UniGrasp: Learning a Unified Model to Grasp with N-Fingered Robotic Hands

Lin Shao\textsuperscript{1}, Fabio Ferreira\textsuperscript{1,2}, Mikael Jorda\textsuperscript{1}, Varun Nambiar\textsuperscript{1}, Jianlan Luo\textsuperscript{3}, Eugen Solowjow\textsuperscript{4}, Juan Aparicio Ojea\textsuperscript{4}, Oussama Khatib\textsuperscript{1}, Jeannette Bohg\textsuperscript{1}

Abstract—To achieve a successful grasp, gripper attributes including geometry and kinematics play a role equally important to the target object geometry. The majority of previous work has focused on developing grasp methods that generalize over novel object geometry but are specific to a certain robot hand. We propose UniGrasp, an efficient data-driven grasp synthesis method that considers both the object geometry and gripper attributes as inputs. UniGrasp is based on a novel deep neural network architecture that selects sets of contact points from the input point cloud of the object. The proposed model is trained on a large dataset to produce contact points that are in force closure and reachable by the robot hand. By using contact points as output, we can transfer between a diverse set of N-fingered robotic hands. Our model produces over 90% valid contact points in Top10 predictions in simulation and more than 90% successful grasps in the real world experiments for various known two-fingered and three-fingered grippers. Our model also achieves 93% and 83% successful grasps in the real world experiments for a novel two-fingered and five-fingered anthropomorphic robotic hand, respectively.

I. INTRODUCTION

The ability of a robot to grasp a variety of objects is of tremendous importance for many application domains such as manufacturing or warehouse logistics. It remains a challenging problem to find suitable grasps for arbitrary objects and grippers; especially when objects are only partially observed through noisy sensors.

The most successful approaches in recent literature are data-driven methods where large-capacity models are trained on labeled training data to predict suitable grasp poses, e.g. \cite{1,2,3}. The main objective of these types of approaches is to generalize over novel objects that were not part of the training data. All of these models are trained for one particular end-effector, which is typically a two-finger gripper. There are only few exceptions that consider more dexterous hands e.g. \cite{4,10,11}. Of those, \cite{10,12,13} go beyond a small set of pre-grasp shapes. \cite{7} is the only approach we are aware of that considers two different grippers in one model but still trains separate grasp synthesis models for each one.

In this paper, we go beyond generalization to novel objects. We propose UniGrasp, a model that additionally generalizes to novel gripper kinematics and geometries with more than two fingers (see Fig. 1). Such a model enables a robot to cope with multiple, interchangeable or updated grippers without requiring retraining. UniGrasp takes as input a point cloud of the object and a robot hand specification in the Unified Robot Description Format (URDF) \cite{15}. It outputs a set of contact points of the same number as there are fingers on the gripper. The model is trained to select contact points that are in force closure and reachable by the robot hand. This contact-based grasp representation allows to infer precision grasps that exploit the dexterity of the multi-fingered hand. It also alleviates the need to define a finite set of grasp pre-shapes and approach directions, e.g. from the top.

This work makes the following contributions: (1) Given a point cloud from an object and a URDF model of a gripper, we map gripper and object features into separate

---

1 Stanford Artificial Intelligence Lab (SAIL), Stanford University, CA, USA. \{lins2, fabiof, mjorda, vnambiar, khatib, bohg\}@stanford.edu
2 Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany
3 Dep. of ME and Dep. of EECS, University of California, Berkeley, CA, USA. jianlanluo@cs.berkeley.edu
4 Siemens Corporate Technology, Berkeley, CA, USA. \{eugen.solowjow, juan.aparicio\}@siemens.com
*The three authors contributed equally to this work. This work has been partially supported by JD.com American Technologies Corporation (JD) under the SAIL-JD AI Research Initiative and by the International Center for Advanced Communication Technologies (InterACT). This article solely reflects the opinions and conclusions of its authors and not JD or any entity associated with JD.com.
lower-dimensional latent spaces. The resulting features are concatenated and form the input to the part of the model that generates sets of contact points. (2) We propose a novel, multi-stage model that selects sets of contact points from an object point cloud. These contact points are in force closure and reachable by the gripper to produce a 6D grasp. (3) In simulation and the real world, we show that both, our mapping scheme and multi-stage model adapts well to a varying number of fingers. The selected contact point sets are typically clustered in a local neighborhood and are therefore well suited for execution on a real robot for which it may be hard to precisely align its fingers with an isolated point. (4) Finally, we publish a new large-scale dataset comprised of tuples of annotated objects with contact points for various grippers, covering many popular commercial grippers.

