IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

HybGrasp: A Hybrid Learning-to-Adapt Architecture for Efficient Robot Grasping

Jungwook Mun[1], Khang Truong Giang[1], Yunghee Lee[1], Nayoung Oh[1], Sejoon Huh[1], Min Kim[1], and Sungho Jo[1]

***Abstract*—Despite the prevalence of robotic manipulation tasks in various real-world applications of different requirements and needs, there has been a lack of focus on enhancing the adaptability of robotic grasping systems. Most of the current literature constructs models around a single gripper, succumbing to a trade-off between gripper complexity and generalizability. Adapting such models pre-trained on one type of gripper to another to work around the trade-off is inefficient and not scalable, as it would require tremendous effort and computational cost to generate new datasets and relearn the grasping task. In this letter, we propose a novel hybrid architecture for robot grasping that efficiently learns to adapt to different gripper designs. Our approach involves a three step process that first obtains a rough grasp pose prediction from a parallel gripper model, then predicts an adaptive action using a convolutional neural network, and finally refines the predicted action with reinforcement learning. The proposed method shows significant improvements in grasping performance compared to existing methods for both generated datasets and real-world scenarios, presenting a promising direction for improving the adaptability and flexibility of robotic manipulation systems.**

***Index Terms*—Deep Learning in Grasping and Manipulation, Reinforcement Learning, Multifingered Hands**

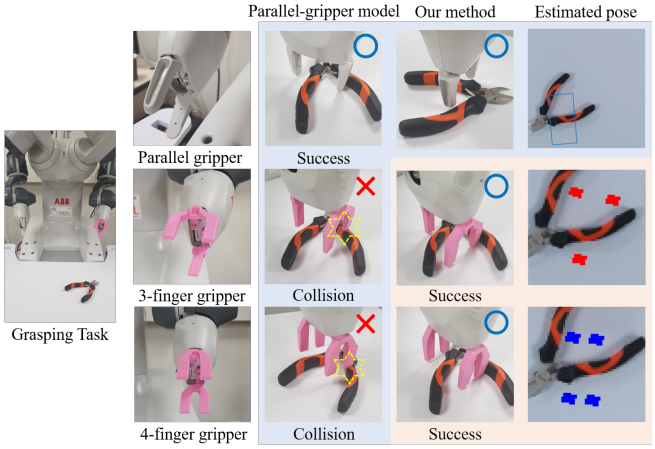


Fig. 1. Illustration of the gripper adaptation task for robot grasping. The previous method GR-ConvNet is robust with parallel grippers but its performance worsens when using complex grippers, even with the same grasp representation. In contrast, our method can successfully adapt to other grippers and grasp challenging objects.

I. Introduction

THE grasping and pick-and-place tasks are essential robotic tasks for industrial and everyday life applications. Many studies on the two tasks have, therefore, been conducted over the years [1], [2], [3]. Recent methods have focused on learning-based approaches [4], [5], [6] and have achieved significant results compared to traditional methods [7], [8]. These grasping detection studies, which can be largely divided into the two categories of 2D planar grasp methods and 3D 6DoF grasp methods [2], mostly consider a parallel gripper, the most basic type of robot end-effector. The parallel gripper uses a simple grasp pose representation consisting of its grasping location, rotation angle, and width, which enables its generalizability to most simple objects. However, the simplicity of the parallel gripper renders it unsuitable for grasping objects of more complex structures. For this reason, some recent studies have tried to solve this issue by using a multi-fingered gripper instead to execute the grasping task. However, such studies require complicated 3D point clouds of the gripper and the target object to find their contact points [9], [10], creating a trade-off between gripper complexity, along with computational cost, and generalizability.

Further studies have tried to work around this trade-off by adapting existing grasp pose representations to new models with different grippers [11]. However, such processes face several limitations. First, for 2D planar methods, the existing grasp representations are designed to simultaneously estimate the grasp locations (i.e., "where to grasp") and the grasping angle-width pairs (i.e., "how to grasp") [5], [6]. Relearning all these tasks to adapt to a specific type of gripper does not solve the issue of high computational cost. Second, several previously proposed grasp representations, such as oriented rectangles, contact points, and oriented arrows, abstract away more precise details of the gripper, such as finger size, finger arrangement, and grasping trajectory. Even if the approximate angle-width prediction of the model is correct, this limitation can lead to a failure case in which a collision occurs as the gripper tries to pick up an object because the model does not take into consideration the specifications of the gripper itself [11].

In this letter, we propose a novel robot-grasping framework to address the problems above. Within this framework, we introduce HybGrasp, a hybrid architecture that combines a

Manuscript received: July 4, 2023; Revised September 20, 2023; Accepted October 15, 2023.

This paper was recommended for publication by Editor Kober Jens upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Technology Innovation Program funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Grant 20007058 and the National Research Foundation of Korea Grant funded by the Korean Government (MSIT) under Grant RS-2023-00208052. (*Corresponding author: Sungho Jo.*)

[1]The authors are with the School of Computing, KAIST, Daejeon 34141, Republic of Korea (e-mail: munjw777@kaist.ac.kr; khangtg@kaist.ac.kr; shjo@kaist.ac.kr.)

Digital Object Identifier (DOI): see top of this page.

gripper-adaptive convolutional neural network and a reinforcement learning model, named GAdaNext and Deep Reinforcement Learning for Refinement (DRLR), respectively, to enable the efficient adaptation to various multi-fingered grippers. GAdaNext uses a "patchify" architecture including multiple downsampling steps to perform angle-width prediction as a classification task. DRLR then refines the output of GAdaNext to increase grasping quality by considering the shapes of the target object and the gripper itself.

