A Two-stage Learning Architecture that Generates High-Quality Grasps for a Multi-Fingered Hand

Dominik Winkelbauer, Berthold Bäuml, Matthias Humt, Nils Thuerey, Rudolph Triebel

A Two-stage Learning Architecture that Generates High-Quality Grasps for a Multi-Fingered Hand

Dominik Winkelbauer[1,2], Berthold Bäuml[1,3], Matthias Humt[1], Nils Thuerey[2], Rudolph Triebel[1,2]

***Abstract*— We investigate the problem of planning stable grasps for object manipulations using an 18-DOF robotic hand with four fingers. The main challenge here is the high-dimensional search space, and we address this problem using a novel two-stage learning process. In the first stage, we train an autoregressive network called the hand-pose-generator, which learns to generate a distribution of valid 6D poses of the palm for a given volumetric object representation. In the second stage, we employ a network that regresses 12D finger joint configurations and a scalar grasp quality from given object representations and palm poses. To train our networks, we use synthetic training data generated by a novel grasp planning algorithm, which also proceeds stage-wise: first the palm pose, then the finger positions. Here, we devise a Bayesian Optimization scheme for the palm pose and a physics-based grasp pose metric to rate stable grasps. In experiments on the YCB benchmark data set, we show a grasp success rate of over 83%, as well as qualitative results grasping unknown objects on a real robot system.**

I. Introduction

Data-driven grasping is becoming more and more popular in recent years as it allows the grasping process to be fast and work with incomplete observations of the object [1]. Many of these grasping methods focus on parallel jaw grippers [2, 3, 4, 5]. While this leads to high success rates, it misses the potential of multi-finger hands which are able to perform more complex grasps.

Compared to parallel jaw grasping, generating grasps for multi-finger hands is more complicated due to the higher number of degrees of freedom (DOF). First, this makes it harder to generate grasps that can be used as training data for the neural network. While it is usually enough to perform random sampling for parallel jaw grasps, this is not possible for multi-finger grasps due to the large search space. Existing approaches therefore either use heuristics, human labels, or a dimensionality reduction of the search space to generate a grasp dataset. We propose instead to use the more sample-efficient Bayesian optimization, a convenient formulation of the search space and an objective function based on the ϵ-quality [6] which can be calculated rapidly together with a full parametrization of the search space. In this way, we are able to plan high-quality grasps in the full high-dimensional grasp space.

Another challenge in multi-finger grasping is the formulation of the grasping task such that it can be learned by a deep neural network. In our approach, we split up the learning task into generating multiple hand poses for a given object and then estimating a specific joint configuration for a given hand pose. In this way, we can still directly predict grasps for a given object without time-consuming sampling or optimization and at the same time make the generative task easier. Summarized, we make the following contributions in this work:

- We propose a novel grasp planner that is able to search the full high-dimensional grasp space of a multi-finger hand. It uses Bayesian Optimization to intelligently select sample locations based on previous samples and formulates the search space in an object-centric way.
- We improve the ϵ-quality in the following ways: We describe the grasp using a dense collection of contact points instead of using only a small number per link. We compute the maximum applicable force at every contact point based on their position relative to the joints and the joint's allowed torque intervals. Also, we describe an efficient way to compute the epsilon quality based on the Minkowski sum.
- We propose a novel formulation of the learning problem for generating more accurate grasps directly: we propose to split up the grasp generation task, s.t. only the hand pose is modeled via a generative network and a discriminative network is used to estimate a specific grasp configuration. Further, we propose to use an autoregressive network architecture similar to PixelCNN++ [8] for the generative network, as we show that this leads to more accurate hand poses compared to a conditional Wasserstein GAN (W-GAN) [9] or a conditional Variational Autoencoder (VAE) [10].

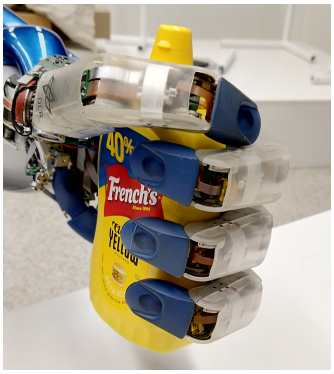

Fig. 1. A high-quality grasp planned in simulation and executed on the DLR-Hand II [7].

[1]DLR Institute of Robotics & Mechatronics, Germany
[2]Technical University of Munich (TUM), Germany
[3]Deggendorf Institute of Technology (DIT), Germany
Contact: Dominik.Winkelbauer@dlr.de
*This work is supported by the Helmholtz Association under the joint research school "Munich School for Data Science - MUDS".

