AdaGrasp: Learning an Adaptive Gripper-Aware Grasping Policy

Zhenjia Xu Beichun Qi Shubham Agrawal Shuran Song
Columbia University
https://adagrasp.cs.columbia.edu

***Abstract*— This paper aims to improve robots' versatility and adaptability by allowing them to use a large variety of end-effector tools and quickly adapt to new tools. We propose AdaGrasp, a method to learn a single grasping policy that generalizes to novel grippers. By training on a large collection of grippers, our algorithm is able to acquire generalizable knowledge of how different grippers should be used in various tasks. Given a visual observation of the scene and the gripper, AdaGrasp infers the possible grasp poses and their grasp scores by computing the cross convolution between the shape encodings of the gripper and scene. Intuitively, this cross convolution operation can be considered as an efficient way of exhaustively matching the scene geometry with gripper geometry under different grasp poses (i.e., translations and orientations), where a good "match" of 3D geometry will lead to a successful grasp. We validate our methods in both simulation and real-world environments. Our experiment shows that AdaGrasp significantly outperforms the existing multi-gripper grasping policy method, especially when handling cluttered environments and partial observations. Code and Data are available at https://adagrasp.cs.columbia.edu.**

I. Introduction

In many real-world systems, a robot's end-effector is designed with a specific application in mind, where its specific geometry and kinematic structure often lead to distinct strengths and weaknesses. However, the vast majority of robotic research has been limited to single end-effector setups where the learned policy cannot generalize to new gripper hardware without extensive retraining. On the other hand, we humans can easily use various tools to accomplish different tasks and quickly adapt to unseen tools. Can we allow our robot system to do the same? This capability would benefit a robot manipulation system in the following ways:

- Versatility via diversity. Since different gripper designs often provide complementary strengths and weaknesses, and by learning to adequately use *a diverse set of* grippers, the system can effectively improve its versatility on handling a larger variety of objects and tasks.
- Adaptability via generalization. Since the learned grasping policy can generalize across different gripper hardware, it can also quickly adapt to *new* grippers by directly analyzing its geometry and structure. It is different from the existing multi-gripper systems [1], [2] that need to collect new training data for any new gripper hardware.

The authors would like to thank Lin Shao and Unigrasp authors for sharing code and models for comparison, Iretiayo A. Akinola for his help in setting up BarretHand Gripper and Google for the UR5 robot hardware. This work was supported in part by the Amazon Research Award and the National Science Foundation under CMMI-2037101

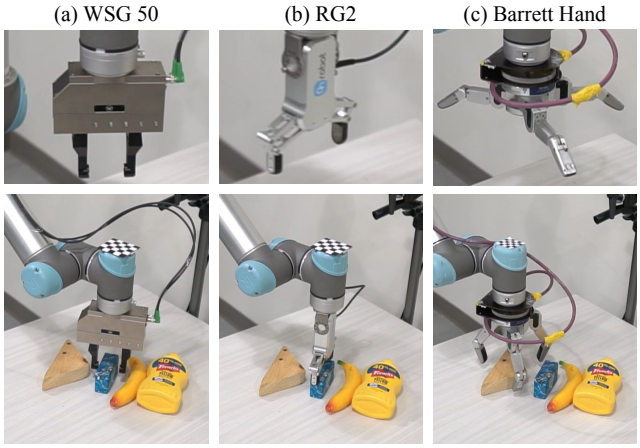

Fig. 1. **Gripper-Aware Grasping Policy.** The goal of AdaGrasp is to produce grasping strategies that are conditioned on input gripper description (a,b,c). For example, since the RG2 gripper has a wider fixed opening than WSG 50 (which can control its opening width), it chooses a different grasp pose to avoid double-picking or collision. Barrett Hand grasps the big triangle shape, which can be challenging for other two-finger grippers.

To achieve this goal, we propose **AdaGrasp**, a learning-based algorithm that learns a unified policy for different grippers and can generalize to novel gripper designs. At its core, AdaGrasp uses cross convolution (CrossConv)[3] operation between the shape encoding of the robot gripper and the scene to infer the grasp score for all possible grasp poses. Intuitively, this operation can be considered as an efficient way of exhaustively matching the scene and the gripper geometry under different grasp poses, where a good "match" of their 3D geometry will lead to a successful grasp.

The 3D geometry of a robot gripper and its kinematic structure often inform how it should be used for a given task [4]. By learning to use a large collection of different grippers, the algorithm should be able to acquire a generalizable knowledge of how different grippers should be used in various tasks. For example, a gripper's opening width determines what object shape can fit into the gripper, and the thickness of each finger determines what narrow space the finger can get into without collision. Fig.1 illustrates different grasp poses that are suitable for different grippers.

The primary contribution of this paper is AdaGrasp, a learning-based grasping algorithm that leverages generalized shape matching via cross convolution to produce a grasping policy that works across different gripper hardwares. We validate our methods in both simulation and real-world environments. Our experiments show that AdaGrasp outperforms the state-of-the-art method for multi-gripper grasping, especially in a cluttered environment and with partial observation.