The grasping task, in essence, can be divided into two steps: grasp localization and pose estimation [2]. HybGrasp purely focuses on efficiently learning pose estimation with a new gripper, leaving the grasp localization step to a pre-trained, robust parallel gripper model.

To address the training data generation issue, we propose a heuristic function to automatically produce weak grasping labels for training. This heuristic labeling is commonly seen in weak supervision [12], [13]. Our method utilizes the available parallel-gripper datasets [14] to generate adaptive labels for each gripper. In particular, our method crops image patches centered at the given locations of parallel-gripper labels and then estimates the new angle-width labels based on depth information. Our proposed heuristic function efficiently analyzes all possible angle-width candidates of the gripper within the cropped depth patch to select several angle-width candidates as weak labels.

The contributions of this paper are as follows:

- We propose a novel hybrid architecture for 2D planar grasping that efficiently learns to adapt to different gripper designs. Our method only requires simple 2D knowledge of the gripper as input and then predicts an adaptive action via the GAdaNext and DRLR networks.
- To train the proposed network, we generate weak labels of grasping angle and width by using a well-designed heuristic function. To the best of our knowledge, our approach is the first to produce a large annotated dataset with respect to specific grippers using weak supervision.
- We demonstrate the effectiveness of our method with various grippers in both the generated dataset and real-world scenarios. The source code is publicly available.

II. RELATED WORK

A. Learning-based Robot Grasping

Due to the success of deep learning in numerous applications, learning-based robot grasping methods have also gained popularity and have made significant progress. These methods estimate grasp pose in either 2D or 3D space. The 2D grasping methods usually use RGB-D input to predict the planar location, rotation angle, and width of the grasp [4], [5], [6]. Meanwhile, the 3D grasping methods use point cloud information as input to estimate 6DoF grasp poses [15]. Although the 3D representation more resembles real-world scenarios, it requires high computational costs and expensive data generation [16], [17]. For this reason, this paper focuses on the 2D grasping problem.

There exist two main ideas to estimate grasp poses [2]. Some methods are based on object pose estimation [18], while others predict a pixel-level grasp map along with a quality score map to then select the best grasp pose based on the quality scores [5], [6]. Both of these types of approaches have achieved state-of-the-art performance, utilizing robust techniques from well-known object detection [19] or segmentation tasks [20]. However, most of these methods are designed specifically for a single type of gripper. When applied to new grippers, the models, as well as their data generation procedures, need to be adaptively modified. This process is inefficient and inflexible.

B. Adaptive Gripper Systems

Traditional methods tackle the adaptive-gripper problem using optimization models [21], [22]. Finger Splitting [21] estimates an adaptive grasp pose for multi-fingered grippers by initially establishing an optimal parallel grasp and then progressively refining the finger and palm poses. However, this method exhibits limitations concerning its adaptability to various grippers and its dependency on expert human input for object localization and gripper configurations. To address these issues, some learning-based methods are able to learn to adapt to different types of grippers and automatically perform grasp localization [9], [10], [23]. UniGrasp [9] learns a latent feature from a gripper point cloud and an object point cloud. Then, it uses a point selection network to sequentially generate a contact point for each gripper finger. However, this contact-based approach requires the input to be of full 3D geometry, instead of partial observations from RGB-D input. To address this problem, AdaGrasp [10] proposed computing cross-convolutions between gripper features and scene features to learn grasp scores in 3D space in which each voxel measures the probability of successful grasping at a specific pixel with a specific gripper orientation. Although AdaGrasp can handle RGB-D inputs and generalize well to various grippers, it requires specific 3D point clouds of the grippers, which we wanted to avoid to improve efficiency. Moreover, AdaGrasp performs searching in a high-dimensional space of the 3D coordinates of possible grasping locations and the possible rotation angles. Therefore, it suffers from a computational burden when predicting grasp scores for all possible actions.

C. Grasping Data Generation

The Cornell dataset [24] is the first grasping dataset to have been collected via real robot experiments. However, due to limited time and resources, this dataset is quite small and is therefore not sufficient for training deep neural networks. Existing large-scale grasping datasets are generated through complicated and expensive simulations [4], [14], [16] in which scenes and grasping actions are randomly sampled to label successful actions. These steps can be applied to both 2D and 3D grasping approaches.

In this letter, we propose an efficient and methodical approach to generate grasping labels for different kinds of grippers. Inspired by weak supervision [12], [13], we design some heuristic rules to quickly estimate a successful grasping score for each grasp candidate. Given a gripper mask and an

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

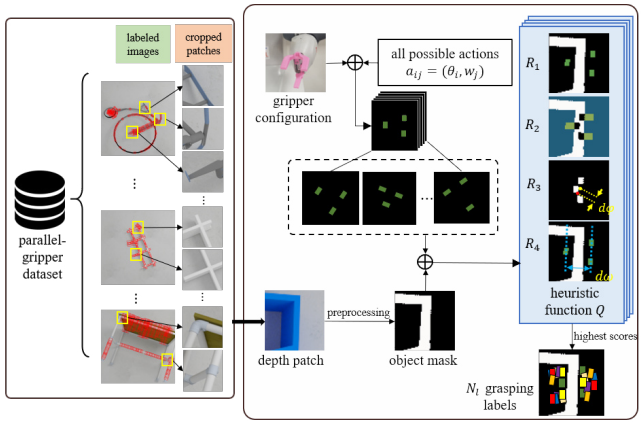


Fig. 2. Training data generation process. This method utilizes a dataset with known parallel-grasping labels to generate adaptive labels for other grippers. For each parallel-grasping location, we first crop an RGB-D patch centered at that location and then estimate the weak labels of the new gripper's angle and width based on a heuristic function Q.

object mask extracted from the depth input, the heuristic functions analyze spatial constraints between the gripper mask and object mask to determine the goodness of the grasp. Although the heuristics cannot cover all cases within the dataset, we find that the produced labeled data is still reasonably accurate to train a robust classification network.