II. RELATED WORK

In the following, we review related work that shares some of the key assumptions with our work, e.g. the object to be grasped is novel and only observed through a noisy depth camera. For a broader review of the field on data-driven grasp synthesis, we refer to [1]. Furthermore, we briefly discuss learning-based methods for processing point clouds.

A. Data-Driven Grasp Synthesis for Novel Objects

Learning to grasp novel objects from visual data has been an active field of research since the seminal paper by Saxena et al. [16]. Since then, many approaches have been developed that use more sophisticated function approximators (CNNs) to map from input data to a good grasp [5, 11, 14, 17, 18]. These approaches differ in the input data format (e.g. 2D images, depth maps or point clouds), the grasp representation (e.g.grasping points, oriented 2D rectangles, a 2D position and wrist orientation or a full 6D pose) and whether they learn to grasp from real world examples or synthetic data.

In our work, we take as input a point cloud of a segmented object and generate a set of contact points, one per finger of the hand. We train our model on synthetically generated object and generate a set of contact points, one per finger basically defining such a precision grasp. Through inverse kinematics, it yields an end-effector pose in $SE(3)$ and a finger configuration that exploits the full dexterity of the hand. Of the aforementioned related works, [10, 12, 13] use a multi-fingered robot hand and go beyond a small set of pre-grasp shapes. [10] suggest to use a CNN that produces heatmaps which indicate suitable locations for fingertip and palm contacts. These contacts are used as a seed for a grasp planner to determine the final grasp pose for the reconstructed object. In follow-up work [19], a CNN completes object shapes from partial point cloud data to provide the grasp planner with a more realistic object model. [12] propose to train a deep conditional variational auto-encoder to predict the contact locations and normals for multi-fingered grasps given an RGB-D image of an object. [13] propose a method that directly optimizes grasp configurations of a multi-fingered hand by backpropagating through the network with respect to these parameters. All these methods are work for a specific hand while our method not only generalizes over novel objects but also over novel robotic grippers.

B. Representation Learning for Point Clouds

Recently, various approaches for processing sparse point clouds have been proposed in related work. [20] propose PointNet to learn object features for classification and segmentation. PointNet++ [21] applies PointNet hierarchically for better capturing local structures. In our approach, we adopt PointNet [20] to extract features of robotic hands and PointNet++ to extract object features. [22] train an auto-encoder network to reconstruct point clouds. The learned representations enable shape editing via linear interpolation inside the embedding space. We train a similar auto-encoder network to learn a representation of various robotic hands.

III. TECHNICAL APPROACH

Given a point cloud of an object and a URDF of a gripper, we aim to select a set of contact points on the surface of the object such that these contact points satisfy the force closure condition and are reachable by the gripper without collisions as shown in Fig. 2.

If the gripper has $N$ fingers, our model $UniGrasp$ selects a set of $N$ contact points from the object point cloud. Due to the diversity in possible hand designs and object shapes, this is a challenging problem that goes beyond what has previously been considered. Related work focuses on generalizing over novel objects, not over novel grippers. In this section, we first discuss how we learn a uniform representation of $N$-fingered robotic hands. This representation can be constructed from a URDF file which describes the kinematics.
Then (to these as boundary configurations. Let \( M \) hand, e.g. that outline the boundaries of the configuration space of the geometry and kinematics as input to UniGrasp. A. Gripper Representation

We aim to find a compact representation of the gripper geometry and kinematics as input to UniGrasp. In this section, we first describe how we encode the geometry of robotic hands. Then we describe how we project a novel robot hand into the learned latent space and construct the input feature (see Fig. 2).