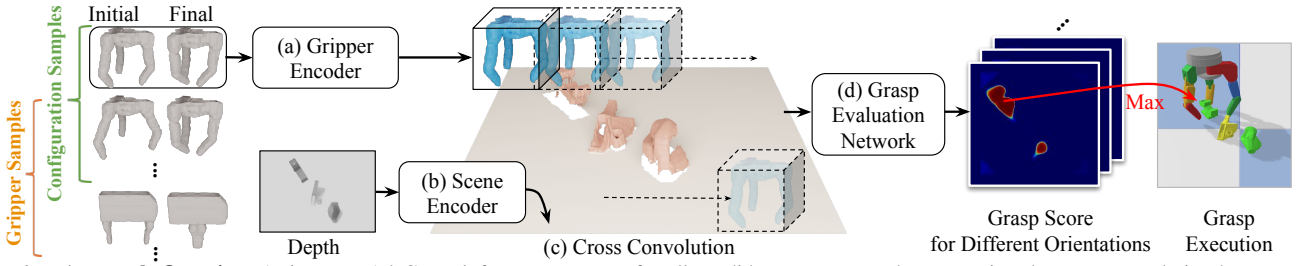

Fig. 2. **Approach Overview.** At its core, AdaGrasp infers grasp scores for all candidate grasp poses by computing the cross convolution between the gripper encoding produced by the gripper encoder (a) and the scene encoding produced by the scene encoder (b). This cross convolution operation (c) matches the scene and gripper encoding under all grasp poses by translating and rotating the gripper kernel, where a good "match" of their encoding results in a high grasp score. Different initial opening configuration of a gripper are treated as different grippers and fed to the network in parallel. The action associated with the highest grasp score is executed. The action for a grasp attempt includes selecting the suitable gripper, deciding its initial joint configuration, and choosing a proper grasp pose.

II. Related Work

Learning-based single-gripper systems. Recent data-driven methods have made great progress on learning object-agnostic grasping policies that detect grasps by exploiting visual features, without explicitly using object-specific prior knowledge [5]–[16]. These algorithms demonstrate the ability to generalize to new objects and scene configurations. However, they are often designed and trained with a fixed hardware setup. Hence, they cannot adapt to any changes in the gripper hardware without extensive retraining.

Learning-based multi-gripper systems. To take advantage of complementary skills between different grippers, more recent works have started to use multiple end-effectors for grasping. For example, both Zeng *et al*. [1], [17] and Mahler *et al*. [18] used a setup with one suction cup gripper and one parallel jaw gripper. However, in both the systems, the algorithm learns a separate policy for each gripper, i.e., their policies cannot generalize to new grippers. As a result, these algorithms are often limited to a small number of grippers.

Contact-based grasping policy. Many analytical grasping models have been proposed to evaluate grasp quality through contact-point reasoning and force-closure analysis [19]–[23]. The work most related to us is UniGrasp [24], where the algorithm takes in the gripper point cloud and a single object point cloud, samples N points from the object point cloud as contact points for N fingers, and uses inverse kinematics to get gripper joint configuration.

While a contact-based policy generalizes to new grippers, it also brings in limitations. First, since measuring precise contact points in real-world is challenging, the algorithm can only be trained with simulation. Moreover, it is trained using static force closure analysis, which does not consider the object dynamics during grasping. Second, to reason about force closure, the algorithm assumes a complete object representation as input which relies on a perception algorithm to perfectly detect the target object and provide full 3D geometry. Since the algorithm only samples contact points on the object surface, a partial observation of the object will lead to unstable contact point selection and inaccurate force closure evaluation, as we showed in our experiments. In contrast, our method's action space won't be limited by partial observation. Furthermore, it does not consider the gripper geometry beyond contact points, which increases the likelihood of collision in cluttered environments. In contrast, our algorithm does not require any explicit contact point supervision or complete object representation. Therefore, it can better handle cluttered environments and partial observation.

III. Approach

The goal of our algorithm is to learn a policy that can produce the optimal grasping strategy for a novel gripper by estimating the probability of grasp success (i.e., grasp score) for all candidate gripper configurations and grasp poses. Concretely, taking a visual observation of the scene (RGB-D images) and the gripper design (defined as URDF files) as input, the algorithm infers the possible grasp poses along with their grasp scores that would allow the gripper to successfully grasp a target object.

The core of our approach is a Grasp Evaluation Network $f_{grasp}(s,g) \rightarrow a$ that infers the grasp score for all candidate grasp poses a by computing the cross convolution between the gripper encoding g and scene encoding s. The grasp pose is parameterized by rotation about the z-axis and 2D translation. This cross convolution operation can be considered as an efficient way of exhaustively matching the scene geometry with gripper geometry in all possible grasp poses by translating and rotating the gripper kernel. The matching score is finally represented as a dense grasp score map, where a higher value indicates a higher chance of a successful grasp. We train the algorithm with a collection of grippers and environment setups and test it with unseen grippers and objects. Fig. 2 shows the network overview, and the following sections provide details of our approach.

A. Gripper and Scene Representation

Gripper encoding. The gripper geometry is captured by 10 depth images and encoded as a 3D TSDF volume [25]. The volume dimension is $64 \times 64 \times 32$ (voxel) with voxel size $v_g = 0.004$ (m). We compute TSDF volume for the gripper at its initial open state and final closed state and stack them as input $I_g \in R^{2\times64\times64\times32}$. The gripper encoder network (Fig 2 a) starts with two 3D convolution layers with kernel size $3\times3\times3$, resulting in a feature $\in R^{64\times32\times32\times16}$. Then we use one 3D convolution with kernel size $1\times1\times16$ reducing the z dimension to 1. Finally, we use 5 2D convolution layers to produce the gripper features $\psi(g) \in R^{16\times32\times32}$.

