Noname manuscript No.
(will be inserted by the editor)

Partial Caging: A Clearance-Based Definition, Datasets, and Deep Learning

Michael C. Welle · Anastasiia Varava · Jeffrey Mahler · Ken Goldberg · Danica Kragic · Florian T. Pokorny

Received: date / Accepted: date

Keywords Deep Learning in Robotics, Grasping, Computational Geometry, Topological Representation and Abstraction of Configuration Spaces

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 computing 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 planar 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 partial caging 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. Experiments show that evaluation of a given configuration on a GeForce GTX 1080 GPU takes less than 6 ms. Furthermore, an additional dataset focused on grasp-relevant configurations is curated and consists of 772 objects with 3.7 million configurations. We also use this dataset for 2D Cage acquisition on novel objects. We study how network performance depends on the datasets, 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. Influence of disturbances is investigated and quantitative results are provided.

Michael C. Welle, Anastasiia Varava, Danica Kragic, Florian T. Pokorny
KTH Royal Institute of technology, Stockholm , Sweden
E-mail: mwelle, varava, dani, fpokorny@kth.se

Jeffrey Mahler, Ken Goldberg
University of California, Berkeley , USA
E-mail: jmahler, goldberg@berkeley.edu

1 Introduction

A rigid object is *caged* if it cannot escape arbitrarily far from its initial position. From the topological point of view, this can be reformulated as follows: an object is caged if it is located in a bounded connected component of its free space. This notion provides one of the rigorous 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 problem that has been extensively studied analytically. However, since global properties of configuration space may also be estimated more robustly than subtle local geometric features used in classical force closure analysis, caging may hold promise particularly as a noise-tolerant approach to grasping and manipulation.

In its topological formulation, caging is closely related to another global characteristic of configuration spaces – path-connectedness, and, in particular, is a special case of the path non-existence problem [3,4]. This is a challenging problem, as it requires reasoning about the entire configuration space, which is currently not possible to reconstruct or approximate [3,4].

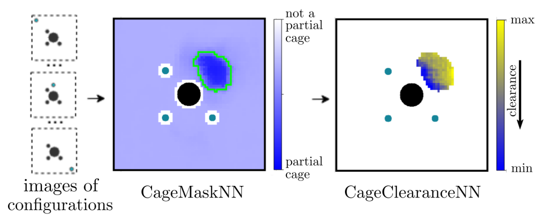

Fig. 1: Given an image of an object (depicted in black) and 3 or 4 caging tools (depicted in green), *CageMaskNN* determines whether a configuration belongs to the "partial cage" subset. If it does, *CageClearanceNN*, evaluates its quality according to the clearance measure learned by the network. On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

Another interesting global characteristic of a configuration space is the maximum clearance of a path connecting two points. In path planning, paths with higher clearance are usually preferred for safety reasons. In contrast, in manipulation, if an object can escape from the manipulator only through a narrow passage, escaping is often less likely. In practical applications, it might be enough to partially restrict the mobility of the object such that it can only escape through narrow passages instead of completely caging it. Such configurations are furthermore less restrictive than full cages, thus allowing more freedom in placing caging tools.

This reasoning leads to the notion of *partial caging*. This generalization of classical caging was first introduced by Makapunyo et al. [5], where the authors define a partial caging configuration as a non-caging formation of fingers that only allows rare escape motions. While [6] and [7] define a similar notion as energy-bounded caging, we propose a partial caging quality measure 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. Assuming motion is random, the quality of a partial cage depends on the width of a "gate" through which the object can escape.

Our quality measure is different from the one proposed in [5], where the authors introduced a measure 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.

One challenge with using sampling-based path planners for partial caging evaluation is that a single configuration requires multiple runs of a motion planner and – in the case of rapidly exploring random tree (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 iteratively 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 partial caging configurations and approximates the partial caging evaluation function (we measured an evaluation time of less than 6 ms for a single given configuration on a GeForce GTX 1080 GPU). 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 1 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.

Furthermore, we explore different shape similarity measures for objects and evaluate them from the partial caging perspective. We propose a way to generate partial caging configurations for previously unseen objects by finding similar objects from the training dataset and applying partial caging configurations that have good quality score for these objects. We compare three different definitions of distance in the space of shapes: Hausdorff, Hamming, and the distance in the latent space of a variational autoencoder (VAE) trained on a set of known objects. Out experiments show that Hamming distance is the best at capturing geometric features of objects that are relevant for partial caging, while the VAE-induced distance has the advantage of being computationally efficient.

This paper is a revised and extended version of our previously published conference submission [8]. The contribution of the extension with respect to the conference paper can be summarized as follows:

1. we define a *grasping band* for planar objects – the area around the object that is suitable for placing caging tools, created a new dataset[1] consisting of partial caging configurations located in the grasping band;
2. we approximate our partial caging quality measure with a deep neural network trained on this new dataset;

[1] `https://people.kth.se/~mwelle/pc_datasets.html`

3. we perform ablation studies to evaluate our deep network architecture;
4. we evaluate the adequacy of our partial caging quality measure by modeling the escaping process as a random walk, and measuring the escape time;
5. we propose a cage acquisition method for novel objects based on known partial caging configurations for similar objects; for this, we explore several different distance metrics;
6. we further evaluate the robustness of the cage acquisition with respect to noise.

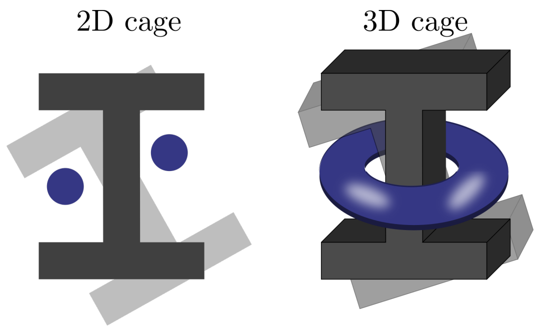

Fig. 2: Example of a 2D cage (left) and a 3D cage exploiting a narrow part of the object.

2 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, an example of a 2D cage can be seen in Fig. 2 on the left-hand side. Rimon and Blake in their early work [9] proposed an algorithm to compute a set of configurations for a two-fingered hand to cage planar non-convex objects. Later, Pipattanasomporn and Sudsang [10] proposed an algorithm reporting all two-finger caging sets for a given concave polygon. Vahedi and van der Stappen in [11] 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 [12, 13, 14]. Another shape feature was later proposed in [15], where we presented a method to cage objects with narrow parts as seen in Fig. 2 on the right-hand side. Makita et al. [16, 17] 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. [18] use approximate cell decomposition to check whether pairs of configurations are disconnected in the free space. Another approach was proposed by Wan and Fukui [19], who studied cell-based approximations of the configuration space based on sampling. McCarthy et al. [3] proposed to randomly sample the configuration space and reconstruct its approximation as a simplicial complex. Mahler et al. [6, 7] extend this approach by defining, 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 approaches 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 (see [20] for an overview of data-driven grasping). 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 [21, 22, 23]. Labels may be curated from human labelers [24, 25, 26], collected from attempts on a physical robot [27, 28], or generated from analysis of models based on physics and geometry [29, 30, 31, 32]. 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, which 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 [32], methods based on dynamic simulation [29, 31], and methods based on geometric heuristics [30]. 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.