II. Related Work

A. Analytical grasp planning

One way of finding stable multi-finger grasps for a given object is to define a grasp quality metric and then let an optimizer search for a grasp configuration maximizing that metric [11, 12]. As this requires exact knowledge of the physical and geometrical details, more recent approaches, including ours, use data-driven methods that can also work with incomplete knowledge [1]. However, analytical grasp planning is still used to generate training data for data-driven methods. To do so, the ϵ-quality metric [6] is often used. While the original definition assumes the same force at each contact, we show how the contact forces can be scaled realistically based on their location. To speed up the computation of the metric, the Partial Convex Hull algorithm only computes the part of the convex hull that is necessary for determining the weakest wrench [13]. We additionally manage to efficiently incorporate the more realistic Minkowski sum formulation of the metric into the algorithm. The ϵ-quality metric can be further extended to also take the object shape into account by only considering external wrenches that can be created through forces on the object surface [14]. Borst et al. [15] proposed to approximate that object wrench space via an ellipsoid which makes the computation of the metric more efficient. There has been much debate about how well analytical grasp metrics represent the stability of a grasp in reality, as they do not take any dynamics into account [16]. We show that with several extensions to the ϵ-quality metric, its ability to identify stable grasps can be improved while still being computationally efficient.

B. Data-driven grasp planning

Compared to analytical grasp planners, data-driven methods have the advantage that they can work with incomplete object models and are faster at test time.

1) Generative and discriminative approaches: In discriminative approaches, the network is trained to predict the success probability of a given grasp. While this makes the model easier to train, at test time a large number of grasps needs to be sampled using a prior and the optimization of the grasping parameters based on their predicted success probability needs to be performed [17, 18, 19, 20]. Therefore, it takes multiple seconds to generate one grasp.

Generative methods, like ours, do not have that problem, as they predict the grasping parameters directly. Some methods only predict one specific grasp for a given object [21, 22]. While this circumvents the need for a generative network architecture, it can lead to inference situations where the predicted grasp is not feasible due to obstacles or the kinematics of the robot. In our approach, we generate multiple grasps and can therefore choose one that is feasible. Other methods that go that way usually rely on a W-GAN or VAE architecture [23, 24], while we propose an autoregressive architecture which we show to generate more accurate grasps. Tobin et al. [25] are also using an autoregressive architecture, however, they are applying it to parallel jaw grasping and discretize the output space.

2) Grasp formulation: Data-driven methods also differ in the way the predicted grasp configuration is formulated. Some methods like Shao et al. [26] let the network predict contact points, usually one per finger. This has the disadvantage that a costly inverse kinematic solver needs to be run at test time to infer the position of hand base and fingers. To reduce the dimensionality, Patzelt et al. [24], Lundell et al. [23] only predict a grasp class or only the finger spread next to the 6D hand pose. While this reduces the complexity of the task, it requires labeling each training sample with the respective class and reduces the space of grasps the network can express. Our network architecture predicts instead the whole grasp configuration including all degrees of freedom. This is similar to Liu et al. [22], however, as noted above, they only predict one grasp for a given object while we model the whole distribution of grasps.

3) Training data generation: All supervised data-driven methods need training data, consisting of stable grasps planned on training objects. One way of creating such a dataset is by manually labelling the grasps [27] or using human demonstrations [28, 29]. This makes the data generation very time-consuming and it is not completely clear how to map a grasp from a human to a robotic hand. In recent years, using a simulator to label the stability of sampled grasps has become more prominent [17, 18, 20, 30, 24]. While this takes hand-object interactions into account and considers the grasp controller used on the robot, running a realistic simulator is very time consuming and therefore the number of grasps that can be tested is low. Alternatively, one can use analytical metrics to label grasps [21, 26, 22, 23], which is usually done via GraspIt! [31]. While GraspIt only makes use of the basic ϵ-quality, we improve the metric through multiple extensions which we show brings it closer to the evaluation performance with a rigid body simulator. Furthermore, we propose a new grasp planner which is able to search through the entire search space and is able to find a diverse set of grasps spread across different approach directions.

4) Learning to grasp via Reinforcement Learning: A completely different way of learning how to grasp is by using reinforcement learning [32]. Here, the network is trained in simulation to control the movement of the hand and joints over multiple timesteps. While these approaches have the advantage, that no explicit training dataset is necessary, the conditions on the real robot need to be well calibrated to match the ones of the simulator. Furthermore, the training of the RL agent might get stuck in a local minimum, such that the most stable grasps are never discovered.

III. Grasp Learning

To perform grasping efficiently at test time, a generative model is trained on directly estimating a set of stable grasps $\{x_1, ..., x_n\}$ for the robotic hand DLR-Hand II [7], shown in Fig. 3. Here, a grasp x_i is defined as a tuple consisting of a 6D hand pose h_i and a 12D joint configuration j_i.