III. TRAINING DATA GENERATION

A. Data Labeling Process

Given an image I, a grasp g can be generally represented by three basic parameters: the grasping location p, the gripper's rotation θ and the gripper's width w [24], [25].

$$g = \{p, \theta, w\} \tag{1}$$

where p is the pixel coordinate in image $I, \theta \in [-\pi, \pi]$, and $w \in [w_{min}, w_{max}]$. This simple representation can be applied to many multi-fingered grippers for the 2D grasping problem. To generate gripper-adaptive grasping labels for training, we utilize the Jacquard parallel-gripper dataset[14]. Our main idea is to fix the grasping position p and then determine the new width and rotation angle to adapt to the new multi-fingered gripper. Fig. 2 presents our proposed adaptive labeling method. For each image with known parallel grasping labels, we first crop image patches centered at the parallel-grasping locations and then estimate the adaptive gripper's rotation and width label for each patch. To do this, for each cropped patch, we assess all possible candidates of gripper rotation and width by introducing four heuristic principles, which are based on the relative position of the gripper mask to the object mask, to check whether the angle-width candidate satisfies a successful grasp or not. Note that because the grasping positions are always fixed at the centers of image patches, we only need to discretize the ranges of the rotation angle and width to initialize all candidates. Finally, after determining the quality scores for all grasping candidates using the proposed heuristic function, our method selects N_l(=10) grasps with the highest scores and considers them as training labels.

B. Heuristic Labeling Function

This section describes the heuristic function for the aforementioned labeling process in detail. The function takes an object mask M_o, which is extracted from the RGB-D image patches, and a gripper pose candidate as inputs. It then mimics a grasping action a to validate the grasp pose. The success of the action can be measured via the four heuristic principles as follows:

1) When the gripper is in its initial state, there should be no collision between the gripper and the object mask.
2) When the gripper performs grasping and ends up in the closed state (width = 0), there should be an overlap between the gripper and the object mask.
3) To encourage stable grasping, the centroid of the overlapping region in 2) should be close to the center of the input mask.
4) Among multiple feasible estimates of the gripper width, the smallest one should be preferred.

To model the principles above, we designed the four reward functions R_1, R_2, R_3, R_4. Given the object mask M_o and the gripper pose a, the four functions are

$$\begin{aligned} R_1(a, M_o) &= \begin{cases} +r_1 \; if \; not \; collision \\ -r_1 \; if \; collision \end{cases} \\ R_2\,(a, M_o) &= |a \cap M_o| \\ R_3\,(a, M_o) &= -\left\| ctr\,(a \cap M_o) - c_M \right\|^2 \\ R_4\,(a, M_o) &= -a_{width} \end{aligned} \tag{2}$$

where $|.|$ is the number of pixels, $crt\,(a \cap M_o)$ is the center coordinate of overlapping region $a \cap M_o$, c_M is the center coordinate of input mask M_o and a_{width} is the width of gripper pose a. Finally, we can define our heuristic function used for labeling as follows:

$$Q\,(a, M_o) = \lambda_1 R_1 + \lambda_2 R_2 + \lambda_3 R_3 + \lambda_4 R_4 \tag{3}$$

where $\lambda_{1 \to 4}$ are weighted hyper-parameters.

IV. PROPOSED GRIPPER-ADAPTIVE SYSTEM

In this section, we propose a novel robot grasping method to adapt to multiple grippers. First, the method detects the grasping location from the input image (Fig.3A). It then crops a patch centered at the estimated location. By fixing the center of the gripper at the center of the cropped patch, our method then estimates the angle rotation and width of the gripper (Fig.3B). Finally, the estimated result is refined within the gripper pose searching space to increase grasp quality (Fig.3C).

A. Grasp Localization

The main goal of this step is to localize the object and then generate potential grasping positions, without any knowledge about the gripper. Therefore, to find the graspable locations, we simply utilize an existing parallel grasping model, GR-ConvNet [6]. This network has an Unet-like model, which is used for the pixel-wise prediction task. After detecting the grasping location, the next step will perform the adaptation to a specific type of gripper.