3 Partial Caging and Clearance

3.1 Partial Caging

In this section, we discuss the notion of partial caging defined in [8]. Let $\mathcal{C}$ be the configuration space of the object[2], $\mathcal{C}_{col} \subset \mathcal{C}$ 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 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 $p : [0,1] \to \mathcal{C}_{free}$ from a configuration c to X_{esc} is called an escaping path. The set of all possible escaping paths is denoted by $\mathcal{EP}(\mathcal{C}_{free}, c)$.*

Let $cl : \mathcal{EP}(\mathcal{C}_{free}, c) \to \mathbb{R}_+$ 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:

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

3.2 The set $\mathcal{C}_{cage}$

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. 3 (left). In this case, the object can easily escape. However, the clearance of this escaping path will be low, because the object is initially located very close to the caging tool. The same clearance value can be achieved in a much better partial caging configuration, see Fig. 3 (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.

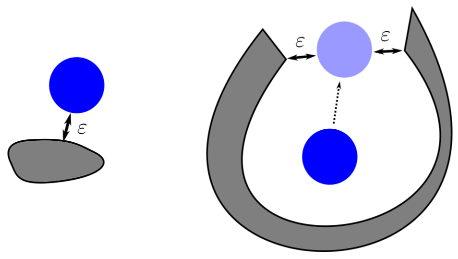

Fig. 3: On the left, an object (blue) can easily escape from the caging tool (grey); on the right, the object is partially surrounded by the caging tool and escaping is therefore harder. Both escaping paths will have the same clearance ε.

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. Fig. 4 illustrates this situation.

Let us formally describe this situation. Let $\mathcal{C}^{\varepsilon}_{free}$ be the free space of the object induced by $\varepsilon-$offset of

[2] Note that in this paper we focus on the case where $\mathcal{C} \subset SE(2)$, but the definition of partial caging holds for arbitrary configuration spaces

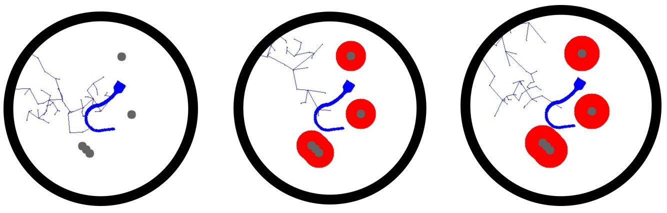
Fig. 4: The object (hook) is shown in blue while the caging tools are gray, the red symbolises the enlargement of the caging tools. The RRT nodes and edges are depicted in purple. From left to right, three enlargements of the caging tools are depicted. The object can always escape until its initial configuration stops being collision-free.

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 robotic manipulation settings, 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. 5)[3].

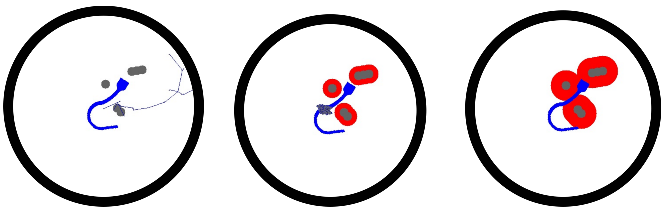
Fig. 5: From left to right: the object (hook) can escape only in the first case, and becomes completely caged when we enlarge the caging tools before a collision with the object occurs.

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. 4.

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.

[3] In Fig. 5 the enlarged caging tools (in red) cage the hook by trapping the larger base.

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 under uncertainty.

4 Gate-Based Clearance Estimation Algorithm

```
Input: object $O$, caging tools $G$, $\varepsilon_{max}$
Output: clearance of an escaping path $\varepsilon_{cl}$
$\varepsilon_{min} \leftarrow 0$;
while Can-Escape($O$, $G$, $\varepsilon_0$) do
    $\varepsilon_{cl} \leftarrow (\varepsilon_{min} + \varepsilon_{max})/2$;
    if Can-Escape($O$, $G$, $\varepsilon_{cl}$) then
        $\varepsilon_{min} \leftarrow \varepsilon_{cl}$
    end
    else
        $\varepsilon_{max} \leftarrow \varepsilon_{cl}$
    end
end
return $\varepsilon_{min}$;
```
Algorithm 1: Gate-Based Clearance Estimation

In this section, we propose a 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. 8. 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. The procedure described in Alg. 1 is then used to estimate $Q_{cl}(c)$.

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 radii 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 [4] 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

[4] 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.

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, CanEscape(O, G, ε_{cl}) returns *True* if the object can escape and is in a collision-free configuration.

5 Grasping favorable configuration in $\mathcal{C}_{cage}$

Depending on the size of the object with respect to the workspace, the bisection search performed in Alg. 1 can be computationally expensive. Uniformly sampling caging tools placements from the entire workspace in order to find configurations in $\mathcal{C}_{cage}$ is also rather inefficient and the number of partial caging configurations of high quality can be low.

Furthermore, not all partial caging configurations defined by Def. 3 ($c \in \mathcal{C}_{cage}$) are equally suitable for certain applications like grasping or pushing under uncertainty. Namely, we would like to place caging tools such that they are not too close and not too far away from the object.

To overcome these limitations, we define a region around the object called *partial caging grasping band* (Fig. 6 illustrates this concept):

Definition 4 *Let O be an object and assume the caging tools have a maximal width*[5] *ct_d. Let O_{min} and O_{max} be objects where the composing disks are enlarged by $dis_{min} = \frac{1}{2}ct_d \cdot (1 + \beta)$ and $dis_{max} = dis_{min} + \frac{1}{2}ct_d \cdot \gamma$ respectively.*

We can then define the grasping band *as follows:*

$$\mathcal{GB} = \{x \in \mathcal{C}_{free} : (x \in O_{min}) \oplus (x \in O_{max})\},$$

Here, β and γ are parameters that capture the impreciseness of the system, such as vision and control uncertainties.

6 Learning Planar Q_{cl}

As RRT is a non-deterministic algorithm, one would need to perform multiple runs in order to estimate Q_{cl}. In real-time applications, we would like the robot to be able to evaluate caging configurations within milliseconds. Thus, the main obstacle on the way towards using the partial caging evaluation function defined above in real time is the computation time needed to evaluate a single partial caging configuration.

[5] The caging tools are composed of disks with ct_d as diameter. As we only consider composed line configurations as caging tools the width never exceeds ct_d.

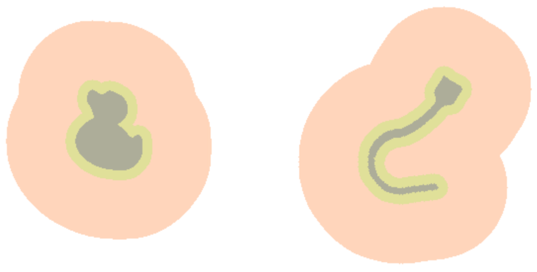
Fig. 6: An illustration of a grasping band for a duck and hook object. The object O is in the center (gray) overlaid by O_{min} (O enlarged by dis_{min}, light green) overlaid by O_{max} (O enlarged by dis_{max}, light orange). The grasping band ($\mathcal{GB}$) is the disjunctive union between O_{min} and O_{max}.

Alg. 1 requires several minutes to evaluate a single partial cage, while a neural network can potentially estimate a configuration in less than a second.