We propose a two-stage approach (see Fig. 2): First, a generative network that suggests multiple promising hand poses h for a given object and then a discriminative model

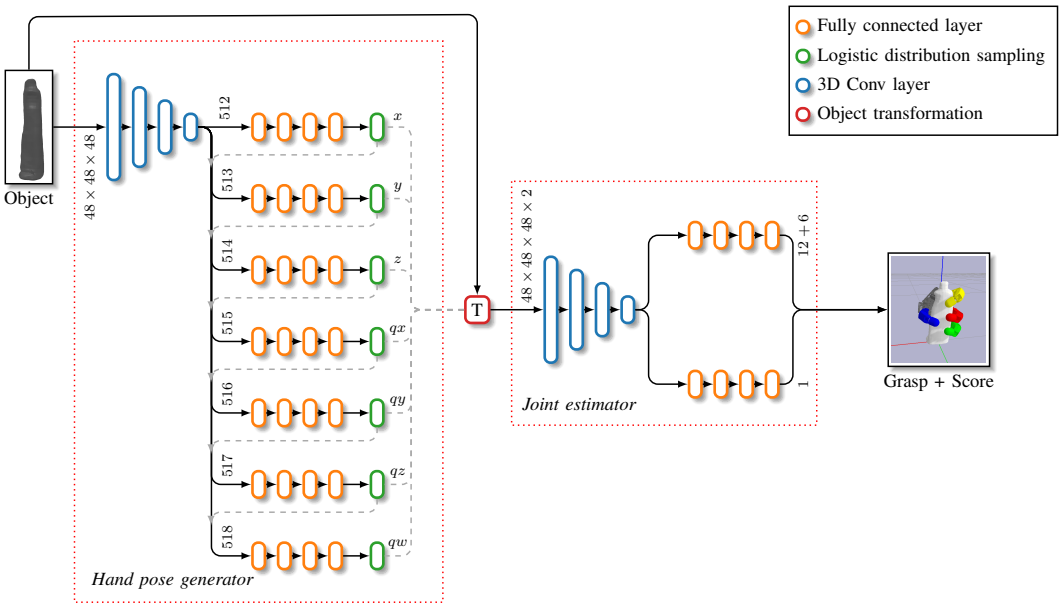

Fig. 2. Overview of our two-stage network architecture which we use to learn the distribution of stable grasps conditioned on a given object. Dashed lines represent connections that are only active during test time. Each of the two stages is trained separately.

that estimates a specific joint configuration j for each of these hand poses. Each of the stages is trained separately and they are only connected at test time. We found splitting the task up into two stages is crucial, as predicting hand pose and joint configurations jointly with a single generative network is harder to learn.

A. Object encoder

Both networks receive the observed 3D object as an input, which we represent as an signed distance field (SDF) of size 48^3, i.e., a voxel grid with containing signed distance values. For encoding the given voxel grid, both networks make use of the same encoding architecture consisting of four 3D convolutional layers, each followed by a pooling layer. The resulting feature map is flattened into a vector of size 512 and used further as the feature encoding of the given object.

B. Generative network

The generative network $\mathcal{G}$ is trained on the mapping from an observation o to a distribution over 6D hand poses h leading to stable grasps:

$$\mathcal{G} : o \rightarrow p(h|o) \tag{1}$$

For representing the rotational component of the hand pose, we make use of quaternions. Therefore, each predicted hand pose is defined by seven parameters. We propose an autoregressive architecture inspired by PixelCNN++ [8] which we found to predict more accurate hand poses compared to other architectures. To do so, we decompose the distribution into multiple single-dimensional distributions, each conditioned on the previous ones.

$$p(h|o) = \prod_{i=1}^{7} p(h_i|h_1, ..., h_{i-1}, o) \tag{2}$$

Next to the object encoder, the generative network consists of seven blocks of fully connected layers each modeling one of these conditional distributions. The blocks are connected in an autoregressive way, meaning each block receives the outputs of the previous blocks. As proposed by PixelCNN++ [8], we approximate each conditional distribution with a mixture of logistic distributions whose parameters are estimated via the respective network block. Each of the L logistic distributions is described with the parameters π_l, μ_l and s_l. Next to the encoding of the given object, each block also receives the values of the conditioned dimensions. During training, these values come directly from the ground truth label, while during inference, the values are sampled from the probability distributions described by the previous network blocks The formulation of the loss term used to train the network is derived from maximizing the log-likelihood (see Salimans et al. [8]).

C. Joint estimator network

The joint estimator network $\mathcal{J}$ models the mapping from observation o and hand pose h to a specific joint configuration j, a hand pose refinement Δh and score s:

$$\mathcal{J} : \{o, h\} \rightarrow \{j, \Delta h, s\} \tag{3}$$

To fuse the two input modalities and to reduce the complexity of the task, we bring the object observation into the local coordinate system of the hand. To still be able to consider the table as an obstacle, we add an SDF encoding of the table into the second channel of the input. Two heads

are applied to the object feature encoding, one estimating the 12D joint configuration plus a small 6D hand pose correction and one estimating the score of the resulting grasp. The 6D correction consists of three translational and three euler angle dimensions which are added to the given hand pose h. This allows the joint estimator to slightly adjust the pose predictions of the first stage. Each of the two heads consists of four fully connected layers using ReLU activation functions.