1) Unsupervised Learning of a Gripper Representation:

We use an autoencoder to learn a low-dimensional latent space that encodes the geometry of robotic hands in a specific joint configuration. For training, we synthesize 20k point clouds of 2000 procedurally generated grippers each in different joint configurations. Each point cloud consists of 2048 points. We use PointNet \(^2\) as encoder without the transformation modules. This yields a k-dimensional feature vector that forms the basis for the latent space. The decoder transforms the latent vector using 3 fully connected layers. The first two layers use a ReLU activation function. The last layer produces a 2048 \( \times \) 3 output, the reconstructed point cloud of the gripper. We use the Chamfer distance \(^2\) to measure the discrepancy between the input and reconstructed point cloud.

2) Representing a Novel Robotic Hand: The autoencoder described in the previous section allows us to encode a robot hand in a specific joint configuration. However, we also want to encode the gripper kinematics and range of joints. In the following, we describe how we parse the URDF \(^3\) file and construct a gripper feature for the example of a 3-fingered hand with 3 DoF.

Let us assume, each finger of the hand has one revolute joint. We denote the joints as \( \theta_1, \theta_2, \theta_3 \). Each joint has joint limits. Let \( L_i \) and \( H_i \) represent the minimum and maximum joint angle. There are \( 2^3 = 8 \) joint configurations that outline the boundaries of the configuration space of the hand, e.g. \((L_1, L_2, L_3)\) \((H_1, L_2, L_3)\) \((L_1, H_2, L_3)\). We refer to these as boundary configurations. Let \( M_i = 0.5H_i + 0.5L_i \). Then \((M_1, M_2, M_3)\) describes the joint configuration of the hand where each joint angle takes the mean value of joint limits. We refer to this as central configuration. We generate point clouds sampled on the surface of the gripper model under all boundary and the central configurations. For the above example of a 3DoF gripper, there are 9 point clouds in total. They represent the kinematic and geometry attribute of this specific robotic hand.

We feed the 9 point clouds into the autoencoder to extract features (see Fig. 3). Note that robotic hands with a different number of DoF have a different number of boundary configurations. To get a fixed-size feature, we apply three pooling operations (max-, min-, and mean-pooling) among the batch dimension. The output features of those three operations are concatenated to get a final robotic hand feature.

B. Contact Point Set Selection

We now have a feature describing an N-fingered robotic hand and a point cloud representing the object \( S_0 = \{p_i\}_{i=1}^n \) where \( L \) denotes the number of points. We aim to select \( N \) points from this point cloud such that these \( N \) points form force-closure and are reachable by the robotic hand without collisions. The task of training a neural network for point set selection is new and challenging.

Fig. 2 shows the process in which \( \text{PSSN} \) selects \( N \) points in a multi-stage way. In Stage One, our model first selects one point \( p_a \) from the original point cloud \( S_0 \). Conditioned on the point \( p_a \), the network continues to select the second point \( p_b \) in Stage Two. If the robotic hand is two-fingered, the point selection procedure ends. The two points \( \{p_a, p_b\} \) are the two contact points to grasp. For a three fingered robotic hand, the third point \( p_c \) will be selected during Stage Three. For N-fingered robotic hands, there will be \( N \) stages. In each Stage \( n \), the neural network will select the next point conditioned on the previously selected \( n − 1 \) points.

For one object and an N-fingered gripper, we denote the list of all valid point sets as \( V = \{\hat{p}_1^n\} \). An item in this list is a set containing \( N \) points. A set is said to be valid if it
forms force closure and is reachable by the associated robotic hand. However, it is challenging to find these valid point sets among the vast number of possible contact point sets in the object point cloud. To add robustness to the point selection procedure, we adopt an approach akin to beam search [24] where during each stage the network selects multiple, most promising candidate points.

The point cloud of an object \( S_0 = \{ p_i \}_{i=1}^L \) has dimension \( (L, 3) \) and forms the input to the point set selection network. We use PointNet++ [21] to extract features and predict a normal per point in this point cloud. We denote these extracted features with a shape of \( \{ p_i \}_{i=1}^L \). The extracted features are of dimension \( (L, 64) \).