To address this limitation of Alg. 1, we design and train two convolutional neural networks. The first, called *CageMaskNN*, acts as a binary classifier that identifies configurations that belong to $\mathcal{C}_{cage}$ following Def 3. The second, architecturally identical network, called *CageClearanceNN*, approximates the caging evaluation function Q_{cl} to estimate the quality of configurations. The network takes two images as input that correspond to the object and the caging tools.The two networks are separated to make training more efficient, as both can be trained independently. Operating both networks sequentially results in pipeline visualized in Fig. 1: first, we identify if a configuration is a partial cage, and if it is, we evaluate its quality.

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 in a particular configuration. We assume that caging tools are normally disconnected, while objects always have a single connected component. In our current implementation, we consider $n \in \{3, 4\}$, and multiple caging tool shapes.

While neural networks require a significant time to train (often multiple hours), evaluation of a single configuration is a simple forward pass through the network and its complexity is therefore not relying on the input size or data size but rather on the number of neurons in the network. In this work, our goal is to show that we can successfully train a neural network that can generalise to unseen input configurations and approximate the algorithm 1 in milliseconds.

6.1 Dataset Generation

We create a dataset of 3811 object models consisting of two-dimensional slices of objects' three-dimensional mesh representations created for the Dex-Net 2.0 framework [32]. We further approximate each model as a union of one hundred discs, to strike a balance between accuracy and computational speed. The approximation error is a ratio that captures how well the approximation (A_{app}) represents the original object (A_{org}, and is calculated as follows: $a_e = \frac{A_{org}-A_{app}}{A_{org}}$. Given the set of objects, two partial caging datasets are generated. The first dataset, called *PC-general*, consists of 3811 objects, 124435 partial caging configurations (belonging to $\mathcal{C}_{cage}$), and 18935565 configurations that do not belong to $\mathcal{C}_{cage}$.

One of the limitations of the *PC-general* dataset is that it contains relatively few partial caging configurations of high quality. To address this limitation, generate a second partial caging dataset called *PC-band* where caging tools placements are only located inside the grasping bands of objects, as this strategy increases the chance that the configuration will be a partial cage of low Q_{cl} as well as the likelihood of a configuration belonging to $\mathcal{C}_{cage}$.

The *PC-band* dataset consists of 772 object with 3,785,591 configurations of caging tools, 127,733 of which do belong to the partial caging subset $\mathcal{C}_{cage}$. We set β to the approximation error a_e for each object and $\gamma = 6$ to define the grasping band.

All configurations are evaluated with Q_{cl} (see algorithm 1). The distribution of partial cages can be seen in Fig. 7.

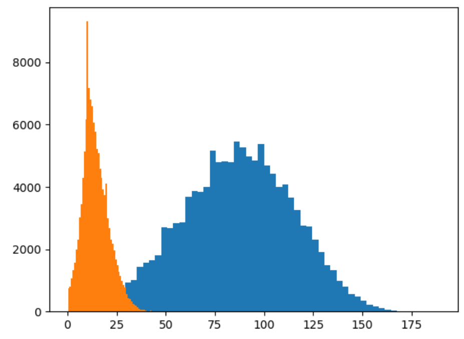

Fig. 7: Distribution of Q_{cl} estimates for the *PC-general* (blue) and the *PC-band*(orange) datasets.

Examples of configurations for both datasets can be seen in Fig. 8. The disk approximation of the object is shown in blue, while the original object is depicted in red. *PC-general* contains configurations placed in the entire workspace while *PC-band* is limited to configuration sampled inside the grasping band.

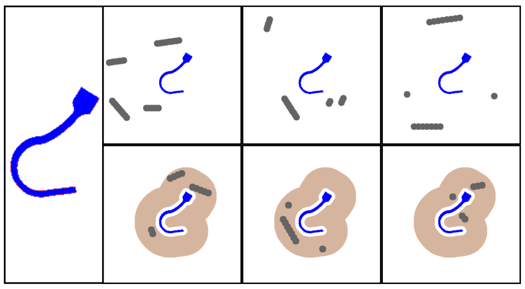

Fig. 8: Left: original representations of a hook objects (red) and in blue their approximation by a union of discs of various sizes closely matching the polygonal shape ($a_e = 0.051$); second and third column: configurations that do not belong to $\mathcal{C}_{cage}$; last column: a partial caging configuration($c \in \mathcal{C}_{cage}$). The top row is from *PC-general*, the bottom from *PC-band*.

6.2 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 [33]. The idea is that the global geometric structure can be best captured with different image sizes, such that the three different branches can handle scale-sensitive features. 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.

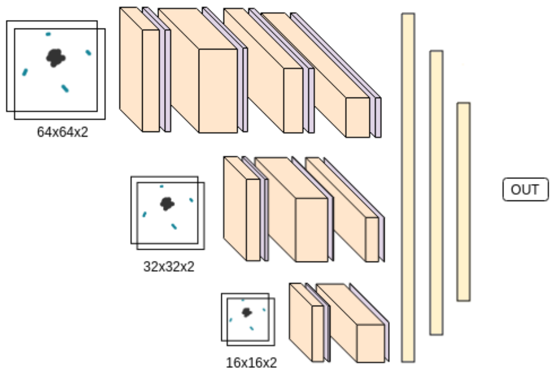

Fig. 9: As caging depends on global geometric properties of objects, a CNN architecture with multi-resolution input was designed to capture these features efficiently.

The architecture of the networks is shown in Fig. 9. Both networks take an image of an object and caging tools on a uniform background 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 the Tensorflow [34] implementation of the Adam algorithm [35]. The loss is defined as the mean-squared-error (MSE) between the prediction and the true label. The batch size was chosen to be 100 in order to compromise between learning speed and gradient decent accuracy. The networks were trained on both of our datasets – *PC-general* and *PC-band*.

7 Training and evaluation of the networks

In this section we describe how we train and evaluate the two networks and perform an ablation study of the architecture. In detail, for *CageMaskNN*, we investigate to what extent the training data should consist of samples belonging to $\mathcal{C}_{cage}$ and evaluate the performance of the best such composition against a simpler network architecture. Following that, we investigate how the number of different objects as well as the choice of dataset influences the performance of *CageMaskNN*.

For *CageClearanceNN*, we also perform an analysis of the effect of the the number of objects in the training data and to what extent the choice of dataset influences the performance and compare it to a simpler architecture. As a final investigation, we investigate the error for specific Q_{cl} intervals.

Note that the training data is composed of samples where the ground truth of the configuration was obtained using algorithm 1. A main goal of the presented evaluation is hence to investigate how well the proposed networks are able to generalise to examples that were not included in the training data (unseen test data). High such generalization performance, is a key indicator for the potential application of the proposed fast neural network based approach (execution in milliseconds) instead of the computationally expensive underlying algorithm 1 (execution in minutes) that was used to generate the training data.

Single-res Architecture: In order to perform an ablation of the previous discussed multi-resolution architecture we compare the performance so a architecture that has only a single resolution as input. The *Single-res Arch.* Takes only the 64x64x2 as input and is missing the other heads completely. In this way we want to see if our assumption that different sized inputs are beneficial to the networks performance.