The network is trained using one L2 loss per head. The mapping learned by this stage is ambiguous, in the way that different joint configurations can lead to equally stable grasps for a given hand pose. Therefore, simply applying an L2 loss using one specific ground truth sample can lead to learning the, often undesired, mean of the underlying distribution of valid joint configurations. To reduce this effect, we collect for each training sample i a set of stable joint configurations $\{\widehat{j}_{i,1}, ..., \widehat{j}_{i,K_i}\}$ and then use the one closest to the network's prediction as the ground truth label in the loss:

$$L_{\text{joint}} = \frac{1}{N} \sum_{i=1}^{N} w_i \min_k (\widehat{j}_{i,k} - j_i)^2 \tag{4}$$

Similar to Sundermeyer et al. [2], the joint configuration loss is weighted by w_i which ensures that the training focuses on good grasps. We apply here a binary weighting, where we set w_i only to 1 if its part of the top 50 grasps found on the respective training object. The grasp quality estimation loss is computed equally for all training samples, such that the network is still forced to learn the difference between stable and unstable grasps:

$$L_{\text{score}} = \frac{1}{N} \sum_{i=1}^{N} (\widehat{s}_i - s_i)^2 \tag{5}$$

As a target $\widehat{s}_i$ we use the score $\widehat{s}_{i,k}$ of the grasp that the networks joint prediction j_i was closest to. The whole network is trained using the combination of both losses:

$$L_{\text{discr}} = L_{\text{joint}} + \alpha L_{\text{score}} \tag{6}$$

To balance both losses we use $\alpha = 0.1$.

IV. Grasp planner

In this section, we describe our grasp planner which we use to generate a dataset of grasps that are necessary to train the grasping network from Section III.

To find stable grasps for a given object, we follow the usual way of formulating the grasping task as an optimization problem. As the search space for a multi-finger hand is quite large, we decided on an efficient analytical grasp metric in our objective function. In that way, we do not need to restrict the search space, e.g. by using Eigengrasps [33], but can consider all possible grasp configurations. As the surface of the objective function is not smooth due to edges in the object and hand geometry and the numerical computation of gradients is computationally costly, we are not using a gradient-based optimizer. Instead, we use Bayesian optimization which is able to sample new points based on previous ones and can handle non-smooth and multi-modal objective functions. Unfortunately, Bayesian optimization scales poorly with the number of samples and dimensions of the search space. That is why we split up the optimization problem into two nested ones: The Bayesian optimizer optimizes the 6D hand pose and an evolutionary optimizer is used to find the best joint configuration for a given hand pose. Both parts are explained in more detail in the following subsections.

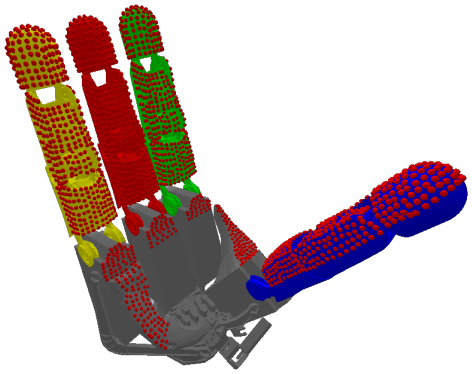

Fig. 3. The DLR-Hand II [7] which we use for grasping annotated. Each red dot represents a potential contact point.

A. Handpose optimization

The outer optimizer searches through the space of all hand poses around the object.

1) Search Space: Just defining the search space as raw 6D poses in the area around the object, would lead to many hand poses where the hand is actually pointing away from the object. To prevent that, we define the search space as the 6D pose of the objects inside the palm of the fixed hand. If, after sampling a 6D pose, the object collides with the open hand, we move it along the up axis until the collision is resolved. This makes it easier to sample power grasps where the hand palm is in direct contact with the object.

2) Objective function: The value of the objective function for a sampled hand pose is defined as the quality of the best grasp using that hand pose. To find the joint configuration of that grasp, the inner optimizer, described in Section IV-B is used.

3) Optimizer: For optimizing the 6D hand pose, we make use of Bayesian Optimization. For performance reasons we do not use a Gaussian process as the surrogate function but instead use tree-structured Parzen estimators [34].

B. Joint optimizer

The joint optimizer is run for each hand pose sampled by the hand pose optimizer and its goal is to find a distribution of stable joint configurations based on the given hand pose.

1) Search space: The search space is defined as the full 12D joint configuration space defined by the four fingers of DLR Hand II. Specifically, each finger is described by one actuator controlling the proximal joint, one controlling the knuckle joint and one controlling the middle and distal joint simultaneously.

2) Objective function: Starting from a sampled joint configuration, we now map each finger to its closest object surface. This simplifies the optimization problem a lot, as now the optimizer does not need to carefully place the finger on the surface by itself. If none of the fingers collides with the object, the table or any other finger, the grasp is rated using the ϵ-grasp metric described in the next subsections.

a) Basic ϵ-quality: To compute the quality of a given grasp, at first, contact points between hand and object are determined: to do so, we check for all 2500 potential contact points (see Fig. 3) across the hand palm and fingers, if they are closer than 5 mm to the object's surface and if the angle between object and flipped hand normal at the respective point is smaller than 30°.