In practice, we find it is easier to train the PSSN when rejecting a large number of invalid point sets. To this end, we leverage simple heuristics [25] to reject invalid point sets based on point positions and the predicted point normals. For two-fingered robotic hands, the angle between the two contact point normals needs to be larger than 120\(^\circ\) to be in force closure [26]. For three-fingered robotic hands, three contact points form a triangle. We set a constraint that each side of the triangle needs to be larger than 1 cm to prevent our model from selecting points which are too close to each other. In addition, we prefer contact points that form a more regular triangle and we set another two constraints. The maximum internal angle of the triangle needs to be less than 120\(^\circ\). For each point, the angles between its normal vector and two incident edge directions are less than 90\(^\circ\). It indicates that the point normal is pointing outwards of the triangle. Generally three contact points in force closure fulfill those constraints above.

The detailed architecture of PSSN is shown in Fig. 4. In the following, we describe each stage in detail.

**Stage One:** We first select \( K_1 \) points denoted as \( S_1 \) from the original point cloud \( S_0 = \{ p_i \}_{i=1}^L \) as follows. Object features with a shape of \( (L, 64) \) are fed into a 1d convolution and soft-max layer to calculate the probability of each point being valid. The model then selects the \( K_1 \) points with the highest scores and gathers the corresponding features denoted as \( F_1 \) with a dimension of \( (K_1, 64) \).

**Stage Two:** This stage aims to select points from \( S_0 \) that conditioned on the points in \( S_1 \) yield valid point sets. As a first step, Stage Two selects \( K_2 \) points from \( S_0 \) in the same way as Stage One. This set of points is referred to as \( S_2 \). The network collects the corresponding point features in \( F_2 \) which is of dimension \( (K_2, 64) \). Then, feature maps \( F_1 \) and \( F_2 \) are put into a reshape-1d-cnn layer visualized in Fig. 4. The network performs a pairwise copy and concatenation to produce a combined feature map \( F_3 \) of dimension \( (K_2, K_1, 64) \). The feature map is then sent to 1d convolution and soft-max layer to get the final score matrix of dimension \( (K_2, K_1) \). We collect the top \( K_3 \) elements denoted as \( S_3 \). For two fingered robotic hands, these two points are the contact points predicted by PSSN.

**Stage Three:** For the \( N \)-fingered robotic hand \( (N \geq 3) \), the network has to select a third contact point. For this purpose,
the network accesses the features of each element in the set $S_3$. Each index in the set $S_3$ refers to two previously selected points of Stage One and Stage Two. The network accesses the corresponding features and concatenates them. The new feature map has a dimension of $(K_3, 128)$ and is fed into a 1d convolutional layer with an output of dimension $(K_3, 64)$ denoted as $F_3$. We select the third point index from the already reduced set $S_3$. For this purpose, the network simultaneously gathers the feature map $F_3$ with dimension $(K_3, 64)$ and feature map $F_1$ with dimension $(K_1, 64)$. These two feature maps are fed into a reshape-1d-cnn layer to get a pairwise concatenation with a dimension of $(K_3, K_1, 64)$ denoted as $F_5$. The feature map $F_5$ is input to a 1d convolution and soft-max to calculate the score matrix of dimension $(K_3, K_1)$. Each element $(u, v)$ in this matrix corresponds to the $u_{th}$ element in $S_3$ and the $v_{th}$ point in set $S_3$. Note that the $u_{th}$ element in the set $S_3$ refers to two points selected in the previous two stages.

Based on the score, the network determines the $K_4$ top contact points sets. These form the candidate grasps for the three-fingered robotic hand. In this paper, we only implemented a network for a maximum of three fingers. For N-fingered robotic hand, there would be N-3 more stages.

C. Loss Function and Optimization

Before PSSN we train the neural network to predict point normals $\{\hat{pn}_i\}_{i=1}^L$. Given the ground truth point normals $\{pn_i\}_{i=1}^L$, the loss of predicted point normals is defined to be

$$L_{pn} = -\sum_{i=1}^L (\hat{pn}_i \cdot pn_i)$$

where $\cdot$ represents the dot product.