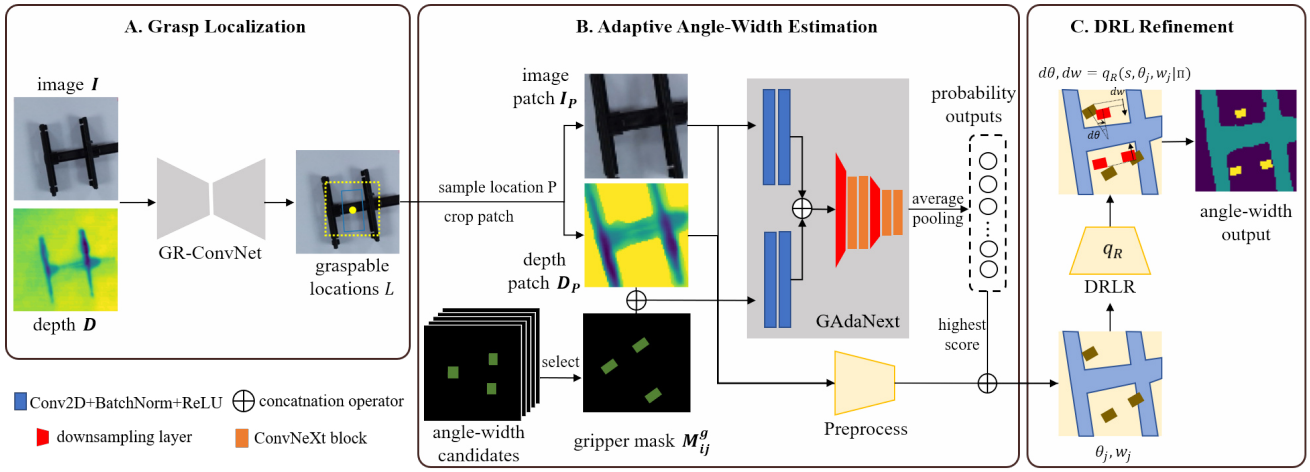


Fig. 3. The proposed gripper-adaptive grasping hybrid Architecture HybGrasp. First, it applies the parallel-grasping model, GR-ConvNet, to detect all graspable locations from the RGB-D image (Section A). After that, the system samples the best suitable location P and crops the image I_P and depth patch D_P. Based on these inputs, the first part of our novel hybrid architecture, GAdaNext, estimates a probabilistic score for each pair of angle-width candidate (Section B) which is selected with the highest score. The second part, DRLR, refines the predicted output to increase a grasp quality to fulfill robust gripper adaptation (Section C).

B. Adaptive Angle-Width Estimation

Given the cropped patches, including the color image I_P and depth D_P, along with simple knowledge of the gripper, we propose a novel approach to estimate the gripper's angle and width. Instead of regressing the outputs directly from the inputs, our method formulates the problem as a binary classification task. In particular, we generate a set of angle-width pairs and then try to categorize the solutions into a positive class (label "1") and the remaining candidates into a negative class (label "0"). Let $\{\theta_0, \theta_1, ..., \theta_M\}$ and $\{w_0, w_1, ..., w_N\}$ be the sampled angle and width candidates respectively, where $\theta_i \in [0, 360]$ and $w_j \in [w_{min}, w_{max}]$. For each angle-width pair (θ_i, w_j), we can draw a gripper mask M^g_{ij} that indicates the pose of the gripper using the known configurations of the gripper (the number of fingers and size of each finger). As a result, our network uses the gripper mask M^g_{ij} and the RGB-D image patches I_P, D_P to predict whether this gripper mask is a solution ("1") or not ("0"). To train our network, we label the grasp poses generated in section III as "1" and the other candidates as "0". Fig.3 illustrates the detailed architecture of our proposed network, GAdaNext. In contrast to existing ResNet-based methods [4], [5], [6], which employ a standard convolution block comprising Conv2D, BatchNorm, and ReLU layers, we propose to use the ConvNeXt block [26]. This block includes Conv2D, LayerNorm, and GELU layers. The structure of the ConvNeXt block is depicted in Fig. 4a.

C. Deep Reinforcement Learning for Refinement

Although GAdaNext estimates new multi-finger gripper configurations, the searching space of gripper configurations is limited within ranges of $\{\theta_0, \theta_1, ..., \theta_M\}$ and $\{w_0, w_1, ..., w_N\}$ as previously described. To compensate for this limitation, the Deep Reinforcement Learning for Refinement (DRLR) model is combined with GAdaNext in this grasp system. DRLR is a PPO-based algorithm [27], which is used in many domains. In general, since the sample poses created by GAdaNext are close to high-quality poses, the sample's quality can be dramatically increased with little calibration effort by DRLR. We define a calibration attempt from receiving object mask M_o, which is a binary contour mask preprocessed with RGB-D and action a_0 from GAdaNext to finishing grasping as one episode. In the environment, the agent can take four discrete actions: turning the gripper either clockwise or counterclockwise, or opening and closing the gripper: $\hat{a} \in \{+d\theta, -d\theta, +dw, -dw\}$. In one episode, the policy model makes a sequence of actions $\{\hat{a}_0, \hat{a}_1, ..., \hat{a}_T\} = \pi(\hat{a}|s, \mu)$ with network parameter μ and the environment responses a simulated trajectory $\{s_0, s_1, ..., s_T\}$. The state s_t at time t consists of the 3 sequential frames of the previous trajectory, $s_t = \{\rho_{t-2}, \rho_{t-1}, \rho_t\}$, as in previous DRL research [28], and each frame ρ_t is composed of the sum of the object mask M_o and gripper mask $M^g_{\hat{a}}$ with action $\hat{a}_t$ (initial state ρ_0 comes from GAdaNext), $\rho_t = M_o \cap M^g_{\hat{a}_t}$. This process can therefore be described as:

$$s_{i+1} = p(s_i, \hat{a}_i), \; s \in S \subset \mathbb{R}^{64\times64\times3} \tag{4}$$

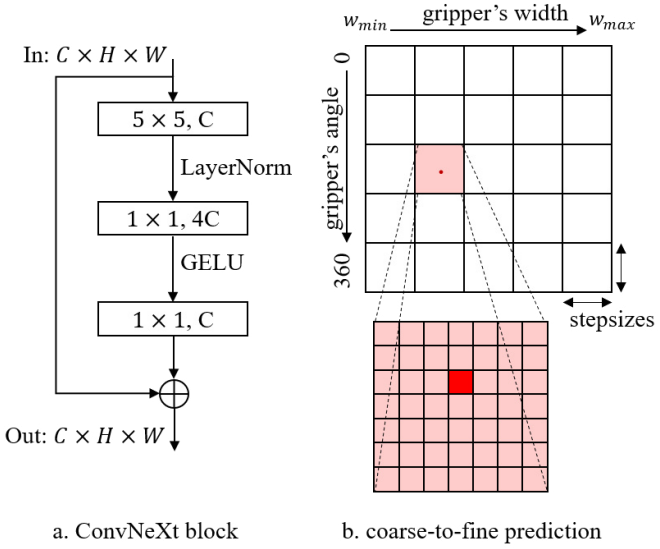
where p is a basis simulation process function and i indicates the i-th time step. The immediate reward is determined using the aforementioned quality equation (3-x quality equation) and the value function:

$$v_i = \sum_{t=i}^{T-1} \left(\gamma^{t-i} r_t\right) + \gamma^{T-i} V(s_T|\xi) \tag{5}$$

where γ is the discount factor, T is the last time step of an episode, ξ is the parameter of the modeled value function, and r_i and V represent the reward and the approximated value function, respectively. Based on this function, we can describe one episode's trajectory with the accumulated sum of immediate rewards using the following equation:

$$R(\tau) = \sum_{t=0}^{T} v_i \tag{6}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



a. ConvNeXt block b. coarse-to-fine prediction

Fig. 4. An illustration for a) the details of ConvNeXt block used in our network, and b) the coarse-to-fine strategy for angle-width estimation.

where τ is the trajectory of one episode, representing a series of states and actions $\tau = \{s_0, \alpha_0, s_1, \alpha_1, ..., s_T, \alpha_T\}$ where α is an action variable in the $\{\theta, w\}$ form at each step.

D. Efficient Training and Prediction

Training. We adopt a negative sampling strategy for training to improve the run-time and memory footprint. Therefore, instead of training all possible angle-width candidates, our method randomly samples a subset of negative candidates to train along with the positive ones. We denote S^{pos} and S^{neg} as the positive and negative set respectively. The loss function now can be computed by using binary cross-entropy loss as follows:

$$L = \frac{1}{|S^{pos}|} \sum_{x \in S^{pos}} log f\left(I_p, D_p, M_x^g\right) + \frac{1}{|S^{neg}|} \sum_{y \in S^{neg}} log\left(1 - f\left(I_p, D_p, M_y^g\right)\right) \quad (7)$$

where $f(.)$ is the proposed network GAdaNext, and I_p, D_p, M^g are the color patch, depth patch, and the grasping candidate mask respectively.

Prediction. To accelerate model prediction, we apply a coarse-to-fine strategy. First, we divide the angle-width space into a grid with low resolution using large step sizes (see Fig. 4b). By only considering the smaller number of candidates, the model can quickly produce a coarse estimate of the gripper's angle and width. Next, our method narrows the step sizes and generates a new set of candidates around the coarse estimation to estimate a finer solution. As a result, our method can produce an accurate output angle and width in an efficient way.

V. EXPERIMENTAL RESULTS

A. Experimental Setups and Learning Details.

Setting. All experiments were performed in an NVIDIA Titan V and Intel Core i9-10900KF CPU environment. The proposed architecture was built in the Pytorch framework. Regarding the gripper, the HybGrasp is designed to adapt diverse gripper types and accurately capture their configurations. We tested it on five different grippers (see Fig 5), validating its effectiveness for real-world use.

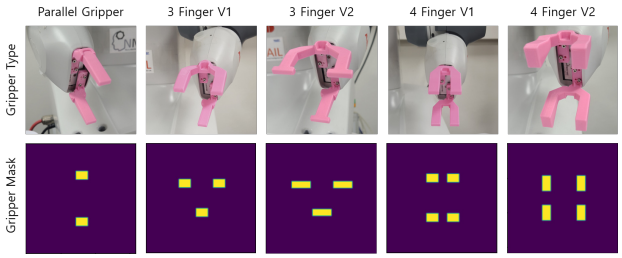


Fig. 5. Test grippers, varying in number of fingers and size features, were chosen to validate the model. The above figure shows these five grippers and their training masks.

Data processing and generating. Training data came from the Jacquard dataset [14], which includes 54K images and 1.1M grasp labels. As outlined in Section III, this data was used to generate 64x64 patches and adaptive grasp labels (see Fig. 2). The dataset was split 9:1 for training and testing.

CNN network training details. The proposed CNN model, GAdaNext was trained by using the Adam optimizer with an initial learning rate of 0.0002. The batch size was set to 128, and the training was conducted over 40 epochs. For the gripper configurations, we set the range value of the gripper angle to $[0^o, 360^o]$ and the gripper width to [9, 49]. We then sampled M angle and N width candidates inside these range values using a step size of 10^o and 4. For evaluation, we first performed a coarse prediction using the step sizes of $(20^o, 8)$, then refined the coarse results at a finer resolution with the step sizes of $(5^o, 2)$.

Reinforcement learning training details. The DRLR model was trained for the partial refinement of the predicted grasp pose on the same device and environment used for the training of GAdaNext. The reinforcement learning framework was implemented using the stable baseline3 [29] library and the simulation was built using the Gym API [30], both of which are widely adopted in the DRL field. The learning process was carried out in 500 parallel environments with a temporal upper limit of 100-time steps per episode. Batch size and learning rate were set to 32 and 0.00006 relatively, and the training process lasted in a total of 10^8 episodes. To preprocess the input patches, the TRACER model[31] was employed as a SOD model. This training took a total of 4 days, 5 hours, 23 minutes, and 33 seconds, and consumed 1.193GB of GPU memory.