After collecting for each finger $\mathcal{F}$ all contact points $p_1, ..., p_{N_\mathcal{F}}$ together with the object normals $n_1, ..., n_{N_\mathcal{F}}$ at each contact point, the forces that can be applied at each point are determined. These forces are described by the friction cone which is approximated via a pyramid having $K = 8$ edges. Based on that, the space of wrenches, i.e. forces and torques, that can be exerted on the object via contact point p_i is defined as the convex hull around the wrenches that are caused by the forces f_i^k along the edges of that pyramid. Each of these wrenches w_i^k are calculated via:

$$w_i^k = \begin{pmatrix} s_i^k f_i^k \\ p_i \times (s_i^k f_i^k) \end{pmatrix}, \quad \text{with scaling factors } s_i^k. \tag{7}$$

b) Realistic contact force scaling: In (7), we introduced a scaling s_i^k to every force vector instead of assuming the same unary length force at each contact point. The scale is determined based on the maximum force that can be applied on that point p_i on the link and into the direction of the force vector f_i^k based on the torque limits $\tau^{\max}$ of the finger.

To relate a scaled force $s_i^k f_i^k$ at a point p_i with the torques τ_i^k applied at the joints, the Jacobian J_i can be used.

$$\tau_i^k = J_i^T (s_i^k f_i^k) \tag{8}$$

Based on that, we determine the highest scaling s_i^k possible such that all torques stay within their limits.

So far only one contact point was taken into account. The torques produced by forces applied at multiple contact points on one finger can be written in a similar manner as:

$$\tau = \sum_{i=1}^{N_\mathcal{F}} \sum_{k=1}^{K} J_i^T (s_i^k f_i^k) \tag{9}$$

Now assuming that every contact force makes use of every joint and creates torques with the same direction, the torque budges $\tau^{\max}$ can be distributed via the weights β_i^k as

$$\tau^{\max} \geq \sum_{i=1}^{N_\mathcal{F}} \sum_{k=1}^{K} \beta_i^k J_i^T (s_i^k f_i^k), \tag{10}$$

where $\sum_i \sum_k \beta_i^k \leq 1$. The space of all wrenches $\mathrm{GWS}_\mathcal{F}$ that one finger $\mathcal{F}$ can exert on the object while staying in its torque limits can be expressed as the convex hull over all wrenches originating from the fully scaled force vectors at each contact point.

$$\mathrm{GWS}_\mathcal{F} = \mathrm{CH}(\bigcup_{i=1}^{N_\mathcal{F}} \bigcup_{k=1}^{K} w_i^k) \tag{11}$$

As each finger can apply wrenches to the object independent from each other, the grasp wrench space of the full grasp can be expressed via the Minkowski sum $\oplus$:

$$\mathrm{GWS} = \mathrm{CH}(\oplus_{\mathcal{F}=1}^{4} \mathrm{GWS}_\mathcal{F}) \tag{12}$$

Based on the space of all wrenches that the grasp can exert on the object, the ϵ-quality metric now is defined as the smallest distance from the origin to the boundary of that space [6].

c) Object wrench space: We further make use of the object wrench space (OWS) extension to the quality metric, first proposed by Pollard [14]. This has the advantages that the metric gets independent of the selection of the reference point used for calculating the torques and is independent of scaling the object. We use the efficient implementation proposed by Borst et al. [15].

d) Computing the metric including Minkowski sum: We make use of the partial convex hull algorithm (PQH) to efficiently determine the smallest distance from the origin to the convex hull [13]. Here, the convex hull is built using the qhull algorithm, which, starting from an initial guess, sequentially adds points to the convex hull until no points are left outside. With the PQH that procedure stops early when the facet closest to the origin has already been fully extended.

As defined in (12), the computation of the points defining the convex hull includes the Minkowski sum. Computing that Minkowski sum explicitly for a usual grasp having 20 contact points per finger is intractable, as it would result in $(20 \times 8)^4 = 6.55 \times 10^8$ points in the wrench space.

We propose a novel extension to the PQH algorithm that is able to implicitly consider the Minkowski sum with little overhead: At each iteration of the PQH algorithm, at first the facet closest to the origin is selected and then the point that is furthest away from that facet is added to the convex hull. We adjust the latter point selection in the following way: Given a set of points $\{p_1^\mathcal{F}, ..., p_{N_\mathcal{F}}^\mathcal{F}\}$ per finger $\mathcal{F}$, find the combination of points $k_{\text{ft}} \in [1, N_\mathcal{F}]^4$ that is furthest away from the facet:

$$k_{\text{ft}} = \arg\max_k d \left(\sum_{\mathcal{F}=1}^{4} p_{k_\mathcal{F}}^\mathcal{F} \right), \tag{13}$$

with $d(p)$ being the distance of a point to the facet with normal n and offset o: $d(p) = n \cdot p + o$. By reformulating (13), this computationally expensive combinatorial problem can be drastically simplified:

$$\begin{aligned} k_{\text{ft}} &= \arg\max_k n \cdot \left(\sum_{\mathcal{F}=1}^{4} p^{\mathcal{F}}_{k_{\mathcal{F}}} \right) + o \\ &= \arg\max_k \sum_{\mathcal{F}=1}^{4} n \cdot p^{\mathcal{F}}_{k_{\mathcal{F}}} = \sum_{\mathcal{F}=1}^{4} \arg\max_{k_{\mathcal{F}}} n \cdot p^{\mathcal{F}}_{k_{\mathcal{F}}} \end{aligned} \tag{14}$$

So instead of finding the combination of points that is furthest away, we can simplify the problem to finding one point per finger that is furthest away from the facet. Therefore, it is enough to compute the distance of every point to the facet once per iteration, just like in the original PQH algorithm. The only additional overhead is that the algorithm runs usually for more iterations, as implicitly more points are available.