7.1 CageMaskNN - % of $\mathcal{C}_{cage}$ and Ablation

We generate 4 datasets containing 5%, 10%, 15%, and 20% caging configurations in $\mathcal{C}_{cage}$ respectively from *PC-general*. This is achieved by oversampling as well as by performing rotational augmentation with 90, 180 and 270 degrees of the existing caging configurations. The *Single-res Arch.* is trained with 10% caging configurations in $\mathcal{C}_{cage}$ for comparison.

The evaluation is performed on a test set consisting of 50% caging examples from $\mathcal{C}_{cage}$. In Fig. 10, we show the F1-curve and Accuracy-curve. All five 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. The MSE for the single resolution network was 0.155 showing the significant gain obtained by utilizing the multi-resolution branches.

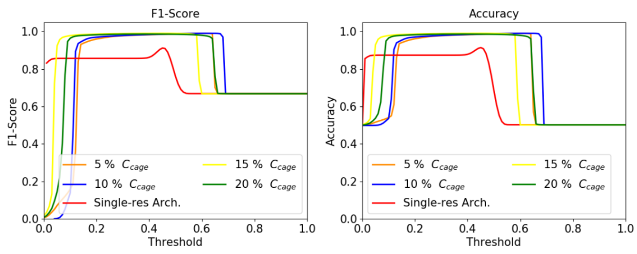

Fig. 10: F1-score and accuracy of the network depending on different thresholds

We observe that the network that was trained on the dataset where 10% of the configurations are partial cages performs slightly better than the other versions. Note however that only the one that was trained with 5% of partial cages performs significantly worse. All versions of the multi-resolution architecture outperform the *Single-res Arch*, which justifies our architecture design.

7.2 CageMaskNN - Number of Objects and Datasets

We investigate how the performance of the networks depends on the size of the training data and how the two training datasets, *PC-general* and *PC-band*, affect the performance of the networks. Table 1 shows the area under ROC curve (AUC) andthe average precision (AP) for *CageMaskNN* for training set composed of 1, 10, 100, and 1000 objects from the dataset *PC-general*, as well as 1, 10, 100, and 617 objects from *PC-band*. We observe that having more objects in the training set results in better performance. We note that the network trained on *PC-general* slightly outperforms the one trained on *PC-band*.

Training set	**pc-general**		**pc-band**	
	AUC	**AP**	**AUC**	**AP**
1 object	0.92	0.88	0.88	0.83
10 objects	0.91	0.88	0.88	0.84
100 objects	0.97	0.92	0.92	0.89
1000 ‖ 617 objects	1.00	1.00	1.00	0.96

Table 1: The area under ROC curve (AUC) and the average precision (AP) for different training set constitutions, evaluated on the test set with 50 % of partial cage configurations. In all training sets 10 % of configurations belong to C_{cage}. We observe that *PC-general* has a slightly better performance than *PC-band*.

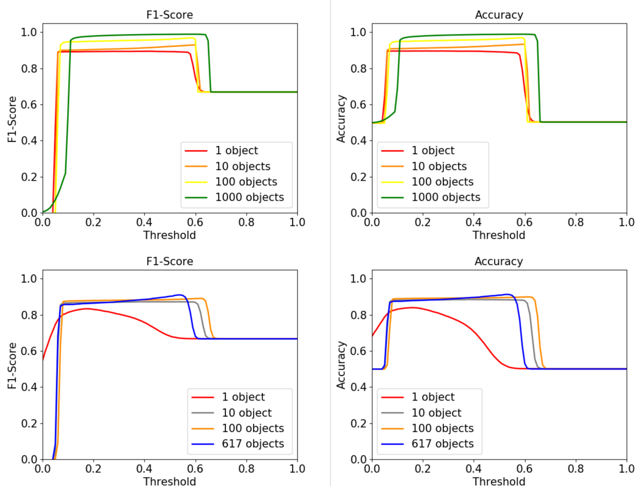

Fig. 11: F1-score and accuracy of the network trained with 1, 10, 100, and 1000 ‖ 617 objects,for *PC-general* (top row) and *PC-band* (bottom row) respectively on a test set with 50 % C_{cage} configuration .

Fig. 11 demonstrates how the performance of the networks increases with the number of objects in the training dataset by showing the F1-score as well as the accuracy for both datasets. We observe that the network, independently of the training dataset, demonstrates acceptable performance even with a modest numbers of objects in the training dataset. One key factor here is the validation set which decreases the generalisation error by choosing the best performance during the entire training run, thus reducing the risk of overfitting. Similarly to the previous results, *PC-general* slightly outperforms *PC-band*.

7.3 CageClearanceNN - Number of Objects and Ablation

The purpose of *CageClearanceNN* is to predict the value of the clearance measure Q_{cl} given a partial caging configuration. We trained *CageClearanceNN* on 1, 10, 100 , 1000 and 3048 objects from *PC-general* as well as a single resolution variant with the same training sets. Additionally, we trained another instance of *CageClearanceNN* with 1, 10, 100, and 617 objects from *PC-band*, and the corresponding single-resolution architecture version for each number of objects. The label is scaled with a factor of 0.1, as we found that the networks performance improves for smaller training input values. The left-hand side of Fig.12 shows a rapid decrease of MSE as we increase the number of training data objects to 1000, and a slight performance increase between 1000 and 3048 training objects for the *PC-general* dataset. We can also see that employing the multi-resolution architecture only leads to significant performance increase when going up to 1000 objects and more. The right-hand side of Fig.12 presents the analogous plot for the network trained on *PC-band*. We observe the same rapid decrease of MSE as we include more objects in the training set. Note that the different number of parameter plays a role as well in the performance difference. Since our current dataset is limited to 617 training examples of object shapes, we do not observe the benefits of the multi-resolution architecture. Note that the difference in absolute MSE stems from the different distributions of the two datasets (as can be seen in Fig. 7). This indicates that further increases in performance can be gained by having more training objects. Increasing the performance for more than 3000 objects may however require a significant upscaling of the training dataset.

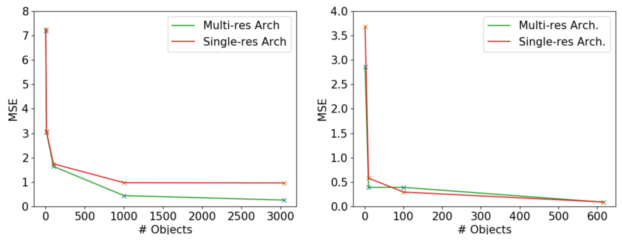

Fig. 12: left: MSE of *CageClearanceNN* trained on *PC-general* with different numbers of objects and a single-resolution architecture; right: MSE of the single-resolution architecture trained on *PC-band* with different numbers of objects.

7.4 CageClearanceNN - Error for specific Q_{cl}

We investigated the MSE for specific Q_{cl} value intervals. Fig. 13 shows the MSE on the test set with respect to the Q_{cl} values (as before, scaled by 0.1). Unsurprisingly, we observe that the network, trained on *PC-general*, that was trained only on one object, does not generalise over the entire clearance/label spectrum. As we increase the number of objects, the performance of the network increases. The number of outliers with large errors decreases significantly when the network is trained on 1000 objects. On the right side, we can see the MSE for the final *CageClearanceNN* network trained on *PC-general*. We observe that low values of Q_{cl} are associated to higher error values. Analysing this behavior on *CageClearanceNN* trained on *PC-band* demonstrates a very similar behavior and is therefore omitted.