Scene encoding The input scene is captured with a top-

Fig. 3. Training and testing grippers used in our experiments.

down depth image and encoded as a 3D TSDF volume. The workspace dimension is $192 \times 192 \times 64$ (voxel) with a voxel size $v_s = 0.002$ (m). In multi-object obstacle cases, the obstacle mask is provided as an additional channel. This channel will be 0 for other cases. The scene volume $I_s \in R^{2\times192\times192\times64}$ is then fed into the scene encoder network (Fig 2 b). Similar to the gripper encoder network, it consists of three 3D convolution layers with downsample scale=4, one layer for z-axis reduction, and five 2D convolution layers. The output is the scene features $\phi(s) \in R^{16\times48\times48}$.

B. Grasp Evaluation via Shape Matching

After the encoding network, the scene and gripper geometry are mapped into a query $\phi(s)$ and key $\psi(g)$ features. We carefully set the number of downsampling size in scene encoder and gripper encoder so that both features share a similar physical receptive field. As a result, the spatial alignment is maintained, and shape matching in feature space (via CrossConv) is meaningful. The algorithm then computes the cross convolution between the $\psi(g)$ and $\phi(s)$ by treating $\psi(g)$ as the convolution kernel (Fig. 2 c). The output shares the same size as the scene feature $\phi(s)$. We repeat this step for $r = 16$ times [26], each time rotate the scene TSDF volume by $\theta = 2\pi/r$ about z-axis. Finally, the output of cross convolution is fed into a grasp evaluation network (Fig 2-d) that estimates dense grasp scores for all possible actions $Q \in R^{X_s\times Y_s\times r}$, where each grasp score $Q(i,j,k)$ in the Q value map corresponds to one grasp pose.

The grasp pose is parameterized by its position (x,y,z) and orientation $\theta = k\pi/r$ about z-axis, where $x = x_{min} + v_s i$, $y = y_{min} + v_s j$, $z = \mathscr{H}(O(i,j)) - 0.05$, $[x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}]$ is the workspace bound, $\mathscr{H}(O(i,j))$ is the height of z-dimention in the scene volume O at location (i,j). During grasp execution, the gripper starts at location (x,y,z_{max}), moves downward along z-axis until having contact with an object or reaching the target position (x,y,z), and then close its finger. The gripper will then move upwards and this execution is considered successful if and only if exactly one target object is lifted > 0.2m. Grasping an obstacle or more than one objects is classified as a failure.

Network training. The whole network is trained end-to-end with self-supervised grasping trials, similar to prior work [5], [26]. Based on the object height after grasping, each grasp trial is labeled with its grasp outcome (1 = success, 0 = failure). The network is trained to predict the grasp outcome for all possible actions, and it is supervised by the grasping outcome of the executed action (one action out of $X_s \times Y_s \times r$ actions) using softmax loss.

During training, the network chooses its action using $\varepsilon-$greedy. We use the normalized predicted grasp scores as the probability of choosing each pose. At training epoch e, ε decreases linearly from ε_{max} to ε_{min}. After n epochs, $\varepsilon = \varepsilon_{min}$. We set $n = 2000, \varepsilon_{min} = 0.2, \varepsilon_{max} = 0.8$. All the grasp trails are stored in a FIFO replay buffer (size=12000). At each training step, we sample a batch of examples from the replay buffer with a 1:1 positive to negative ratio. We also used data augmentation to overcome overfitting. The scene inputs obtained from the replay buffer have a probability of 0.7 to be randomly shifted and rotated. We applied the same transformation to the corresponding grasp pose. The final model is trained for 5000 epochs, 8 sequences of data collection, and 32 iterations of training per epoch with Adam optimizer and learning rate 0.0005.

C. Improving Grasp Quality via Gripper Selection

To execute the grasp, the algorithm selects the predicted best action from the grasp evaluation network $a = \arg\max_a Q$. However, depending on the input gripper, sometimes even the best action might still not be good enough to achieve a successful grasp (e.g., the input gripper or its initial configuration is too small to enclose the object inside). In such cases, the algorithm will compare and select between different input grippers to improve its grasp quality.

To do so, the network predicts a grasp score for a list of N candidate grippers, then selects the one that produces the highest grasp score. Note that the list of candidate grippers can include *completely different grippers* or *the same gripper with different initial joint configurations*. Since the grasp evaluation network is trained for many grippers, the estimated grasp score for different grippers is naturally comparable, where a higher score indicates a better gripper for the task. During testing, we allow the algorithm to choose the best configuration for a given gripper (AdaGrasp-fixGripper in Tab. I) or choose both the best gripper and its best configuration at the same time (AdaGrasp in Tab. I).

Configuration Sampling. To sample possible initial configuration for a given gripper, we linearly map the gripper's joint configuration into a scalar value in the range [0,1], where 0 represents the fully closed state, and 1 represents the fully open state. Note that the algorithm only needs to choose grippers' initial configuration, since the final configuration is determined – the gripper will always try to close its fingers all the way to its fully closed state.

