partial caging quality measure. We note furthermore, that CageMaskNN obtains an approximately correct region-mask of partial caging configurations for this novel object.

Example b demonstrates the same object with elongated caging tools. Observe that this results in a larger region for possible placement of the additional tool.

Example c depicts the same object but the fixed disc-shaped caging tool has been removed and we are considering three instead of four total caging tools. This decreases the number of possible successful placements for the additional caging tool. We can see that our framework determines the successful region correctly, but is more conservative than the ground truth.

In the example d, we consider an object with two large concavities and three caging tools. We observe that CageMaskNN identifies the region for C_{cage} correctly and preserves its connectivity. Similarly to the previous experiments, we can also observe that the most promising placements (in blue) are located closer to the object.

B. Random Disc Caging Tools Configurations at Fixed Distances

We now study the performance of our networks for four disc-shaped caging tools with a fixed distance from the object ε given by $a = 6$, $b = 9$ and $c = 12$ times the radius of the displayed disc-shaped caging tools. We analyze these three different values of ε and see how the performance changes. We randomly sample $n = 10000$ configurations for each of the different values of ε. The caging tools are positioned along the indicated distance offset lines and we do not allow configurations in self-collision (see Fig. 12).

Table II shows the F1 score as well as the accuracy for CageMaskNN and the MSE for CageQualityNN for two novel objects and the three considered distance offsets.

ε	object 1 (top)			object 2 (bottom)		
	$F1$	Acc.	E-GT	$F1$	Acc.	E-GT
a	0.64	0.93	6.82	0.36	0.9	8.96
b	0.72	0.93	21.31	0.4	0.89	19.11
c	0.78	0.94	34.33	0.49	0.87	24.13

TABLE II: Performance of CageMaskNN in terms of F1 score and accuracy, and CageClearanceNN in terms of Mean Squared Error relative to ground truth evaluation by RRT for each of the three distance thresholds and two previously unseen objects.

We observe that the F1 and accuracy performance increases as the caging tools are further away from the object. We believe this is because, as we increase distance, the impact of objects' local concavities decreases, while at close proximity the detailed local geometry matters more. On the other hand, MSE increase with distance, as the absolute magnitude of encountered clearance values also increases.

In Fig. 12 the highest scoring configurations among 10000 random samples per offset are plotted for each distance. We

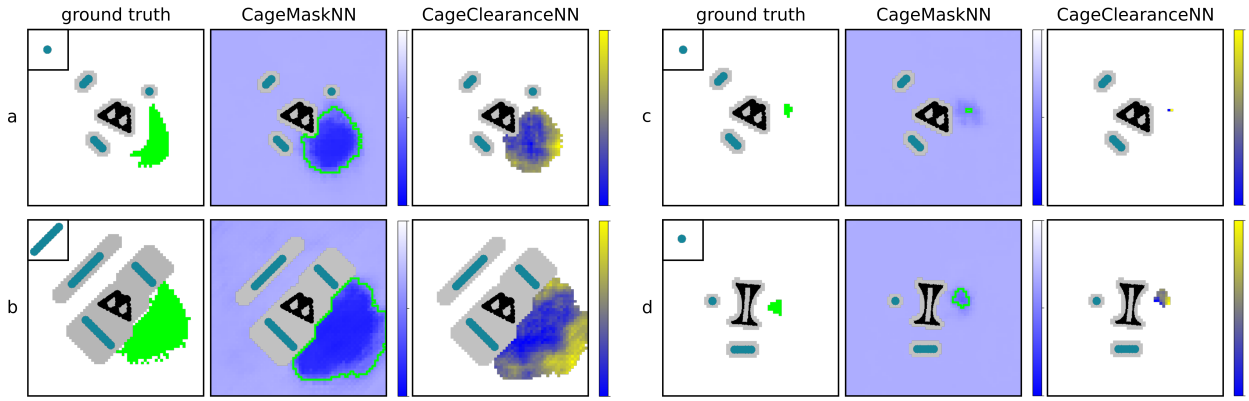

Fig. 11: Here, we depict the results of four different experiments. The green region indicates configuration where the additional caging tool completes the configuration in such a way that the resulting configuration is a partial cage. The small squares in the ground truth figures depict the caging tools that are being placed (their orientations are fixed). We plot the output for each configuration directly and visualize the result as a heatmap diagram (blue for partial caging configurations, white otherwise). The best placements according to CageClearanceNN are depicted in dark blue, and the worst ones in yellow. The results are normalized between 0 and 1. Grey area corresponds to the placements that would result in a collision.