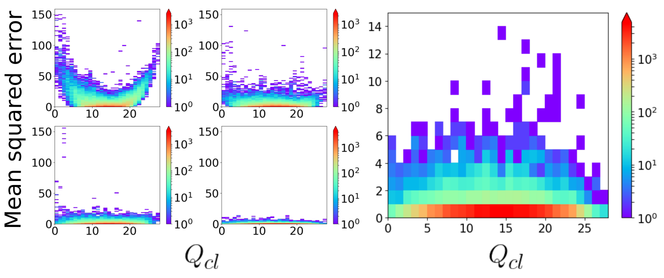

Fig. 13: MSE for each test case sorted for labels. Left: shows performance of 1, 10, 100, 1000 objects (top left, top right, bottom left, bottom right). Right: shows MSE of entire test set for the final *CageClearanceNN*. Note that the figure on the right is zoomed in as errors are significantly smaller (see the left y-axis of that figure).

8 Planar Caging Pipeline Evaluation

8.1 Last 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. 15 illustrates this approach. We use the pipeline trained with *PC-general* as it covers the entire workspace. 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.

8.2 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. 14. 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.The offset in *CageClearanceNN* is larger but consistent, which can be explained by the fact that similar objects seen during training had a lower clearance as the novel hourglass shaped object. In the second example, the clearance of the partial cage decreases continuously as the caging tools get closer to the object. Predicted clearance values from *CageClearanceNN* 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*.

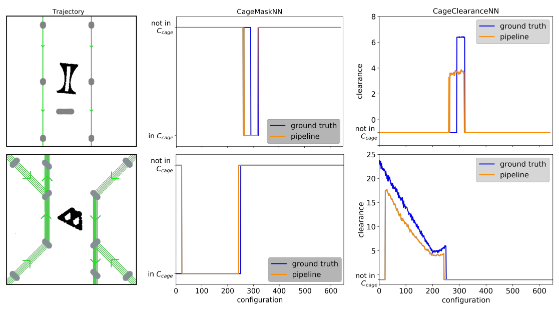

Fig. 14: Evaluation of the pipeline along two trajectories. The trajectory (left, green) is evaluated with *CageMaskNN* (middle) and *CageClearanceNN* (right), which evaluates Q_{cl} for those configurations where *CageMaskNN* returns 0. The predictions by the networks are displayed in orange while ground truth is shown in blue.

8.3 Experimental evaluation of Q_{cl}

In this section, we experimentally evaluate our partial caging quality measure Q_{cl} by simulating random shaking of the caging tools and measuring the needed time for the object to escape. Intuitively, the escape time should be inversely proportional to the estimated Q_{cl}; this would indicate that it is difficult to escape the partial cage. A similar approach to partial caging evaluation has been proposed in [5]. Where the escape time was computed using probabilistic motion planning methods like RRT, RRT*, PRM, SBL as well as a random planner was measured.

8.3.1 Random partial caging trajectories

We apply a simple random walk X_n as a sequence of independent random variables $S_1, S_2, ..., S_n$ where each S is is randomly chosen from the set $\{(1,0),(0,1),(1,1),(-1,0),(0,-1),(-1,-1)\}$ with equal probability.

$$X_n = X_0 + S_1 + S_2 + ... + S_n),$$

where X_0 is the start position of the caging tools. and a stride factor α determines at what time the next step of the random walk is performed.

In this experiment, unlike in the rest of the paper, caging tools are moving along randomly generated trajectories. We assume that the object escapes a partial cage when it is located outside of the convex hull of the caging tools. If the object does not escape within t_{max} seconds, the simulation is stopped. The simulation is performed with the software pymunk that is build on the physic engine Chipmunk 2D [36]. We set the stride factor $\alpha = 0.05s$ so that a random step S of the random walk X_n is applied to the caging tool every 0.05 seconds. As pymunk also facilitates object interactions, the caging tool can push the object around as well as drag it with them. Figure 16 illustrates this process.

The experiment was performed on 5 different objects, depending on the object we used between 437-1311 caging tool configurations. For each of them the escape time was estimated as described above. As it is not deterministic, we performed 100 trials for each configuration and computed the mean value. The mean escape time of 100 trials was normalized such that the values range between 0 and 1. Furthermore, for each configuration we computed Q_{cl} and the Pearson correlation coefficient[6]. Fig. 17 illustrates the results.

Our results show that the longer it takes for the object to escape the partial cage, the higher the variance of the escape time is. This indicates that a partial cage quality estimate based on the average escape time would require a high number of trials, making the method inefficient.

Furthermore, we demonstrate that our clearance-based partial caging quality measure shows a trend with the average escape time for strong partial cages, which suggests the usefulness of the proposed measure.

8.4 Different metrics in the space of shapes for partial caging

A natural extension of our partial caging evaluation framework is partial cage acquisition: given a previously unseen object, we would like to be able to quickly synthesise partial cages of sufficient quality. In this section, we make the first step in this direction, and propose the following procedure: given a novel object, we find similar objects from the training set of the *PC-band*, and consider those partial caging configurations that worked well for these similar objects.

The key question here is how to define a distance function for the space of objects that would capture the most relevant shape features for partial caging. In this experiment, we investigate three different shape distance functions: Hausdorff distance, Hamming distance, and Euclidean distance in the latent space of a variational autoencoder, trained on the set of objects used in this work. Variational autoencoders (VAEs) are able to encode high-dimensional input data into a lower-dimensional latent space while training in an unsupervised manner. In contrast to a standard encoder/decoder setup, which returns a single point, a variational autoencoder returns a distribution over the latent space, using the KL-cost term as regularisation.

[6] The Pearson correlation coefficient measures the linear correlation between the escape time from random shaking and the defined clearance measure Q_{cl}.

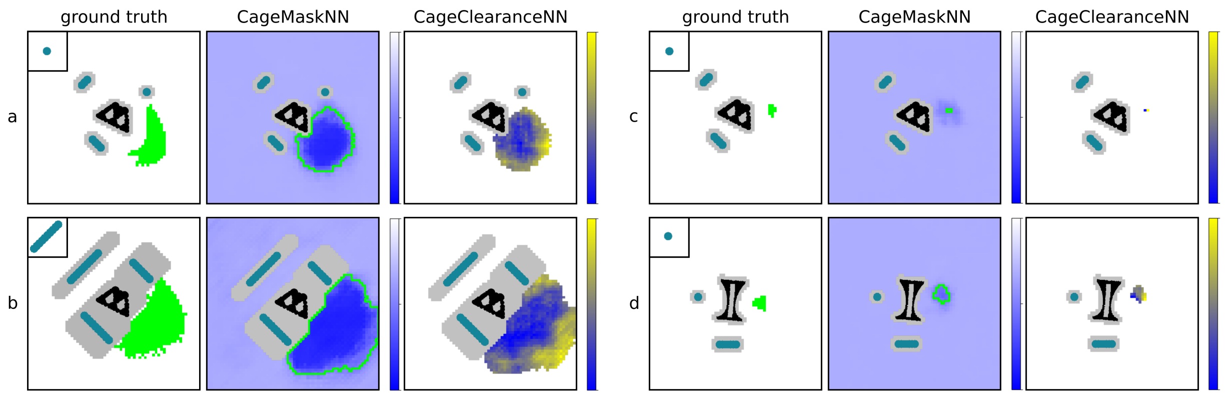

Fig. 15: Here, we depict the results of four different experiments. The green region indicates configuration where the additional caging tool completes the configuration in such a way that the resulting configuration is a partial cage. The small squares in the ground truth figures depict the caging tools that are being placed (for simplicity the 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.