3) Optimizer: For finding stable joint configurations for a given hand pose, we make use of an evolutionary optimizer. Each run of the joint optimizer consists of 144 sampling steps.

V. Experiments

A. Experimental setup

We use the grasp planner, described in Section IV, to generate a training dataset. As training objects we make use of 2100 objects, up to 50 per category, from the ShapeNet dataset [35]. From all found grasps, we use the top 50 grasps and 50 other random grasps per object to form our training dataset. For training the generative network we only use the top 50 grasps. The generative network is trained for 3×10^5 iterations using Adam optimizer with a learning rate of 10^{-4} and a batch size of 64. The joint estimator network is trained for 10^5 iterations, using the same optimizer parameters. To generate the voxel grids containing signed distance values, we build on the algorithm proposed by Denninger and Triebel [36], which can handle meshes that are not completely watertight. We normalize all inputs and labels to have a mean of 0 and a standard deviation of 1.

B. Grasp metric comparison

To show that our grasp metric can be used as an indicator for the success of the grasp execution, we compare it with the success rate of grasping and lifting the object in a physics based simulator. Furthermore, we also compare how the different extensions to the original ϵ-quality [6] influence these results. To simulate a given grasp, we start with the hand placed 30 cm moved back along the approach direction and then approach the object with opened up fingers. During the last two centimeters of the approach, the fingers are closing toward the joint configuration specified by the given grasp. A joint-level impedance controller is now used to control the grasp. After the object is grasped, the hand lifts the object upwards. If at any point during the simulation, the object drops out of the hand, the grasp is marked as not successful. In this way, we label a set of 22,000 grasps across 225 objects from our ShapeNet-based training dataset. For each grasp metric, we now calculate the ratio of successful grasps in the set of grasps with the top 1% and top 10% grasp metric values.

TABLE I
Comparison of ϵ-Metric Extensions

	Success ratio	
	Top 1%	Top 10 %
Original ϵ-quality [6]	61.4%	66.1%
+ force scaling	84.4%	69.4%
+ object wrench space [15]	90.2%	79.3%
+ minkowski sum	90.6%	81.2%

Table I shows the results of this evaluation. It can be seen that the original ϵ-quality only poorly correlates with the success of a grasp in simulation. This coincides with the findings of similar evaluations in other works [16]. However, especially adding the more realistic force scaling and the object wrench space lead to a major increase in the ratio of successful grasps. Adding the Minkowski sum, which allows each finger to exert forces independently, only causes a minor improvement. This is plausible, as the cases where this becomes relevant are usually rare.

C. Evaluation in simulation

Using the simulation-based evaluation strategy described in the previous subsection, we now rate the stability of grasps predicted by our trained neural network. Here we now use 16 objects from the YCB dataset [37], each is evaluated three times with different random rotations around the up-axis. For each object, we let the network sample 1024 grasps and then perform the grasp that was annotated with the highest score by the network. The ratio of grasps that could lift the object successfully is shown in Table II.

Fig. 4 shows some of these grasps on six YCB objects covering various shapes. It has to be noted, that the visualized grasps represent the direct output of the neural network, so no post-processing was applied. Based on that, it can be seen that the network manages to place the fingers quite accurately on the object's surface. Furthermore, the network seems to be able to adjust its grasps based on the object's shape: for smaller objects, like the baseball or the banana, the network predicts a precision grasp, while for bigger objects like the pitcher or the bleach bottle, it outputs power grasps where the hand palm is in contact with the object.

D. Ablation study

Table II further shows how the components we proposed for the network architecture influence the results.

First, we look at the influence of transforming the input object of the joint estimator into the local coordinate system of the hand pose. Therefore, we train a joint estimator which gets the object in its original pose and the hand pose as an additional input. As can be seen in the results, this makes the learning problem a lot harder and results in less stable grasps. Further, we train an autoregressive generative architecture on

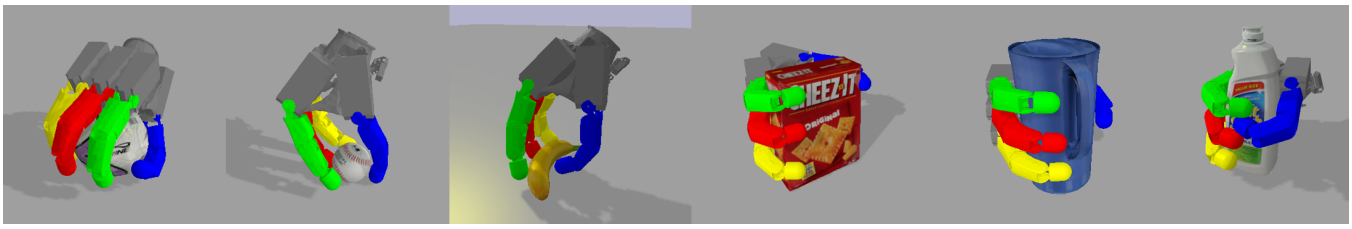

Fig. 4. Grasps predicted by our neural network across multiple YCB objects. Each grasp is the one that received the highest predicted score on the respective object.

predicting not only the hand pose but the full grasp. The generative learning task becomes now more difficult to learn which results in a doubling of the ratio of failures. Finally, we confirm that using a random grasp sampled by the network leads to worse results compared to using the one that was scored highest by the network.