B. Experiment on Jacquard

Evaluation metrics. To evaluate the validity of the model proposed in this paper, we first analyzed it with the Jacquard

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE I
JACQUARD RESULTS

Algorithm	2-Finger Gripper		3-Finger V1		3-Finger V2		4-Finger V1		4-Finger V2	
	SR	Latency	SR	Latency	SR	Latency	SR	Latency	SR	Latency
Rule base	-	1927ms	-	2215ms	-	2232ms	-	2450ms	-	2457ms
Only GAdaNext	74.5%	25.1ms	76.1%	28.08ms	73.6%	26.3ms	70%	24.74ms	67.1%	26.4ms
Only DRLR	73.3%	101.5ms	64.7%	103.7ms	58.9%	118.1ms	66.4%	120.5ms	40.7%	129.3ms
GAdaNext + DRLR	**89.6%**	127.0ms	**90.34%**	131.7ms	**88.3%**	144.7ms	**85.6%**	153.7ms	**89.4%**	157.0ms

* SR: Success Rate

public dataset. The HybGrasp model was tested with the generated dataset, which contains various objects. The success of grasping was determined by the following criteria:

1) Collision check: The gripper mask, obtained with the grasping angle and width predicted by the HybGrasp model, does not overlap with the object mask.
2) Grasping check: When reducing the width of the gripper to perform grasping, the overlapped cross-section between the object and gripper is feasible.

Results. The experimental results are documented in Table I, which provides a summary of both success rate and execution time. To provide a comparative basis, we tested a rule-based method that evaluates all possible grasping poses, where each grasping angle and width were sampled using the step sizes of 1^o and 20 units, respectively. This inclusion of the rule-based method facilitates a comparison of the time savings achieved in the robot's real-time performance when using the learned network model. Additionally, Table I provides the results when using the DNN-based GAdaNext and the DRL-based DRLR model independently. These results are used to compare with our model. HybGrasp achieves the highest accuracy for each gripper type and a latency performance about 16 times faster than that of the Rule-based cases.

C. Simulation Experiment

Evaluation metrics. In this simulation experiment, the effectiveness of the algorithm is assessed using the grasp success rate, computed as the ratio of successful grasps to the total number of grasp attempts. Evaluations are performed individually for each type of gripper, and the average performance is considered.

Experiment setup. In our simulation setting, we use Pybullet for the environment. The test objects used in the simulation come from the Dexnet 2.0 dataset[4] and are randomly placed within a designated rectangular area. For a comparative study, we decided to compare the performance of our HybGrasp model with the recent state-of-the-art adaptive-gripper system, AdaGrasp[10]. However, due to fundamental differences between the two systems, we maintained fairness in this comparison by following a standardized procedure. We converted each 3D gripper point cloud into a 2D gripper mask, which served as the input for both HybGrasp and AdaGrasp. While training the AdaGrasp model with these 2D gripper inputs, we adhered to its original training setups. In contrast, our HybGrasp model did not require any fine-tuning with the simulation data.

Results. The results of this comparison are concisely summarized in Table II. As depicted in Table II, our HybGrasp method achieved a competitive success rate in comparison to AdaGrasp, while demonstrating significantly faster run-time (approximately 5 times faster). These findings underscore the effectiveness and efficiency of our approach.

TABLE II
SIMULATION RESULTS

Gripper Type	AdaGrasp-2D Input*		HybGrasp	
	SR(%)	Time(ms)	SR(%)	Time(ms)
Barrett Hand 2F	77	1809	**87**	**361**
Robotiq 2F-85	**88**	1826	86	**319**
Barrett Hand	**89**	1818	86	**343**
Barrett Hand-B	**81**	1822	79	**359**
Overall	83.8	1819	**84.5**	**346**

* 2D Input: Converting 3D gripper clouds to 2D masks as inputs.

D. Real World Experiment

Evaluation metrics. In this experiment, we evaluated the proposed architecture by conducting several comparative experiments with a physical robot in a real-world environment. The experiments were designed to investigate the performance of the system under different conditions, including the types of test objects, models, and applied grippers. We defined a successful grasp to be one that satisfied the following three conditions:

1) No collision occurs while attempting to pick up an object using the predicted grasping pose.
2) The object is securely held by the gripper.
3) As the object is transported to the collection point, it must be lifted to at least 20 cm and must remain within the grasp of the gripper at all times.

Experiment setup. The manipulator used in this experiment was an IRB 14000 Yumi robot with 7-DOF dual-arms manufactured by ABB. Microsoft's Azure Kinect was used as the RGB-D camera and it was mounted above the center of the robot. Regarding the gripper setting, five types of grippers were created using a 3D printer and mounted onto the robotic system. The robot system is illustrated in Fig. 6.

Test objects. For real-world testing, we prepared experimental objects from three primary categories. The first category consists of household objects that are commonly found in everyday life. The second category comprises tools or equipment used in the workplace. Lastly, the third category

TABLE III
REAL WORLD EXPERIMENT RESULTS

Gripper Type	Approach	Household Obj		Tool-like Obj		3D Adversarial Obj		Total SR	
		SR	CFR	SR	CFR	SR	CFR	SR	CFR
2-Finger Gripper	PM [6]	86.0%(43/50)	42.9%	66.0%(33/50)	94.1%	58.8%(47/80)	75.0%	68.8%(124/180)	76.8%
	Ours	90.0%(45/50)	0%	88.0%(44/50)	33.3%	87.5%(70/80)	20.0%	**88.3%(159/180)**	**19.0%**
3-Finger V1 (Asymmetric)	PM [6]	88.0%(44/50)	33.3%	52.0%(26/50)	91.7%	50.0%(40/80)	90.0%	61.1%(110/180)	85.7%
	Ours	94.0%(47/50)	0%	96.0%(48/50)	0%	83.7%(67/80)	7.7%	**90.0%(162/180)**	**5.6%**
3-Finger V2 (Asymmetric)	PM [6]	82.0%(41/50)	33.3%	42.0%(21/50)	86.2%	33.8%(27/80)	69.8%	49.4%(89/180)	71.4%
	Ours	90.0%(45/50)	0%	82.0%(41/50)	44.4%	76.3%(61/80)	36.8%	**81.7%(147/180)**	**33.3%**
4-Finger V1 (Symmetric)	PM [6]	84.0%(42/50)	25.0%	66.0%(33/50)	88.2%	46.3%(37/80)	81.4%	62.2%(112/180)	76.5%
	Ours	96.0%(48/50)	0%	88.0%(44/50)	16.7%	90.0%(72/80)	50.0%	**91.1%(164/180)**	**31.3%**
4-Finger V2 (Symmetric)	PM [6]	72.0%(36/50)	71.4%	70.0%(35/50)	93.3%	43.8%(35/80)	82.2%	58.9%(106/180)	82.4%
	Ours	92.0%(46/50)	25.0%	92.0%(46/50)	50.0%	76.3%(61/80)	42.1%	**85.0%(153/180)**	**40.7%**