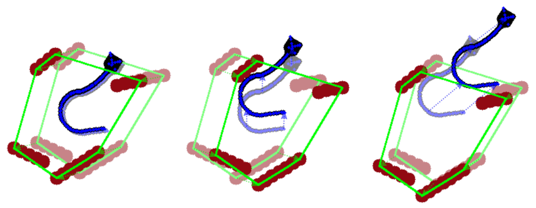

Fig. 16: Random trajectory for caging tools. Left: time $t = 0s$(transparent) to $t = 0.83s$(not escaped), middle: $t = 0.83s$(transparent) to $t = 1.67s$ (not escaped), right: time $t = 1.67s$ (transparent) to $t = 2.47s$ (escaped). Note that the caging tools do not necessarily run in a straight line but rather follow the randomly generated trajectory with a new step every $0.05s$. As a simple physics simulator is used, the caging tools can also induce movement of the object by colliding with it.

We evaluate different distance functions with respect to the quality of the resulting partial cages. Given a novel object, we calculate the distance to each known object in the dataset according to the three distance functions under consideration, and for each of them we select five closest objects. When comparing the objects, orientation is an important factor.We compare 360 rotated versions of the novel object with the known objects from the dataset and pick the one closest following the chosen metric.

8.4.1 VAE-based representation

For our experiment, we train a VAE based on the ResNet architecture with skip connections with six blocks [37] for the encoder and the decoder. The imput images have resolution 256x256. We use a latent space with 128 dimensions, dropout of 0.2 and a fully connected layer of 1024 nodes. The VAE loss was defined as follows:

$$\mathcal{L}_{vae}(x) = E_{z\sim q(z|x)}[\log p(x|z)] + \beta \cdot D_{KL}(q(z|x)||p(z))$$

The first term achives reconstruction, while the second term tries to disentegel the destinct features. z denotes latent variable, $p(z)$ the prior distribution,and $q(z|x)$ the approximate posterior distribution. Note that the Bernoulli distribution was used for $p(x|z)$, as the images are of a binary nature. The batch size was set to 32. As the sizes of the objects vary significantly, we invert half of the images randomly when loading a batch. This prevents the collapse to either pure black or pure white images.

8.4.2 Hausdorff distance

The Hausdorff distance is a well known measure for the distance between two sets of points in a metric space ($\mathbb{R}^2$ for our case). As the objects are represented with disks we use the set of x and y points to represent the object. This is a simplification of the object as the radius of the circles is not considered. The general Hausdorff distance can be computed with [38]:

$$d_H(X,Y) = \max\left\{\sup_{x\in X}\inf_{y\in Y} \mathrm{d}(x,y), \sup_{y\in Y}\inf_{x\in X} \mathrm{d}(x,y)\right\}$$

8.4.3 Hamming Distance

The Hamming distance [39] is defined as the difference of two binary data strings calculated using the XOR operation. It captures the exact difference between the two images we want to match, as it calculates how many

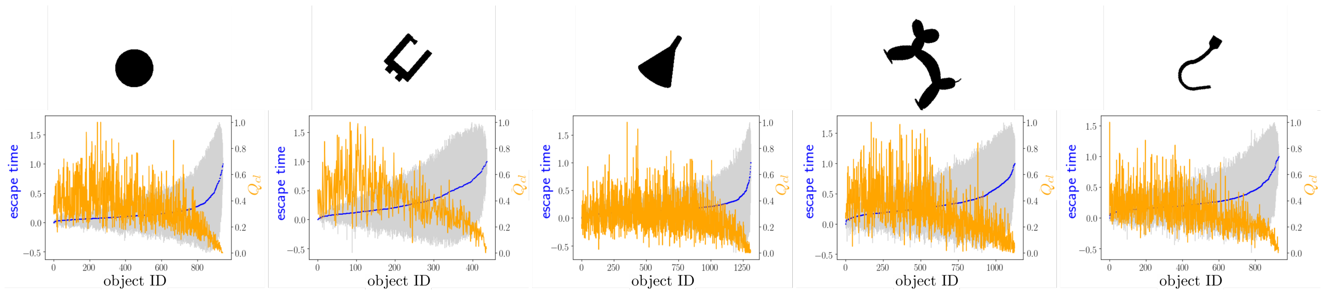

Fig. 17: Correlation between escape time from random shaking and Q_{cl}. Top row shows evaluated objects (disk, clench, cone, balloon animal, and hook, on the bottom row the partial cages are sorted according to respective average escape time, and plot the average escape time (in blue), its variance (in gray), and Q_{cl} (in orange). Pearson correlation coefficient of the escape time and Q_{cl} (from left to right) are: -0.608, -0.462, -0.666, -0.566, -0.599

pixel are different. We pre-process the images by subtracting the mean and reshaping the images to a 1D string.

8.5 Performance

We compare the performance of the three different similarity measures, as well as a random selection baseline, on 500 novel object. The percentage of collision-free caging tools placements, as well as the average clearance score is shown in Table 2. We report the average percentage of collision-free caging tool placements taken from the *PC-band* of partial cages for top 1 and top 5 closest objects.

Furthermore, we evaluate the collision-free configurations using Alg. 1 to provide Q_{cl} values as well as check if the configuration still belongs to $\mathcal{C}_{cage}$. In the Table 2, the top 1 column under cage evaluation shows the percentage of configurations that belong to $\mathcal{C}_{cage}$. To the right is the average Q_{cl} for the most promising cage from the closest object. The top 25 column shows the same results for the five most promising cages for each of the five closest objects. Examples for three novel objects and the closest retrieved objects are shown in Fig. 18. In the left column, the closest objects with respect to the chosen metric are shown given the novel query object. The right column shows the acquired cages, transferred from the closest known objects. Note that a collision free configuration does not necessarily have to belong to $\mathcal{C}_{cage}$.

For the VAE-model, it takes approximately 5 milliseconds to generate the latent representation, any subsequent distance query can then be performed in 0.005 milliseconds. The Hausdorff distance requires 0.5 milliseconds to compute, while the Hamming distance takes 1.7 milliseconds per distance calculation[7].

[7] The time was measured on a Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.

	collision-free		cage evaluation			
			top 1		top 25	
	top 1	top 5	$\in \mathcal{C}_{cage}$	Q_{cl}	$\in \mathcal{C}_{cage}$	Q_{cl}
VAE	90.9%	86.6%	53.2%	5.05	48.4%	5.92
Hausdorff	75.5%	75.2%	33.6%	5.21	32.2%	5.83
Hamming	74.4%	73.9%	35.4%	3.87	34.3%	4.51
Random	61.6%	62.3%	27.0%	13.31	25.4%	14.12

Table 2: Average results for 500 novel objects cage acquisition using different distance metrics to find similar objects in *PC-band*, and applied cages from retrieved objects to novel objects.