During training, each gripper has 4 initial configuration options randomly sampled between 0.4 and 1.0. Since two fingers of Barrett Hand have flexible palm joints, we define the following 3 presets: (1) palm joint = 0, two flexible fingers are parallel and next to each other. (2) palm joint = 0.1π, the angle between two flexible fingers is 0.2π. (3) palm joint = 0.5π and remove the finger with a fixed palm joint. This configuration mimics a broken Barrett Hand with two remaining fingers (Barrett Hand-B).

TABLE I

GRASP SUCCESS RATE.

Algorithm	Single object				Multi-object				Multi-object w. obstacles			
	O_{tr}-G_{tr}	O_{tr}-G_{te}	O_{te}-G_{tr}	O_{te}-G_{te}	O_{tr}-G_{tr}	O_{tr}-G_{te}	O_{te}-G_{tr}	O_{te}-G_{te}	O_{tr}-G_{tr}	O_{tr}-G_{te}	O_{te}-G_{tr}	O_{te}-G_{te}
SceneOnly	0.613	0.730	0.596	0.686	0.493	0.528	0.497	0.531	0.347	0.368	0.271	0.303
SingleGripper [26]	-	0.930	-	0.930	-	0.788	-	0.886	-	-	-	-
UniGrasp [24]	-	-	-	0.812	-	-	-	0.228	-	-	-	-
AdaGrasp-initOnly	0.721	0.792	0.719	0.753	0.637	0.674	0.638	0.626	0.494	0.353	0.513	0.343
AdaGrasp-fixConfig	0.766	0.855	0.770	0.854	0.706	0.751	0.685	0.764	0.612	0.523	0.603	0.569
AdaGrasp-fixGripper	0.923	0.905	0.959	0.938	0.849	0.842	0.875	0.854	0.775	0.658	0.813	0.703
AdaGrasp	**0.960**	**1.000**	**0.970**	**0.990**	**0.912**	**0.908**	**0.896**	**0.936**	**0.853**	**0.747**	**0.887**	**0.793**

Test case is labeled by $O_{\text{object type}}$-$G_{\text{gripper type}}$ (tr: train, te: test). Note: UniGrasp is tested with 4-camera input, all others are tested with 1-camera input.

IV. EXPERIMENTS

We run the following experiments to verify that the proposed AdaGrasp algorithm is able to (1) learn different grasping strategies for different grippers, (2) generalize to new grippers, (3) select a suitable gripper and gripper configuration for a given task. We have also provided real-world experiments to validate our approach.

Scene setup: We use Pybullet [27] as our simulation environment. The target objects and obstacles are randomly dropped within a rectangular workspace. All objects used in simulation are from Dexnet 2.0 [28] object dataset. The training dataset has 801 objects: 400 from the 3DNet subset and 401 from the Kit subset. The test dataset has 57 objects: 13 from Adversarial subset and the remaining object from the Kit category that are not used in training.

For our method, we use a single top-down RGB-D camera to capture the scene. For UniGrasp, we use 3 additional cameras to provide a complete 3D point cloud input since it is sensitive to partial observation. Tab. II studies both algorithm's performance with respect to scene visibility. We tested the following scenarios:

- Single object. One random object is dropped into the scene with random position and orientation.
- Multiple objects. There are 5 objects in the scene, and the gripper is expected to grasp one object at a time until the scene is empty or a maximum attempt of 7 is reached.
- Multiple objects with obstacles. There are 3 targets and 3 obstacles. We provide the obstacle mask. The algorithm needs to grasp the target object while avoiding obstacles.

Gripper: We have 7 training grippers and 4 testing grippers as shown in Fig. 3. One of the testing grippers is Barrett Hand with one finger missing, which is equivalent to a 2 finger gripper. During training, grippers are globally scaled by a random factor of $t \in (0.8, 1.2)$ to increase the training gripper diversity. During testing, gripper scale is fixed at 1.

Metric: The algorithm performance is measured by grasp success rate = $\frac{\#\text{successful\_grasps}}{\#\text{total\_grasp\_attempts}}$. The grasp success for each attempt is measured by whether the gripper grasps strictly one target. For example, in the multi-object setup, grasping two objects simultaneously is considered a failure (double-picking). The objects can be grasped in any order.

We evaluate the algorithms on all grippers separately and use the average performance, except in our final policy, the algorithm has the freedom to select from a set of grippers. For each type of scene, the test scene generation is consistent across all algorithms and grippers.

Algorithm comparisons:

- UniGrasp [24]: it takes in the gripper point cloud and object point cloud (background removed), samples N (2 or 3) points from the object as contact points for N fingers, respectively, and use inverse kinematics to compute gripper joint configurations for grasp execution. We directly test the pre-trained model provided by the authors.
- SceneOnly: a single policy trained using all training grippers (uniformly sampled during training). The policy can only access the scene observation without gripper information; hence, it predicts uniformly across all grippers.
- SingleGripper [26]: a learning based grasping method from Zeng et al. using only Robotiq 2F-85.
- AdaGrasp-initOnly: the gripper input is the initial gripper state. The policy selects the best grasp pose (position and orientation) for a given gripper.
- AdaGrasp-fixConfig: same as AdaGrasp-initOnly, but gripper input has both its initial and final state.
- AdaGrasp-fixGripper: the algorithm linearly samples the gripper configurations and infer grasp score for each configuration. Then, the algorithm selects the gripper configuration with the highest grasp score to execute.
- AdaGrasp: On top of the gripper configuration and grasp pose, this algorithm also selects the best gripper with the highest grasp score to use. This is our final policy.