* Objects in categories 1 and 2 had 5 trials each, and category 3 had 10, all with random orientations and positions.

* **CFR(Collision Fail Rate)**: The proportion of failure cases due to gripper entry collisions among total failure cases.

* **PM**: Parallel Gripper Model [6] / * **SR**: Success Rate.

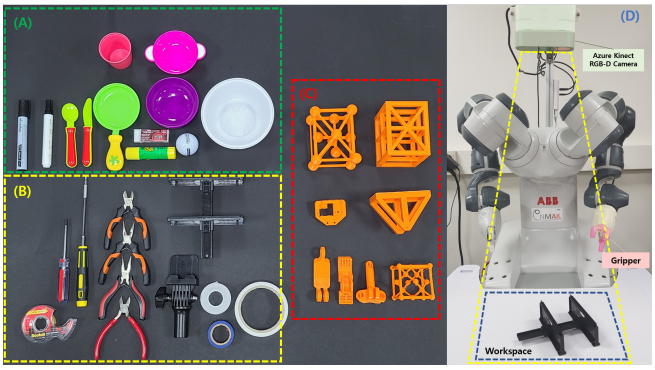


Fig. 6. Real world setting. The experimental objects (A), (B), and (C) represent household, tool-like, and 3D-printed adversarial parts, respectively. The experiments were performed under the same robot and camera settings as in (D).

comprises 3D-printed adversarial objects, intended to evaluate the model's reliability when dealing with objects possessing complex geometries. These test objects are visually depicted in Fig. 6.

Model validation process. For the real-world experiment, the pick-and-place results of the proposed HybGrasp model were compared with the results of GR-ConvNet[6], which is a state-of-the-art parallel-gripper method using a rectangle representation of grasp pose as our baseline. For the test objects, a total of 180 grasping attempts were conducted for each gripper type and the success rate was evaluated based on the aforementioned pre-defined criteria. There were various scenarios of failure that occurred during the pick and place task, such as grasping in the air at an incorrectly predicted location or an object slipping and falling during lifting. In particular, there also exist scenarios in which the gripper and the object collide as the gripper moves along the z-axis from the predicted pose to grab the object, causing a failure to grip. The collision rate, or the ratio of such collision cases out of all failure cases, was also documented in Table III.

Results and failure analysis. Through the experiment, several analyses were obtained. Even with a change in gripper shape, the existing parallel-based model demonstrates a success rate above a certain threshold for grasping cylindrical or box-shaped objects that are of relatively simple shape. However, as the geometric complexity of the object increased, the success rate of the parallel model decreased significantly. Moreover, the failure cases of our proposed model were mostly due to unstable grasping points or slipping, while the parallel model had a higher incidence of collision failures, accounting for the majority of the failure cases, as shown in Table III. In contrast, our proposed model increased the overall success rate by utilizing a grasp pose that was refined with knowledge of the gripper configurations to greatly reduce the collision rate between the gripper and the object during grasping attempts. Exemplary success cases of our model are shown in Fig. 7.

VI. CONCLUSION AND FUTURE WORKS

In this letter, we introduce HybGrasp, a novel hybrid architecture that can efficiently adapt to different gripper designs. We demonstrate the effectiveness of HybGrasp with various grippers in diverse experiments. Our results show that the proposed approach significantly improves grasping performance, particularly through collision avoidance. However, HybGrasp has certain limitations. Three main limitations can be identified: (1) the grasping angle-width information used in the grasp pose refinement process of the DRLR model is extracted from the camera depth data and this process is highly sensitive and noisy, leading to unstable predictions, and (2) fundamentally, the location pose information is dependent on the performance of the existing parallel model. This is an advantage when sharing rectangle-based datasets for learning, but it can also cause various unstable failures from the perspective of grasping task performance. (3) Despite the advances, our grasp representation, angle/width, cannot fully define more complex grippers such as Allegro Hand or Shadow Hand. In the future, we believe that it is possible to improve