[27] found that formulating grasp success prediction as a ranking problem yields higher accuracy than formulating it as a binary classification problem. Therefore we formulate point selection at every stage as a learning-to-rank [28] problem. For an object point cloud and a gripper, every point set which is in force closure and reachable by this specific robotic hand is annotated as positive. All other point sets are labeled negative. Therefore, each point set has a binary label $y$. In Stage n (where $n \leq N$), the network predicts a list of $K_n$ point sets and their corresponding probability $\hat{y}$ of whether this set is valid or not. Based on their predicted probabilities, we have a ranked list of these point sets. We use a variant of the ListNet [29] loss to increase the $\hat{y}$ of point sets with positive labels.

$$L_n = -\sum_{j=1}^{K_n} y_j \log\left(\frac{\exp(\hat{y}_j)}{\sum_{j=1}^{K_n} \exp(\hat{y}_j)}\right)$$

We use the Adam optimizer [30] for training the network, set the learning rate to $10^{-4}$, and split the data into 80/20 for training and test sets. We train the neural network stage by stage. First, we only train Stage One. Only the loss of Stage One is computed and the gradients are back-propagated to the weights of the layers in Stage Two. After training Stage Two, we fix the weights in Stage One and continue to train the layers in Stage Two. Only the loss of Stage Two is computed. The training procedure continues until Stage N.

IV. GRASP DATASET GENERATION

To train UniGrasp model, we require data that consists of object point clouds annotated with sets of contact points that are in force closure and reachable by specific grippers. We generate this data set in simulation as commonly done for other data-driven approaches [4, 6, 7, 32]. To construct this dataset, we select 1000 object models that are available in Bullet [31] and scale each object up to five different sizes to yield 3275 object instances. We use 11 different robotic hands. Eight of these hands are two-fingered robotic grippers and three of them are three-fingered. The data generation process is visualized in Fig. 5.

We place the object on a horizontal plane and render the object from eight viewpoints to generate RGB-D images. We reconstruct a point cloud from those RGB-D images and down-sample them to 2048 points. We find 2048 points are dense enough to represent the geometry of the object given GPU memory constraints.

Given a robotic hand with $N$ fingers ($N= 2, 3$,...), we generate all possible sets of $N$ points that are in force closure and reachable by the given robotic hand. For an object represented by a point cloud of 2048 points and a three-fingered robotic hand, the total number of combinations is $\binom{2048}{9} \approx 1.4 \times 10^9$. We first utilize the heuristics described in Section III-B to reject large amounts of point sets and then use the FastGrasp [33] to efficiently evaluate the remaining point sets. We compute the grasp quality score $Q^{-}$ assuming a friction coefficient of 0.5 for two-fingered grippers and of 0.65 for three-fingered grippers. We use a polygonal approximation of the friction cone with 16 faces. We label a point set to be in force closure, if $Q^{-} > 0.0001$.

If a set of $N$ points is in force closure, we use inverse kinematics for each $N$-fingered hand to determine (a) whether the points are reachable and (b) whether the grasp is collision-free - both specific to the hand. All sets of contact points that are reachable and collision-free by a specific gripper, then it is added to the final dataset used to train UniGrasp.
are in force closure, reachable and collision-free are labeled as positive for a specific gripper and negative otherwise.

V. Experiments

In this section, we extensively evaluate the proposed grasp synthesis method in terms of the accuracy of contact point selection for various known N-fingered robotic hands both in simulation and in the real world experiments. We demonstrate the validity of the learned gripper embedding space and show that our model can produce valid contact points for different novel robotic hands, both in simulation and the real world experiments.

A. Robotic Hand Presentation Learning

For training the autoencoder to represent the gripper, we split the gripper point clouds described in Sec. III into training and test set with a ratio of nine to one. We use five consecutive layers of 1D convolution in PointNet [20]. The feature channel dimensions of each convolution layers are (64, 64, 128, 128, 256, 256) starting from input layers. The final feature embedding space has a dimension of 256. The training and test Chamfer Distance losses are $4.87 \times 10^{-5}$ and $4.44 \times 10^{-5}$ respectively.