In testing, SceneOnly, AdaGrasp-initOnly, and AdaGrasp-fixConfig uses a random initial configuration sampled from [0.5, 0.625, 0.75, 0.875, 1.0]; AdaGrasp-fixGripper and AdaGrasp will select the configuration from the same list.

A. Experimental Results

Comparison to prior work. We compare our approach with state-of-the-art multi-gripper system UniGrasp [24]. The number of cameras during AdaGrasp's training is randomly chose in {1,2,3,4}. Both algorithms are evaluated on test objects and test grippers under a fixed-gripper and fixed-camera setting (i.e., the algorithm can choose the input gripper's initial configuration but cannot switch gripper). In the single object case, AdaGrasp-fixGripper achieves better performance (+10%) comparing to UniGrasp. The advantage is much more salient in multi-object case, where AdaGrasp-fixGripper is able to outperform UniGrasp by around 60%.

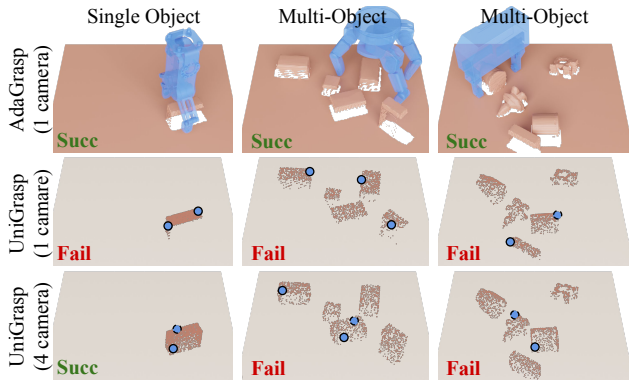

Fig. 4. **Comparisons.** UniGrasp often fails on incomplete input point clouds since it samples contact points directly from the pointcloud (2nd row with 1 camera). It also struggles with cluttered scenes, frequently sampling contact points on multiple objects or failing to account for collision. AdaGrasp is able to handle both partial observability and scene clutter.

TABLE II

GRASP SUCC RATE W.R.T PARTIAL OBSERVATION.

	Single Object				Multi Object			
# Camera	4	3	2	1	4	3	2	1
UniGrasp [24]	0.812	0.788	0.768	0.732	0.228	0.258	0.254	0.175
AdaGrasp-fixGripper	0.896	0.892	0.889	0.891	0.854	0.863	0.846	0.821

The success rate of UniGrasp degrades as the number of cameras and scene visibility decreases, whereas AdaGrasp performs consistently throughout. Both algorithms are tested with our test objects and test grippers under a fixed-gripper setting.

This result highlights AdaGrasp's ability in handling cluttered environments. Fig. 4 shows qualitative comparisons, where UniGrasp samples contact points on multiple objects or misses potential collisions.

Another advantage of AdaGrasp is its ability in handling partial observations. UniGrasp is very sensitive to the quality and visibility of scene observation since it directly samples contact points from the input point cloud, which is limited to the observed surface (Fig. 4-a). In contrast, AdaGrasp is able to reason about the object grasp point beyond the visible surfaces using 3D TSDF representation. Results in Tab. II demonstrate that when the scene observation is incomplete (i.e., with fewer cameras), UniGrasp's performance decreases significantly, while AdaGrasp has consistent performance. Inference time of AdaGrasp is 1.05s for each gripper with 5 initial configurations and 16 rotations.

Can AdaGrasp learn gripper-aware grasping policy? To verify AdaGrasp's ability to infer different grasping strategies conditioned on the input gripper, we perform the following experiments. All models in Tab. I are trained and tested under single-camera setting. First, we compare AdaGrasp-fixConfig with an "SceneOnly" policy, i.e., a single policy trained with all training grippers without the gripper as input. Results in Tab. I shows that AdaGrasp-fixConfig's performance is always significantly better than the "SceneOnly", which demonstrates that AdaGrasp-fixConfig improves the grasp prediction by analyzing the input gripper. We visualize the top grasp pose prediction for different grippers given the same scene setup (Fig. 6 7). From the visualization, we can see that the algorithm is able to infer diverse grasp poses that are suitable for each input gripper and configuration.

Can AdaGrasp generalize to new grippers? To test the algorithm's adaptability to new gripper hardware, we tested the learned policy with five unseen grippers, including three 2-finger grippers, one 3-finger grippers, and a "damaged" 3-finger gripper (Barrett hand with a missing finger). While test grippers are never used during training, AdaGrasp-fixGripper is able to get performance comparable to that on the training grippers. In Tab. I AdaGrasp-fixGripper improves the SceneOnly policy performance by 18% to 54%.

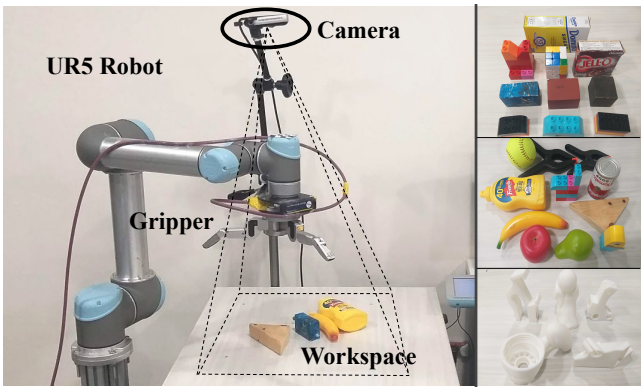

Fig. 5. **Real-world Setup.** Robot and camera setup (left) and test objects (right). Videos of experiments are available in supp. video and website.