Our experiments show that, while the VAE-induced similarity measure performs best in terms of finding collision-free caging tools placements, Hamming distance significantly outperforms it in terms of the quality of acquired partial cages. We did not observe a significant difference between Hausdorff distance and the VAE-induced distance. While Hamming distance appears to be better at capturing shape features that are relevant for cage acquisition task, it is the least efficient approach in terms of computation time. Furthermore, in our opinion, VAE-induced distance may be improved significantly if instead of using a general-purpose architecture we introduce task-specific geometric and topological priors.

9 Limitations and Challenges for Future Work

In this section, we discuss the main challenges of our work and the possible ways to overcome them.

9.1 Data generation challenges

One of the main challenges in this project is related to data generation: we need to densely sample the space of the caging tools' configurations, as well as the spaces of shapes of objects and caging tools. This challenge is especially significant when using the *PC-general* dataset,

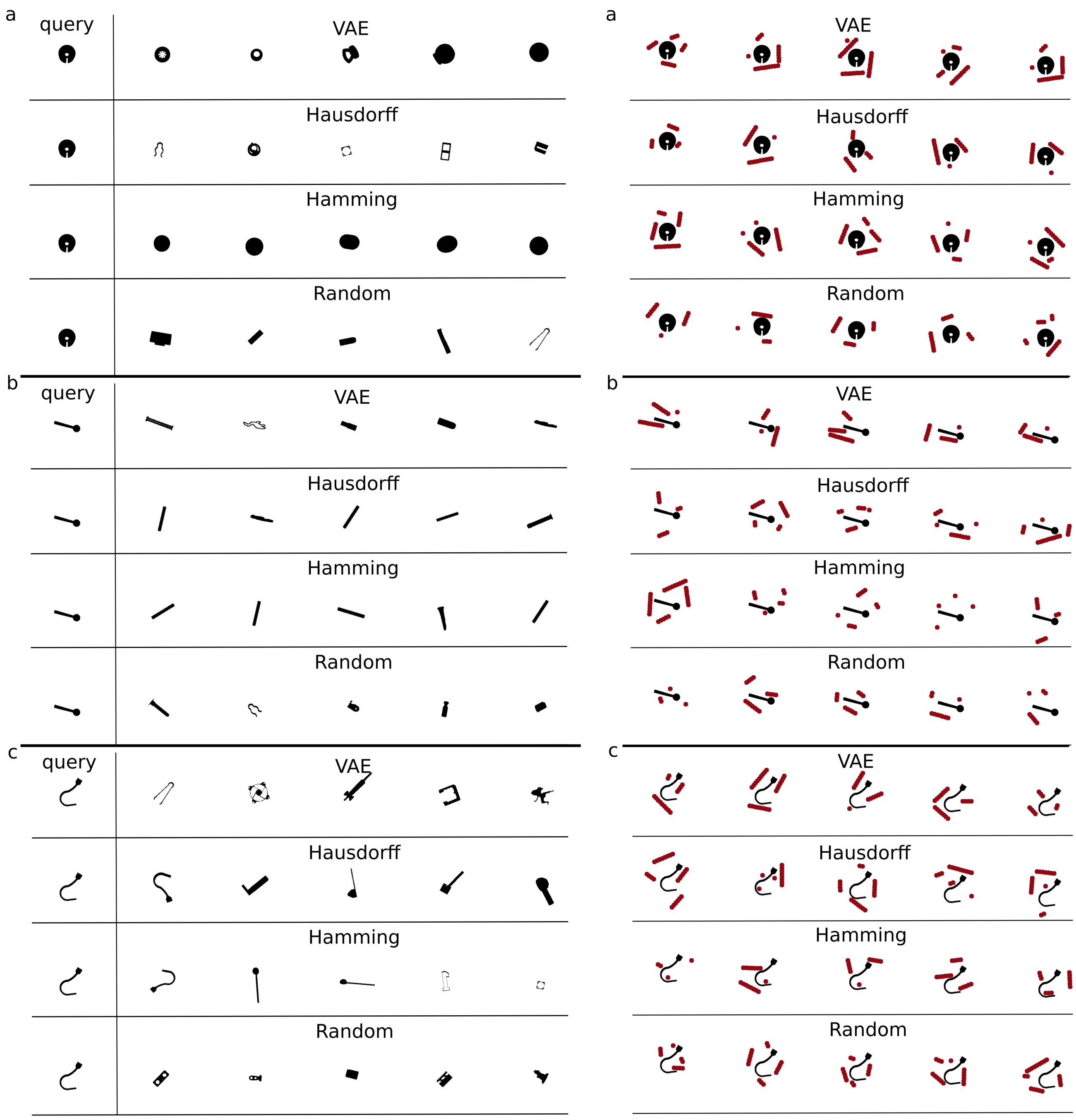

Fig. 18: On the left-hand side, we consider 3 different query objects washer(a), pin(b) and hook(c), and for each distance function visualize respective 5 closest objects from the training dataset; on the right-hand side, for each of the query object (a)-c)) and each distance function, we visualize the acquired partial caging configurations.

as the space of possible caging tools configurations is large.

While the experimental evaluation indicates that the chosen network architecture is able to achieve low MSE on previously unseen objects, in applications one may want to train the network with either a larger distribution of objects, or a distribution of objects that are similar to the objects that will be encountered in practice.

In Fig.19, we illustrate how a lack of training data of sufficiently similar shapes can lead to poor performance of *CageMaskNN* and *CageClearanceNN*, for example, when only 1, 10, 100, or 1000 objects are used for training. Similarly, even when the networks are trained on the full training dataset of 3048 objects, the subtle geometric details of the partial caging region cannot be recovered for the novel test object, requiring more training data and further refinement of the approach.

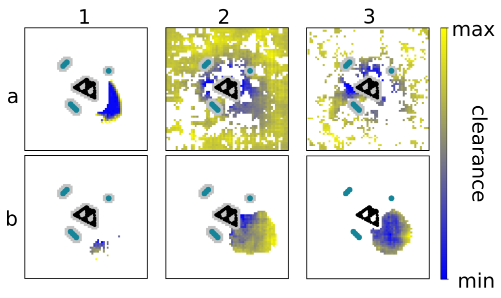

Fig. 19: Performance of *CageMaskNN* and *CageClearanceNN* given different numbers of training objects and evaluated on a single novel object. The top left (a1) displays the ground truth mask and clearance values for a fourth missing disc-shaped caging tool, a2: only 1 object is used for training, a3:10 objects are used for training, b1: 100 objects, b2: 1000 objects, b3: all 3048 objects are used for training. Note that the threshold had to be adjusted to 0.6 for the single object (a2) and 0.61 for the 10 object case (a3) to yield any discernible mask results at all.

9.2 Robustness under noise

In the cage acquisition scenario, the VAE-induced and Hamming distances work directly with images, and hence can be susceptible to noise. To evaluate this effect, we generate salt and pepper noise as well as Gaussian blur and analyse the performance of the VAE-induced and Hamming metrics under four different noise levels (0.005%, 0.01%, 0.05%, 0.1%) and four different kernel sizes (11x11, 21x21, 41x41, 61x61) [8]. Fig. 20 shows the result of the top 3 retrieved objects for the hook object. Left column shows the query objects with respective disturbance. The next three columns depict the closest objects retrieved according to the VAE-induced metric, while the last three columns show the objects retrieved with Hamming metric.