TABLE II
EVALUATION OF THE GRASPING NETWORK ON YCB

	Success ratio
Our proposed network architecture	83.3%
No trafo of the object input	54.2%
One generative network on full grasps	66.7%
Random grasp selection (no scoring)	68.7%

E. Evaluation of the generative stage

To compare our proposed autoregressive architecture with the often used W-GAN and VAE architectures, an isolated evaluation is conducted. For each YCB object, we sample 16 hand poses using each of the evaluated generative models. Afterwards, the evolutionary optimizer described in Section IV-B is used to find the best joint configuration for each sampled hand pose. The final grasps are rated based on their execution in simulation and based on their grasp quality. The results shown in Table III confirm that the autoregressive architecture allows sampling more accurate hand poses leading to better grasp qualities and higher success rates.

This also becomes apparent when looking at the qualitative example shown in Fig. 5. Here, hand poses sampled on the YCB chips object can are visualized from a top down perspective. While W-GAN and VAE produce hand poses intersecting with the object or being several centimeters away, the autoregressive architecture persistently produces accurate poses. We suspect that this behavior comes from the fact that the autoregressive network directly predicts the specific distribution, while the other two networks need to learn a mapping from every latent vector to a valid hand pose.

To check the influence of the parameter order when using the autoregressive architecture, we evaluate different permutations with the same procedure. The results in Table III show that besides small perturbations all evaluated permutations lead to similar results.

TABLE III
ISOLATED EVALUATION OF THE GENERATIVE STAGE

	Mean grasp quality	Success ratio
Autoregressive	8.59	68.2%
VAE	5.64	56.3%
W-GAN	4.92	50.0%
Autoregressive with different parameter order:		
$q_w q_z q_y q_x t_z t_y t_x$	8.71	69.1%
$q_x q_y q_z q_w t_x t_y t_z$	8.34	68.1%
$q_x t_x q_y t_y q_z t_z q_w$	8.52	68.7%

F. Sim-to-real transfer

To evaluate how well the predicted grasps generalize to the real world, we executed some of the grasps on the real robot. We consider two scenarios: First, we assume the 3D model of the object is given and the object pose is determined via the iterative closest point algorithm [38]. In that case, the input to the neural network is calculated in the same way as in the simulation-based evaluation. In the second scenario, we assume an unknown object which we only observe via a fused 3D model from the robot's depth sensor [39]. Here, we perform first a shape completion step to fill up the incomplete 3D model. We show both cases in the video accompanying this paper, executed on DLR's humanoid robot Agile Justin [40], using a learning-enhanced motion planner [41] for moving the hand to the pre-grasp

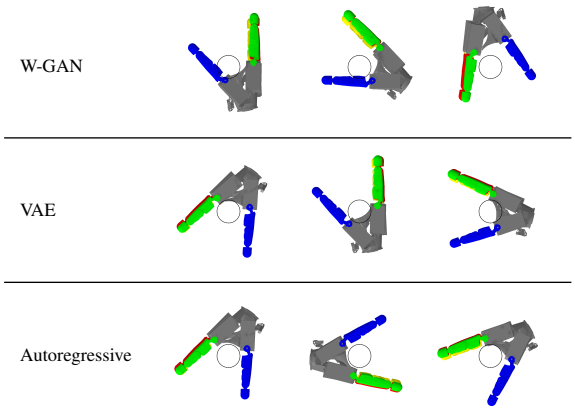

Fig. 5. Hand poses sampled by different generative models for the YCB chips object can. The object is seen from the top and its boundary is visualized by a black circle. This example visualizes the precision of the autoregressive generator compared to a W-GAN and a VAE.

pose without collisions.

VI. Conclusions

We presented a novel approach for efficiently finding high quality grasps for a multi-finger hand on a given object using Bayesian optimization coupled with evolutionary optimization. Our objective function is based on the analytical ϵ-metric which makes it efficient to compute and lets us formulate the search space with full DOF. To make the grasp metric more realistic, we propose a procedure to scale the contact forces based on their location instead of assuming the same force at every contact point. Furthermore, we propose an efficient way for computing the Minkowski sum-based ϵ-metric. Using the training data generated by the grasp planner, we train a neural network on the task of generating grasps for a given object. We propose a two-stage approach, in which first suitable hand poses are generated and then specific stable joint configurations per hand pose are regressed. This makes the generative task easier, while still being accurate. Our generative network consists of an autoregressive architecture predicting logistic distributions which we found to outperform the VAE and W-GAN architecture.