TABLE III

REAL-WORLD GRASP SUCC RATE ON UNSEEN GRIPPERS AND OBJECTS.

	WSG 50	RG2	Barrett Hand	Barrett Hand-B
Single Object	0.92	0.92	0.88	0.80
Multiple Objects	0.90	0.88	0.78	0.66

Can AdaGrasp select the right configuration and gripper for a given task? To check whether the predicted grasp score is informative for comparing and selecting the gripper's initial configuration, we compare the algorithm performance with and without configuration selection (AdaGrasp-fixConfig v.s. AdaGrasp-fixGripper). Both algorithms predicts the grasp scores for the same gripper. The difference is that AdaGrasp-fixGripper selects the configuration with the highest grasp score while AdaGrasp-fixConfig randomly picks one configuration. Compared to AdaGrasp-fixConfig, AdaGrasp-fixGripper performance is better in all cases, improving 5% to 21%. This result validates that the predicted grasp score is informative for selecting the best initial configuration. Fig. 7-b shows an example of configuration selection for WSG 50.

Similarly, we showed that the grasp score is also comparable across different grippers. As a result, the algorithm is able to further improve its grasping performance by choosing the "right tool" (gripper) for a given task at hand (object to grasp). Comparing AdaGrasp with AdaGrasp-fixGripper in Tab. I, we can see the 1% to 9% improvement in all scenarios. The performance of AdaGrasp is also better than SingerGripper, which only evaluates on Robotiq 2F-85. This result indicates that if combined with an automatic tool changing hardware [29], AdaGrasp can improve the grasping performance by allowing the system to properly use a diverse set of grippers.

Is gripper final state encoding helpful? The input gripper encoding includes both gripper's initial and final state. It

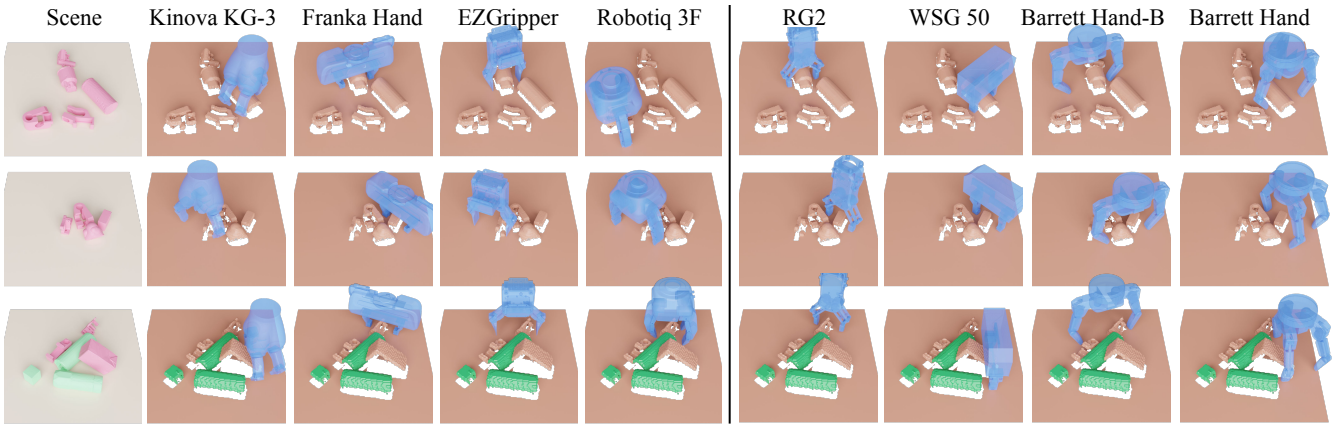

Fig. 6. **Gripper-Aware Grasping Policy.** Given the same input scene in each row, AdaGrasp predicts a different grasp pose suitable for each gripper. Here are example grasps inferred by AdaGrasp for training grippers (left) and testing grippers (right) in multi-object setups (Row 1-2), and multi-object + obstacle setups (Row 3). Brown surface: input TSDF. Green surface: obstacles input as additional mask. More examples available on our website.