To demonstrate that the learned representation is able to capture robotic hand attributes like prismatic and revolute joint movement as well as variety in geometry, we show the linear interpolation ability between two points in the latent space. We take two point clouds $Pc_1$ and $Pc_2$ and feed them into the autoencoder. We extract the corresponding features from the encoder denoted as $F_1$ and $F_2$. We generate new features $F_3 = 0.8 F_1 + 0.2 F_2$ and $F_4 = 0.2 F_1 + 0.8 F_2$ by interpolation between $F_1$ and $F_2$. $F_1 - F_4$ features are put into the decoder to reconstruct point clouds. The results are shown in Fig. 6 and indicate the meaningful interpolation not only in configuration space but also in shape space.

B. Grasp Point Set Selection Performance

In simulation PPSN selects $K_1 = 1024$ in Stage One, $K_2 = 1024$ and $K_3 = 1024$ in Stage Two and $K_4 = 1024$ in Stage Three. We run six experiments to demonstrate the strength of our proposed method in predicting valid contact points for up to three-fingered robot hands. We adopt two evaluation metrics to reflect prediction accuracy. Top1 refers to the prediction accuracy of the highest ranked point to be within 5mm of a valid point set $\mathcal{V}$, i.e. of a point set that forms a force closure grasp, is reachable and collision-free by the input hand. Top10 refers to the percentage of test cases where at least one valid point set of the top 10 predicted ones is within 5mm of a valid grasp. We allow the 5mm relaxation for two reasons. a) There is always annotation noise e.g. in the point normal or force closure estimation. b) In the real world, fingertips often have a width of 10mm. With a 5mm tolerance the fingertip still covers the valid contact point. A similar relaxation is also used in previous work [13]. Neighbor reports the ratio of valid point sets over all possible point sets within a local neighborhood of our prediction. We set the radius of this local neighborhood around each contact point to be 5mm. A high number shows that there are more dense valid points around a predicted contact point. This is helpful during grasp acquisition where noise in the point clouds and in robot actuation may lead to alignment errors between the robot fingers and contact points.

To test our model’s generalization ability to novel objects, we train a neural network on the training dataset and report its performance on the test dataset. The results are given in Fig. 8 We parse the URDF of various grippers and generate their features as described in Sec. III. During each training stage, our model selects point sets and ranks them. For example, given a two-fingered robotic hand, the network has two prediction stages.

For two-fingered grippers, our model achieve 92.8% and 83.7% Top1 accuracies for Saywer and Franka, respectively. We also test the grasping performance for three-fingered robotic hands. We train our point set selection network on the objects with annotation of Kinova, Robotiq and Barrett hand, respectively. Our model achieves 77.1%, 86.9% and 89.6% Top1 test accuracy.

To test our model’s generalization ability to novel robotic hands, we first train the neural network model on the dataset annotated with all two-fingered grippers except the Sawyer. In the test stage, we first parse the URDF file of Sawyer and extract corresponding gripper features. Then we feed the
Fig. 8. Summary of our grasping evaluation in simulation for two-fingered and three-fingered grippers. We report prediction accuracies of PSSN at different stages for different grippers. We distinguish between two different evaluation strategies and denote them by T and U respectively. A gripper prefixed with T (e.g. T Sawyer) indicates the gripper was trained and evaluated on train-test-splits dataset while both splits contained samples of that gripper (among other grippers). In contrast, U-prefixied grippers denote that samples of the gripper were not used during training.

Sawyer gripper feature into our neural network model and evaluate the grasp performance on the test dataset annotated for the Sawyer gripper. Our model achieves 86.6% accuracy on the Top1 prediction. There are on average 43% valid points that are Neighbors of the predicted contact point. This shows that our model clusters valid contact point sets in regions on the object surface. This is beneficial for real-world grasp executions under perception and actuation noise.

Fig. 7 illustrates that our model generates distinct contact point sets on the same example object for various robotic hands. This shows that our model generates contact points that are specific to the robotic hand attributes.