References

[1] J. Bohg *et al.*, "Data-driven grasp synthesis-A survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, sep 2014.

[2] M. Sundermeyer *et al.*, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *IEEE International Conference on Robotics and Automation*, 2021.

[3] A. Mousavian, C. Eppner, and D. Fox, "6-DOF graspnet: variational grasp generation for object manipulation," in *arXiv*, 2019.

[4] H.-S. Fang *et al.*, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.

[5] J. Mahler *et al.*, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[6] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3. Publ by IEEE, 1992, pp. 2290–2295.

[7] J. Butterfaß *et al.*, "DLR-Hand II: Next generation of a dextrous robot hand," in *Proc. IEEE International Conference on Robotics and Automation*, 2001, pp. 109–114.

[8] T. Salimans *et al.*, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[9] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[11] A. A. T. Miller *et al.*, "Automatic grasp planning using shape primitives," *Robotics and Automation*, vol. 2, pp. 1824–1829, 2003.

[12] A. Okamura, N. Smaby, and M. Cutkosky, "An overview of dexterous manipulation," in *IEEE International Conference on Robotics and Automation*, 2000.

[13] S. Liu and S. Carpin, "Partial convex hull algorithms for efficient grasp quality evaluation," in *Robotics and Autonomous Systems*, vol. 86, 2016, pp. 57–69.

[14] N. S. Pollard, "Parallel Methods for Synthesizing Whole-Hand Grasps from Generalized Prototypes," *Technology*, no. 1989, 1994.

[15] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2004.

[16] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *IEEE International Conference on Robotics and Automation*, 2015.

[17] Z. Yilun and H. Kris, "6DOF Grasp Planning by Optimizing a Deep Learning Scoring Function," *RSS Workshop on Revisiting Contact - Turning a Problem into a Solution*, 2017.

[18] U. R. Aktas *et al.*, "Deep dexterous grasping of novel objects from a single view," 2019.

[19] Q. Lu *et al.*, "Multi-Fingered Grasp Planning via Inference in Deep Neural Networks," 2020.

[20] M. Van Der Merwe *et al.*, "Learning Continuous 3D Reconstructions for Geometrically Aware Grasping," in *IEEE International Conference on Robotics and Automation*, 2020.

[21] P. Schmidt *et al.*, "Grasping of Unknown Objects Using Deep Convolutional Neural Networks Based on Depth Images," in *IEEE International Conference on Robotics and Automation*, 2018.

[22] M. Liu *et al.*, "Deep differentiable grasp planner for high-DOF grippers," 2020.

[23] J. Lundell, F. Verdoja, and V. Kyrki, "DDGC: Generative Deep Dexterous Grasping in Clutter," in *IEEE Robotics and Automation Letters*, vol. 6, no. 4, 2021, pp. 6899–6906.

[24] F. Patzelt, R. Haschke, and H. Ritter, "Conditional StyleGAN for Grasp Generation," no. Icra, pp. 4481–4487, 2021.

[25] J. Tobin *et al.*, "Domain Randomization and Generative Models for Robotic Grasping," in *IEEE International Conference on Intelligent Robots and Systems*, oct 2018, pp. 3482–3489.

[26] L. Shao *et al.*, "UniGrasp: Learning a Unified Model to Grasp with Multifingered Robotic Hands," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020, pp. 2286–2293.

[27] J. Lundell *et al.*, "Multi-FinGAN: Generative Coarse-To-Fine Sampling of Multi-Finger Grasps," 2020.

[28] M. Kopicki *et al.*, "One-shot learning and generation of dexterous grasps for novel objects," *International Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.

[29] S. Brahmbhatt *et al.*, "ContactGrasp: Functional Multi-finger Grasp Synthesis from Contact," in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 2386–2393.

[30] Q. Lu, M. van der Merwe, and T. Hermans, "Multi-Fingered Active Grasp Learning," 2020.

[31] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[32] M. Q. Mohammed, K. L. Chung, and C. S. Chyi, "Review of Deep Reinforcement Learning-Based Object Grasping: Techniques, Open Challenges, and Recommendations," *IEEE Access*, vol. 8, pp. 178 450–178 481, 2020.

[33] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dimensionality reduction for hand-independent dexterous robotic grasping," in *IEEE International Conference on Intelligent Robots and Systems*, 2007, pp. 3270–3275.

[34] J. Bergstra *et al.*, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[35] A. X. Chang *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[36] M. Denninger and R. Triebel, "3d scene reconstruction from a single viewport," in *European Conference on Computer Vision*. Springer, 2020, pp. 51–67.

[37] B. Calli *et al.*, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[38] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[39] R. Wagner, U. Frese, and B. Bäuml, "Real-time dense multi-scale workspace modeling on a humanoid robot," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2013.

[40] B. Bäuml *et al.*, "Agile justin: An upgraded member of DLR's family of lightweight and torque controlled humanoids," in *Proc. IEEE International Conference on Robotics and Automation*, 2014.

[41] J. Tenhumberg and B. Bäuml, "Speeding up optimization-based motion planning through deep learning," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2022.