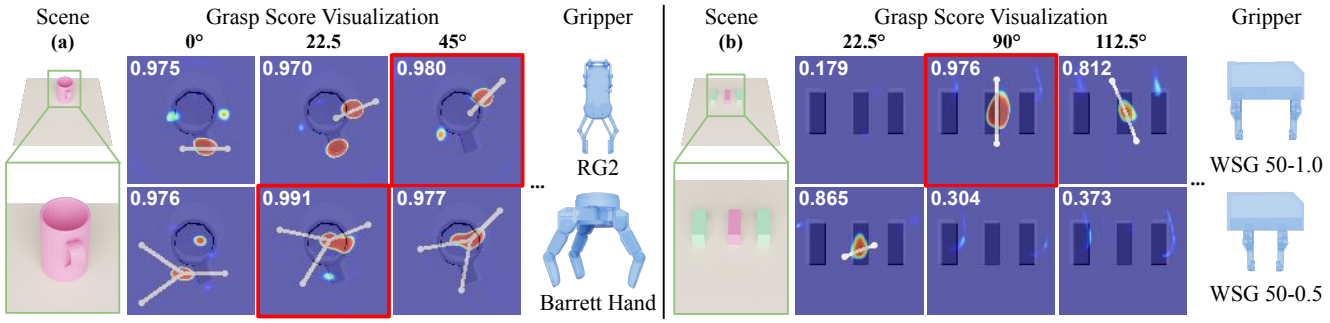

Fig. 7. **Grasp Score Visualization.** Dense grasp score predictions are shown for 3 out of 16 different grasp orientations. The highest grasp score for each orientation is shown at the top left. For each gripper, the orientation with the highest score is highlighted in red. In scene (a), the target object is a mug. RG2 prefers to grasp the cup's edge or handle, while Barrett Hand prefers to grasp across the whole cup. In scene (b), the target object is surrounded by two obstacles (green). We visualize the grasp poses for the WSG 50 gripper under different initial configurations (opening size). With a larger opening, the algorithm chooses to grasp vertically (90°) to avoid collisions, while with a smaller opening, it chooses to grasp horizontally (22.5°) since the object's length is now larger than the gripper width. Between these two configurations, the algorithm chooses the wider opening.

allows the algorithm to reason about the gripper's dynamics during the closing action beyond its static 3D geometry. To see the effect of final state encoding, we compare the model without the final-state, which is AdaGrasp-initOnly. In almost all test cases, AdaGrasp-fixConfig has a higher success rate, and it is most salient in the multi-object with obstacles setup (up to +23% improvement). Moreover, AdaGrasp-fixConfig demonstrates better generalizability when testing on new gripper hardware.

Real-robot experiment Finally, we validate our method on a real-world robot platform with a UR5 robot and a calibrated RGB-D camera (Intel RealSense D415). Fig. 5 shows the real-world setup and test objects. In this experiment, we directly tested AdaGrasp-fixGripper policy trained in simulation on four different physical grippers – WSG 50, RG2, Barrett Hand, and Barrett Hand-B, all of which are unseen during training. The test objects used in this experiment include 20 objects from YCB dataset [30] and five 3D printed adversarial objects from DexNet 2.0, all unseen during training. For single object tests, we place a single object randomly. For multi-object tests, we created 8 scenes each containing 4 randomly chosen objects and made sure that the placement of objects in 8 scenes is consistent across grippers for fair comparison. For each multi-object scene, we provide 7 attempts to a gripper for grasping objects. The grasp success rates are reported in Tab. III. The average success rates for single object and multi-object are 86% and 80.5%, respectively, comparable with the algorithm performance in simulation. We noticed that unlike parallel jaw grippers, Barrett Hand and Barrett Hand-B have a curved grasping gait, i.e., fingers take a curved trajectory while closing in. Thus, the Barrett Hand cannot create contact at a smaller height and fails to grasp shorter objects like banana and adversarial objects. On the other hand, Barrett Hand is good at grasping bigger objects like big triangle or baseball ball, which are challenging for smaller grippers like RG2.

V. Conclusion and Future Directions

We introduced AdaGrasp, a unified policy that generalizes to novel gripper designs. Extensive experiments demonstrate that AdaGrasp is able to improve the system's versatility and adaptability, and outperforms the current state-of-the-art multi-gripper grasping method. However, since our algorithm focuses on the gripper geometry for mechanical gripper, it does not extend to other gripper types (e.g., suction or deformable) and variable physical parameters (*e.g.* friction). It is also limited to top-down grasps due to the reduced action space. As future directions, it will be interesting to investigate larger range of gripper types in general dexterous manipulation.

REFERENCES

[1] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. Chavan-Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[2] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[3] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," in *Advances In Neural Information Processing Systems*, 2016.

[4] H. Ha, S. Agrawal, and S. Song, "Fit2form: 3d generative model for robot gripper form design," *arXiv preprint arXiv:2011.06498*, 2020.

[5] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," *Robotics and Automation Letters*, 2020.

[6] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *ICRA*, 2015.

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

[8] 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.

[9] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Int'l Symp. on Robotics Research*, 2017.

[10] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2901–2910.

[11] M. Gualtieri and R. Platt, "Learning 6-dof grasping and pick-place using attention focus," in *Proceedings of 2nd Conference on Robot Learning (CoRL 2018)*, 2018.

[12] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *RSS*, 2018.

[13] C. Choi, W. Schwarting, J. DelPreto, and D. Rus, "Learning object grasping for soft robot hands," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2370–2377, 2018.

[14] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-dof grasping interaction via deep geometry-aware 3d representations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3766–3773.

[15] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "PointNetGPD: Detecting grasp configurations from point sets," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[16] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6232–6238.

[17] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

[18] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[19] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 4415–4420.

[20] M. Veres, M. Moussa, and G. W. Taylor, "Modeling grasp motor imagery through deep conditional generative models," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 757–764, 2017.

[21] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5062–5069.

[22] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2442–2447.

[23] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1957–1964.

[24] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, "Unigrasp: Learning a unified model to grasp with multifingered robotic hands," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2286–2293, 2020.

[25] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.

[26] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[27] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[28] 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," *RSS*, 2017.