Table 3 reports the performance with respect to finding collision-free configurations, configurations belonging to $\mathcal{C}_{cage}$, and their average values of Q_{cl}. The results are averaged over 500 novel objects. We can see that the VAE-induced metric is affected by strong salt and pepper noise as the number of generated collision-free and partial caging configurations decreases. Furthermore, the resulting Q_{cl} of the generated partial cages increases, meaning it is easier to escape the cage.

[8] Note that sigma is calculated using the standard OpenCV [40] implementation ($\sigma = 0.3 \cdot ((ksize - 1) \cdot 0.5 - 1) + 0.8$).

According to the experiment, the Hamming distance-based lookup is not significantly affected by salt and pepper noise. One explanation here may be that this kind of disturbance leads to a uniform increase of the Hamming distance for all objects. The Gaussian blur has a more negative effect on the Hamming distance lookup then the VAE-based lookup, as can be seen in the retrieved example objects in Fig. 20. Table 3 shows small decrease in the percentage of collision-free and partial caging configurations. Interestingly, the quality of the partial cages does not decrease.

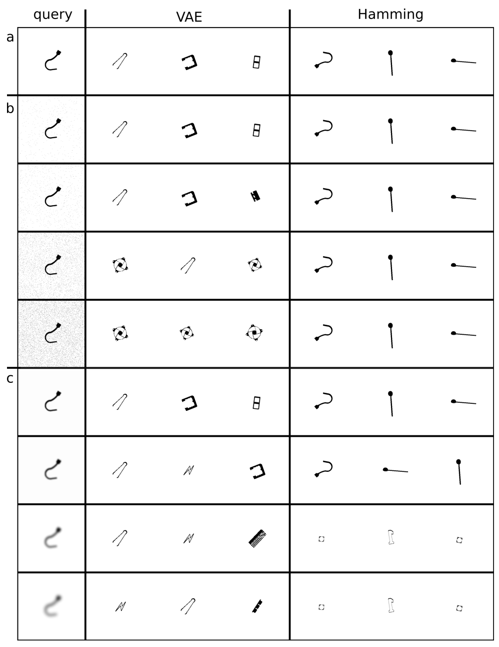

Fig. 20: Top three retrieval results for query images with different levels of disturbance for the VAE-induced and Hamming metric. *a* results without disturbance, *b* show retrieval for different level of salt and pepper noise, *c* retrieved objects when Gaussian blur is applied to query object (hook).

Dist.	VAE			Hamming		
	$\mathcal{C}_{free}$	$\mathcal{C}_{cage}$	Q_{cl}	$\mathcal{C}_{free}$	$\mathcal{C}_{cage}$	Q_{cl}
no dist.	90.9 %	53.2 %	5.05	74.4 %	35.4 %	3.87
S&P 0.005	91.1 %	52.8 %	4.94	74.9 %	34.5 %	3.86
S&P 0.01	90.5 %	52.4 %	4.98	74.6 %	35.4 %	3.88
S&P 0.05	83.4 %	45.0 %	10.12	71.8 %	35.0 %	3.81
S&P 0.1	83.2 %	43.5 %	10.95	73.2 %	33.8 %	3.87
Gb 11x11	91.8 %	52.5 %	4.93	73.7 %	34.6 %	3.78
Gb 21x21	90.3 %	52.8 %	4.83	73.1 %	33.8 %	3.78
Gb 41x41	84.7 %	49.2 %	4.59	69.8 %	33.4 %	3.74
Gb 61x61	84.7 %	45.3 %	4.21	68.0 %	29.3 %	3.73

Table 3: Performance for VAE-induced and Hamming metrics given different level of salt and pepper noise as well as Gaussian blur for different kernel sizes.

9.3 Real World Example and Future Work

As the VAE-framework just takes an image in order to propose suitable cages for a novel object, we showcase a concluding application example in Fig. 21 where a novel object (a hand drill) is chosen as input to the VAE cage acquisition. The image is preprocessed by a simple threshold function to convert it to a black and white image, next the closest object from the dataset are found by comparing the distances in the latent space of the VAE and the three best partial caging configurations are retrieved and applied to the novel object.

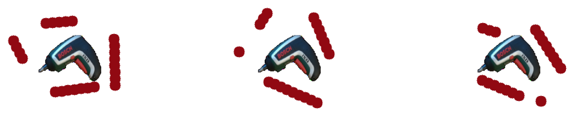
Fig. 21: Proposed partial cages using the VAE cage acquisition method. The novel object (hand drill) is feed into the cage acquisition and the best three cages from the closest object in the dataset are shown (in red).

In the future, we would like to extend our approach to 3-dimensional objects, As illustrated in Fig. 22, partial cages may be a promising approach for transporting and manipulating 3D objects without the need for a firm grasp, and fast learning based approximations to analytic or planning based methods may be a promising direction for such partial 3D cages. Furthermore, we would also like to to investigate the possibility of leveragingother caging verification methods such as [4] for our approach.

Fig. 22: An example for future partial caging in 3D. A complex object needs to be safely transported without the need to firmly grasp it.

10 Acknowledgements

This work has been supported by the Knut and Alice Wallenberg Foundation, Swedish Foundation for Strategic Research and Swedish Research Council.

References

1. A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 348–353.
2. A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, pp. 886–900, 2012.
3. Z. McCarthy, T. Bretl, and S. Hutchinson, "Proving path non-existence using sampling and alpha shapes," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2563–2569.
4. A. Varava, J. F. Carvalho, F. T. Pokorny, and D. Kragic, "Free space of rigid objects: Caging, path non-existence, and narrow passage detection," in *Workshop on Algorithmic Foundations of Robotics*, 2018.
5. T. Makapunyo, T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, "Measurement framework of partial cage quality," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2012, pp. 1812–1816.
6. 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.
7. 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.
8. A. Varava, M. C. Welle, J. Mahler, K. Goldberg, D. Kragic, and F. T. Pokomy, "Partial caging: A clearance-based definition and deep learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1533–1540.
9. 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.
10. P. Pipattanasomporn and A. Sudsang, "Two-finger caging of concave polygon," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 2137–2142.
11. 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.
12. F. T. Pokorny, J. A. Stork, and D. Kragic, "Grasping objects with holes: A topological approach," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1100–1107.
13. J. A. Stork, F. T. Pokorny, and D. Kragic, "Integrated motion and clasp planning with virtual linking," in *IROS*, Tokyo, Japan, 2013.
14. ——, "A topology-based object representation for clasping, latching and hooking," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 138–145.
15. 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.
16. S. Makita and Y. Maeda, "3d multifingered caging: Basic formulation and planning," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2697–2702.

17. 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.
18. 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.
19. 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.
20. J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2013.
21. 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.
22. 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.
23. 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.
24. D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4304–4311.
25. I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.
26. 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.
27. 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.
28. L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.
29. 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.
30. M. Gualtieri, A. Ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 598–605.
31. E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4461–4468.
32. J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.
33. 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.
34. 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.
35. D. Kinga and J. Adam, "A method for stochastic optimization int," in *Conf. on Learning Representations (ICLR,)*, 2015.
36. S. Lembcke, "Chipmunk 2d physics engine," *Howling Moon Software*, 2013.
37. B. Dai and D. Wipf, "Diagnosing and enhancing vae models," *arXiv preprint arXiv:1903.05789*, 2019.
38. A. A. Taha and A. Hanbury, "An Efficient Algorithm for Calculating the Exact Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2153–2163, 2015.
39. R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.
40. G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.