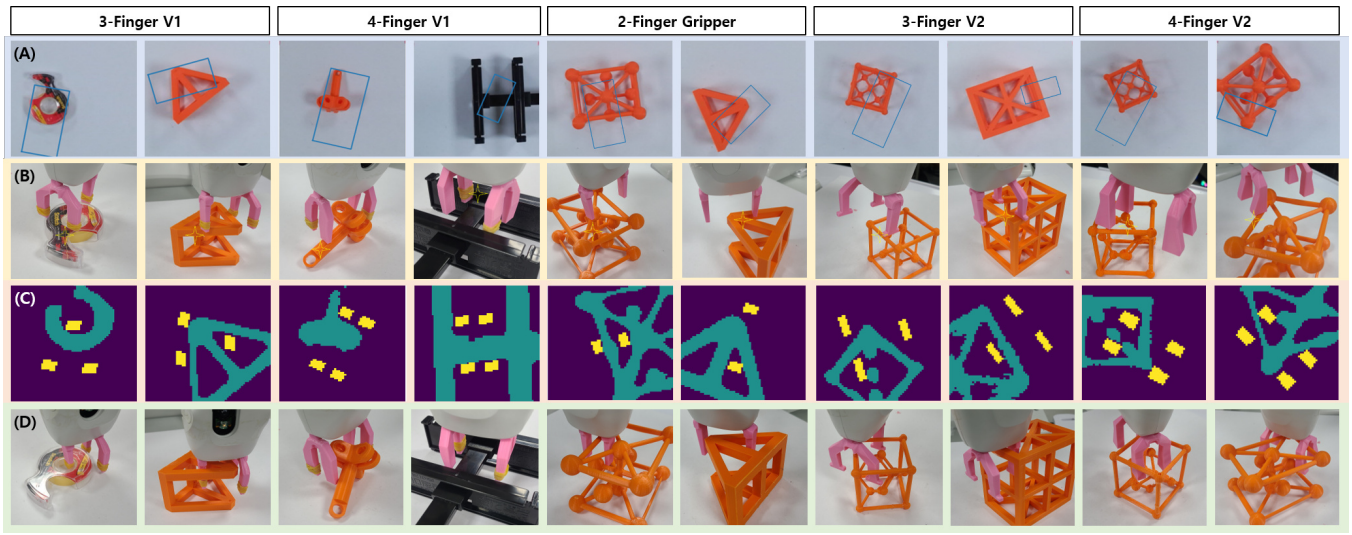


Fig. 7. Experiment success cases. The baseline model generates poses with a rectangular representation, as seen in row A. When executing a grasp with these generated poses using a new gripper, collisions occur due to the differences in gripper configuration, as seen in row B. In contrast, HybGrasp adapts to the new gripper and calibrates the poses generated by the baseline model, enabling the robot system to grasp successfully as seen in rows C and D.

these limitations and expect more robust performance for the dexterous grasping task.

REFERENCES

[1] Y. Cong, R. Chen, B. Ma, H. Liu, D. Hou, and C. Yang, "A comprehensive study of 3-d vision-based robot manipulation," *IEEE Transactions on Cybernetics*, 2021.

[2] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[3] Q. M. Marwan, S. C. Chua, and L. C. Kwek, "Comprehensive review on reaching and grasping of objects in robotics," *Robotica*, vol. 39, no. 10, pp. 1849–1882, 2021.

[4] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[5] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.

[6] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in *IROS*. IEEE, 2020, pp. 9626–9633.

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

[8] A. Bicchi, "On the closure properties of robotic grasping," *The International Journal of Robotics Research*, vol. 14, no. 4, pp. 319–334, 1995.

[9] 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 RA-L*, vol. 5, no. 2, pp. 2286–2293, 2020.

[10] Z. Xu, B. Qi, S. Agrawal, and S. Song, "Adagrasp: Learning an adaptive gripper-aware grasping policy," in *ICRA*. IEEE, 2021, pp. 4620–4626.

[11] D. Wang, C. Liu, F. Chang, N. Li, and G. Li, "High-performance pixel-level grasp detection based on adaptive grasping and grasp-aware network," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 11, pp. 11 611–11 621, 2021.

[12] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," *NeurIPS*, 2016.

[13] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf, "Fidelity-weighted learning," *arXiv preprint arXiv:1711.02799*, 2017.

[14] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IROS*, 2018, pp. 3511–3516.

[15] P. Ni, W. Zhang, X. Zhu, and Q. Cao, "Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds," in *ICRA*. IEEE, 2020, pp. 3619–3625.

[16] 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 *ICRA*. IEEE, 2018, pp. 3766–3773.

[17] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *CVPR*, 2020, pp. 11 444–11 453.

[18] W. Chen, X. Jia, H. J. Chang, J. Duan, and A. Leonardis, "G2l-net: Global to local network for real-time 6d pose estimation with embedding vector features," in *CVPR*, 2020, pp. 4233–4242.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.

[20] L. Han, T. Zheng, L. Xu, and L. Fang, "Occuseg: Occupancy-aware 3d instance segmentation," in *CVPR*, 2020, pp. 2940–2949.

[21] Y. Fan, T. Tang, H.-C. Lin, and M. Tomizuka, "Real-time grasp planning for multi-fingered hands by finger splitting," in *IROS*. IEEE, 2018, pp. 4045–4052.

[22] Y. Fan, X. Zhu, and M. Tomizuka, "Optimization model for planning precision grasps with multi-fingered hands," in *IROS*. IEEE, 2019, pp. 1548–1554.

[23] N. Khargonkar, N. Song, Z. Xu, B. Prabhakaran, and Y. Xiang, "Neuralgrasps: Learning implicit representations for grasps of multiple robotic hands," in *Conference on Robot Learning*. PMLR, 2023, pp. 516–526.

[24] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *ICRA*. IEEE, 2011, pp. 3304–3311.

[25] Y. Domae, H. Okuda, Y. Taguchi, K. Sumi, and T. Hirai, "Fast graspability evaluation on single depth maps for bin picking with general grippers," in *ICRA*. IEEE, 2014, pp. 1997–2004.

[26] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *CVPR*, 2022, pp. 11 976–11 986.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[29] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," 2019.

[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[31] M. S. Lee, W. Shin, and S. W. Han, "Tracer: Extreme attention guided salient object tracing network (student abstract)," in *AAAI*, vol. 36, no. 11, 2022, pp. 12 993–12 994.