[29] A. industrial Automation, "Automatic / robotic tool changers," -. [Online]. Available: http://engineering.purdue.edu/~mark/puthesis

[30] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

TABLE A1

GRASP SUCCESS RATE OF EACH GRIPPER

Scene	Algorithm	Object	Training Grippers								Testing Grippers					
			WSG 32	Sawyer	Franka	Robotiq 2F-140	EZGripper	Kinova KG-3	Robotiq 3F	**Avg.**	WSG 50	RG2	Robotiq 2F-85	Barrett Hand-B	Barrett Hand	**Avg.**
Single Object	SceneOnly	Train	0.360	0.340	0.710	0.820	0.740	0.600	0.720	**0.613**	0.750	0.720	0.670	0.730	0.780	**0.730**
		Test	0.380	0.330	0.530	0.860	0.820	0.620	0.630	**0.596**	0.680	0.560	0.640	0.830	0.720	**0.686**
	AdaGrasp -initOnly	Train	0.600	0.530	0.800	0.730	0.850	0.710	0.830	**0.721**	0.900	0.770	0.640	0.770	0.880	**0.792**
		Test	0.610	0.490	0.690	0.740	0.860	0.790	0.850	**0.719**	0.860	0.650	0.660	0.810	0.785	**0.753**
	AdaGrasp -fixConfig	Train	0.650	0.570	0.810	0.890	0.810	0.800	0.830	**0.766**	0.910	0.840	0.810	0.870	0.845	**0.855**
		Test	0.610	0.500	0.780	0.950	0.890	0.810	0.850	**0.770**	0.880	0.890	0.780	0.870	0.850	**0.854**
	AdaGrasp -fixGripper	Train	0.890	0.990	0.980	0.960	0.900	0.920	0.820	**0.923**	1.000	0.890	0.910	0.910	0.815	**0.905**
		Test	0.950	0.980	1.000	1.000	0.920	0.970	0.890	**0.959**	0.990	0.940	0.970	0.940	0.850	**0.938**
	AdaGrasp	Train	-	-	-	-	-	-	-	**0.960**	-	-	-	-	-	**1.000**
		Test	-	-	-	-	-	-	-	**0.970**	-	-	-	-	-	**0.990**
Multi Object	SceneOnly	Train	0.324	0.364	0.596	0.688	0.668	0.428	0.380	**0.493**	0.676	0.584	0.580	0.348	0.450	**0.528**
		Test	0.328	0.320	0.484	0.736	0.728	0.496	0.384	**0.497**	0.580	0.572	0.576	0.456	0.472	**0.531**
	AdaGrasp -initOnly	Train	0.556	0.504	0.664	0.656	0.752	0.616	0.712	**0.637**	0.760	0.652	0.532	0.648	0.776	**0.674**
		Test	0.544	0.440	0.568	0.732	0.788	0.672	0.724	**0.638**	0.700	0.484	0.504	0.712	0.730	**0.626**
	AdaGrasp -fixConfig	Train	0.596	0.504	0.724	0.904	0.784	0.664	0.764	**0.706**	0.780	0.824	0.684	0.688	0.778	**0.751**
		Test	0.528	0.440	0.692	0.904	0.864	0.684	0.684	**0.685**	0.804	0.800	0.684	0.760	0.770	**0.764**
	AdaGrasp -fixGripper	Train	0.860	0.912	0.936	0.852	0.852	0.820	0.712	**0.849**	0.916	0.892	0.856	0.760	0.784	**0.842**
		Test	0.884	0.916	0.956	0.912	0.848	0.904	0.704	**0.875**	0.940	0.876	0.880	0.796	0.780	**0.854**
	AdaGrasp	Train	-	-	-	-	-	-	-	**0.912**	-	-	-	-	-	**0.908**
		Test	-	-	-	-	-	-	-	**0.896**	-	-	-	-	-	**0.936**
Multi Object with Obstacle	SceneOnly	Train	0.193	0.193	0.400	0.513	0.507	0.307	0.313	**0.347**	0.367	0.420	0.400	0.293	0.360	**0.368**
		Test	0.133	0.140	0.260	0.433	0.407	0.267	0.260	**0.271**	0.353	0.300	0.267	0.307	0.287	**0.303**
	AdaGrasp -initOnly	Train	0.453	0.347	0.613	0.673	0.533	0.440	0.400	**0.494**	0.253	0.433	0.460	0.360	0.257	**0.353**
		Test	0.447	0.333	0.580	0.753	0.647	0.367	0.467	**0.513**	0.260	0.360	0.487	0.353	0.253	**0.343**
	AdaGrasp -fixConfig	Train	0.487	0.480	0.700	0.753	0.687	0.587	0.593	**0.612**	0.620	0.673	0.580	0.247	0.493	**0.523**
		Test	0.480	0.387	0.633	0.773	0.733	0.587	0.627	**0.603**	0.640	0.720	0.607	0.340	0.537	**0.569**
	AdaGrasp -fixGripper	Train	0.813	0.860	0.873	0.693	0.720	0.780	0.687	**0.775**	0.780	0.760	0.693	0.427	0.630	**0.658**
		Test	0.847	0.867	0.867	0.793	0.780	0.820	0.720	**0.813**	0.787	0.800	0.787	0.473	0.667	**0.703**
	AdaGrasp	Train	-	-	-	-	-	-	-	**0.853**	-	-	-	-	-	**0.747**
		Test	-	-	-	-	-	-	-	**0.887**	-	-	-	-	-	**0.793**