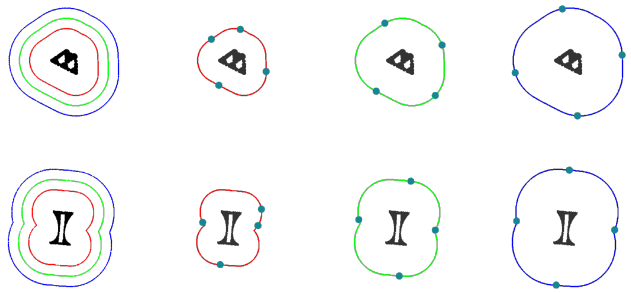

Fig. 12: Random caging tool configuration sampled at distance ε for $a = 6$ (red), $b = 9$ (green) and $c = 12$ (blue) times the radius of the displayed disc caging tools. The shown configurations achieved the lowest clearance score among $n = 10000$ randomly sampled configuration for each ε.

observe that, as we increase the distance, an equally spaced out configuration is selected as optimal choice for caging, confirming intuition. Furthermore, for the bottom object, the large concavity causes two of the caging tools to be placed alongside the concavity, while for the object in the first row, the local concavity at the top right appears to not have influenced the ranking sufficiently. Both objects have not been seen by the network during training, so that even the detection of the large concavity in the second row is in our view a success given the modest size of utilized training data and infinite dimensionality of the space of possible object shapes.

C. Evaluating Q_{cl} along a trajectory

We now consider a use case of Q_{cl} along a caging tool trajectory during manipulation enabled by the fact that the evaluation of a single caging configuration using CageMaskNN and CageClearanceNN takes less than 6ms on a GeForce GTX 1080 GPU.

The results for two simulated sample trajectories are depicted in Fig. 13 and a video is available in the supplementary material. In the first row, we consider a trajectory of two parallel caging tools, while in the trajectory displayed in the bottom row, we consider the movement of 4 caging tools: caging tool 1 moves from the top left diagonally downwards and then straight up, caging tool 2 enters from the bottom left and then exits towards top, caging tool 3 enters from the top right and then moves downwards, while caging tool 4 enters from the bottom right and then moves downwards.

The identification of partial caging configurations by CageMaskNN is rather stable as we move the caging tool along the reference trajectories, but occurs at a slight offset from the ground truth. In the second example, the clearance of the partial cage decreases continuously as the caging tools gets closer to the object. Predicted clearance values display little noise and low absolute error relative to the ground truth. Note that a value of -1 in the quality plots refers to configurations identified as not being in $\mathcal{C}_{cage}$ by CageMaskNN.

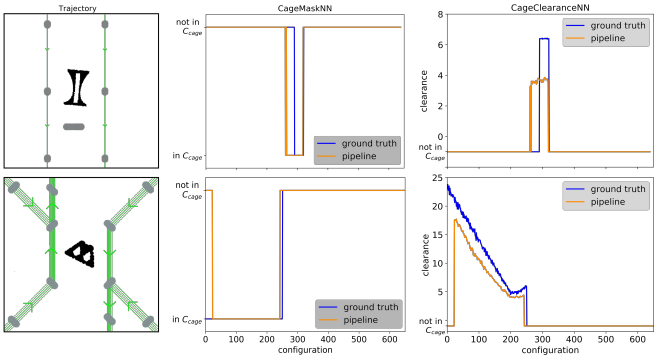

Fig. 13: Evaluation of the pipeline along two trajectories. The trajectory (left, green) is evaluated with CageMaskNN (middle) and CageClearanceNN (right), which evaluates Q_{cl} for those configurations where CageMaskNN returns 0. The predictions by the networks are displayed in orange while ground truth is shown in blue.

VII. Limitations and Future Work

The main challenge of our approach is the requirement to generate sufficiently dense training data over the joint space of all caging tools' configurations and caging tool and object

shapes. While our experimental evaluation in terms of MSE indicated that our network is able to achieve surprisingly low average errors on novel objects, one may in applications want to train the network with either a larger distribution of objects, or a distribution of objects as similar as possible to the objects that will be encountered in practice. In Fig.14, we illustrate how a lack of training data of sufficiently similar shapes can cause a break-down of CageMaskNN and CageClearanceNN, e.g. when only 1 or 10 objects are used for training. Similarly, even when trained on the full training dataset of 3048 objects, the subtle geometric details of partial caging region cannot be recovered for the novel test object, likely requiring further training data and refinement of the approach.