C. Real World Experiments

We evaluate the performance of our model on 22 objects shown in the corresponding video. The object set includes eight adversarial objects [6] and three objects from the YCB object data set [39]. Our hand-eye camera setup uses an RGB-D camera to capture the depth images of a given object. The depth images are inpainted [40] using OpenCV to remove invalid values. Utilizing the camera matrix, we can reconstruct 3D point clouds. We feed the gripper and object point clouds into our UniGrasp model. UniGrasp outputs each finger’s contact point. We solve the inverse kinematics for the robotic gripper and command the robot to approach the object using a Cartesian space controller. We close the fingers and lift the object, both open-loop.

We apply our model in the real world using Kinova KG-3 and Robotiq-3F grippers. To test the generalization of our model over grippers, we remove one finger of Kinova KG-3 in the URDF file to create a novel two-fingered gripper. We also apply UniGrasp to the Schunk SVH five-fingered robotic hand [40]. Our UniGrasp model produces three contact points for the thumb, index and ring finger. After solving inverse kinematics for those three fingers, we control the five fingers by having the middle finger mimic the thumb and the pinkie mimic the ring finger. The results are shown in Fig. 2. Our method achieve 92% and 95% successful grasps for Kinova-KG3 and Robotiq-3F grippers. It also generates 93% for the novel two-fingered gripper. Although in our experiment the thumb of Schunk hand is broken, our model achieves 83% successful grasps.

| Accuracy | Stage One Train | Stage One Test | Stage Two Train | Stage Two Test | Stage Three Train | Stage Three Test | Neighbor |
|----------|----------------|---------------|----------------|---------------|------------------|-----------------|---------|
|          | Top1  Top10    | Top1  Top10    | Top1  Top10    | Top1  Top10    | Top1  Top10      | Top1  Top10      | Neighbor |
| T Sawyer | 99.8 100.0     | 99.6 100.0     | 93.6 97.6      | 92.8 96.8      | -                | -                | 53      |
| U Sawyer | -               | 97.4 100.0     | -              | 86.6 95.4      | -                | -                | 43      |
| T Franka | 99.0 99.8      | 98.7 100.0     | 84.7 95.6      | 83.7 91.0      | -                | -                | 43      |
| T Kinova | 96.6 99.7      | 96.8 99.8      | 91.6 96.3      | 91.6 96.3      | 76.3 89.5        | 77.1 90.7        | 26      |
| T Robotiq| 98.2 99.8      | 98.4 99.6      | 95.6 98.3      | 96.3 98.4      | 86.6 94.2        | 86.9 95.1        | 36      |
| T Barrett| 98.4 100.0     | 98.7 99.8      | 95.6 98.3      | 97.1 98.9      | 90.6 96.4        | 89.6 96.3        | 40      |

Fig. 9. Summary of our grasping evaluation in the real world for various grippers. The prefix T and U are the same as in Fig. 8. RunTime represents computation time spent on PPSN.

|       | Trials | Success Rate(%) | RunTime(s) |
|-------|--------|-----------------|------------|
| T Kinova-KG3 | 64     | 92              | 0.13       |
| T Robotiq-3F | 65     | 95              | 0.20       |
| U Kinova-2F | 60     | 93              | 0.06       |
| U Schunk SVH | 60     | 83              | 0.21       |

VI. CONCLUSION

We present a novel data-driven grasp synthesis approach named UniGrasp that generates valid grasps for a wide range of grippers from two-fingered parallel-yaw grippers to articulated multi-fingered grippers. UniGrasp takes point clouds of the object and the URDF of robotic hand as inputs. The outputs are the contact points on the surface of objects. We show in quantitative experiments that our method predicts over 90% valid contact points for various known two- and three-fingered grippers in Top10 predictions and over 90% valid grasps in real world experiments. Our model also generates 93% and 83% successful grasps for novel two-fingered and five-fingered robotic hands in real-world experiments.

REFERENCES

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis survey,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, April 2014.
[2] A. Sahbani, S. El-Khoury, and P. Bidaud, “An overview of 3d object grasp synthesis algorithms,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326 – 336, 2012, autonomous Grasping.
[3] 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.
[4] D. Kappler, J. Bohg, and S. Schaal, “Leveraging big data for grasp planning,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4304–4311.
[5] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3406–3413.
[6] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” 2017.