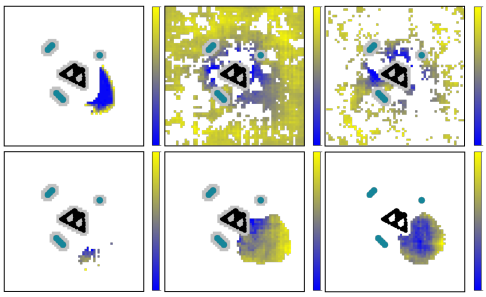

Fig. 14: Performance of CageMaskNN and CageClearanceNN given different numbers of training objects and evaluated on a single novel object. The top left displays the ground truth mask and clearance values for a fourth missing disc-shaped caging tool, top middle: only 1 object is used for training, top right:10 objects are used for training, bottom left: 100 objects, bottom middle: 1000 objects, bottom right: all 3048 objects are used for training. Note that the threshold had to be adjusted to 0.6 for the single object and 0.61 for the 10 object case to yield any discernible mask results at all.

In the future, we will focus on evaluating our cage clearance measure using robotic experiments. We also aim to generalize our approach to 3D caging as well as caging in the presence of a potential energy field.

VIII. Acknowledgements

This work has been supported by the Knut and Alice Wallenberg Foundation.

References

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," ser. ICRA, vol. 348, 2000, p. 353.

[2] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, pp. 886–900, 2012.

[3] T. Makapunyo, T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, "Measurement framework of partial cage quality," in *ROBIO*, 2012.

[4] E. Rimon and A. Blake, "Caging planar bodies by one-parameter two-fingered gripping systems," *The International Journal of Robotics Research*, vol. 18(3), pp. 299–318, 1999.

[5] P. Pipattanasomporn and A. Sudsang, "Two-finger caging of concave polygon," *ICRA*, pp. 2137–2142, 2006.

[6] M. Vahedi and A. F. van der Stappen, "Caging polygons with two and three fingers," *The International Journal of Robotics Research*, vol. 27(11-12), pp. 1308–1324, 2008.

[7] F. T. Pokorny, J. A. Stork, and D. Kragic, "Grasping objects with holes: A topological approach," in *ICRA*, 2013, pp. 1100–1107.

[8] J. A. Stork, F. T. Pokorny, and D. Kragic, "Integrated motion and clasp planning with virtual linking," in *IROS*, Tokyo, Japan, 2013.

[9] ——, "A topology-based object representation for clasping, latching and hooking," in *HUMANOIDS*, 2013.

[10] A. Varava, D. Kragic, and F. T. Pokorny, "Caging grasps of rigid and partially deformable 3-d objects with double fork and neck features." *IEEE Transactions Robotics*, vol. 32, no. 6, pp. 1479–1497, 2016.

[11] S. Makita and Y. Maeda, "3d multifingered caging: Basic formulation and planning," in *IROS*, 2008, pp. 2697–2702.

[12] S. Makita, K. Okita, and Y. Maeda, "3d two-fingered caging for two types of objects: Sufficient conditions and planning," *International Journal of Mechatronics and Automation*, vol. 3, no. 4, pp. 263–277, 2013.

[13] L. Zhang, Y. J. Kim, and D. Manocha, "Efficient cell labelling and path non-existence computation using c-obstacle query," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1246–1257, 2008.

[14] W. Wan and R. Fukui, "Efficient planar caging test using space mapping," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 278–289, 2018.

[15] Z. McCarthy, T. Bretl, and S. Hutchinson, "Proving path non-existence using sampling and alpha shapes," in *ICRA*, 2012, pp. 2563–2569.

[16] J. Mahler, F. T. Pokorny, Z. McCarthy, A. F. van der Stappen, and K. Goldberg, "Energy-bounded caging: Formal definition and 2-d energy lower bound algorithm based on weighted alpha shapes," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 508–515, 2016.

[17] J. Mahler, F. T. Pokorny, S. Niyaz, and K. Goldberg, "Synthesis of energy-bounded planar caging grasps using persistent homology," *IEEE Transactions on Automation Science and Engineering*, 2018.

[18] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv:1806.10293*, 2018.

[19] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Conference on Robot Learning*, 2017, pp. 515–524.

[20] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *arXiv:1710.01330*, 2017.

[21] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," 2015.

[22] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," vol. 34, no. 4-5, pp. 705–724, 2015.

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

[24] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[25] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *ICRA*, 2016, pp. 3406–3413.

[26] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4243–4250.

[27] M. Gualtieri, A. Ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *IROS*, 2016, pp. 598–605.

[28] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *IROS*, 2016, pp. 4461–4468.

[29] 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," 2017.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[32] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *ICLR*, vol. 5, 2015.