2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

6-DoF Closed-Loop Grasping with Reinforcement Learning

Sverre Herland[1], Kerstin Bach[1], and Ekrem Misimi[2]

***Abstract*— We present a novel vision-based, 6-DoF grasping framework based on Deep Reinforcement Learning (DRL) that is capable of directly synthesizing continuous 6-DoF actions in cartesian space. Our proposed approach uses visual observations from an eye-in-hand RGB-D camera, and we mitigate the sim-to-real gap with a combination of domain randomization, image augmentation, and segmentation tools. Our method consists of an off-policy, maximum-entropy, Actor-Critic algorithm that learns a policy from a binary reward and a few simulated example grasps. It does not need any real-world grasping examples, is trained completely in simulation, and is deployed directly to the real world without any fine-tuning. The efficacy of our approach is demonstrated in simulation and experimentally validated in the real world on 6-DoF grasping tasks, achieving state-of-the-art results of an 86% mean zero-shot success rate on previously unseen objects, an 85% mean zero-shot success rate on a class of previously unseen adversarial objects, and a 74.3% mean zero-shot success rate on a class of previously unseen, challenging "6-DoF" objects.**

Raw footage of real-world validation can be found at https://youtu.be/bwPf8Imvook

I. INTRODUCTION

Grasping an arbitrary object in one's vicinity is a trivial human skill, yet an essential part of solving robotic manipulation. The full pipeline involves perceiving the world, identifying what and how to grasp, planning, and executing the motion with the robot. In a static and observable environment, this can be done once and executed in an open loop. However, if the world is dynamic or only partially observable, a closed-loop solution may be more appropriate.

Grasping problems in the literature can generally be categorized based on the number of degrees of freedom (DoFs) that are allocated to the gripper. The most prevalent formulation is 4-DoF grasping [1], [2], [3], [4], [5], [6]. In this scenario, the gripper is typically oriented downwards. Three of the DoFs are used to control its 3D position, while the fourth controls the yaw - the angle around the downward direction. A less explored but more powerful formulation is 6DoF grasping [7], [8], [9], [10], [11]. In this approach, the gripper has the freedom to assume any position in 3D space and can rotate around all axes. This freedom provides the robot with a broader set of potential grasps, thus creating a more versatile system since 6-DoF grasps can generalize better to arbitrary grasping poses and align to the graspable part of the objects better than in the case of 4-DoF grasping [12]. However, the added flexibility also poses a larger search space, complicating the control problem.

In recent years, there's been a noticeable shift towards incorporating Deep Learning into the grasping pipeline. A common approach involves replacing the first two components, computer vision and grasp synthesis, with deep neural networks [1], [2], [9], [13]. The resulting grasps are then fed into a conventional planning algorithm to generate robot motion that satisfies the grasping pose. However, other works aim to replace the entire pipeline and learn it end-to-end, typically through DRL [5], [3], [11], [13].

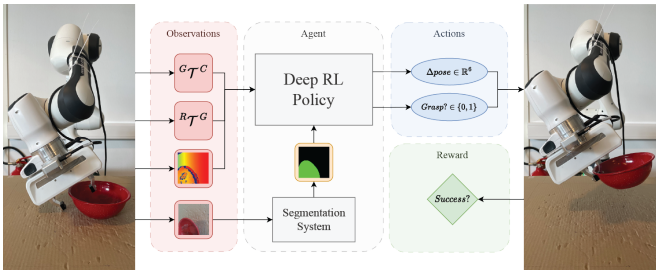


Fig. 1. Our proposed 6-DoF grasping pipeline. The RL-agent is presented with a configurable gripper-to-camera pose, the current end-effector pose, a depth image, and an RGB image. To locate the object, a dedicated module transforms the RGB image into a segmentation mask. The RL agent generates desired $\mathbb{R}^6$ changes in end-effector pose and decides when to close the gripper. A binary reward is given to successful episodes.

Reinforcement Learning (RL) offers a toolkit for learning control from experience [14]. At its core, RL is centered on discovering a policy (controller) that selects actions to optimize an arbitrary reward signal. What sets RL apart is its self-reliance, the algorithm actively samples its own training data to learn the controller autonomously. Additionally, it allows high-level optimization objectives, which gives it a unique advantage as a control method [15].

The restricted supervision of RL is a double-edged sword. For grasping tasks, a natural minimal reward is a binary variable indicating success. However, in the vast space of potential trajectories, successful ones are rare, making it hard to learn. One solution is to *shape* the reward to provide a denser learning signal. For instance, [3] formulates a reward based on how close the gripper is to a set of predefined grasp poses. Others report learning of grasping from a simple binary reward [5], [4], though not with a 6-DoF action space.

RL is famously data-intensive [16]. Given enough resources, complex learning problems can be solved from real-world interactions [4]. However, due to cost and safety, it is often preferable to carry out training in simulation [3]. Since a simulated and physical setup may differ, this introduces a new problem, ensuring that the trained agent can generalize away the differences between the two domains.

A similar grasping problem is explored by [1]. They use a Convolutional Neural Network (CNN) to map images

[1]NTNU IDI, Norway; [2]SINTEF Ocean, Norway

from a wrist-mounted depth camera to candidate grasps and associated confidence scores in a single network evaluation. This one-pass approach results in very high throughput, allowing their grasp-synthesis pipeline to run in closed-loop at 50Hz. However, the CNN is trained with supervised learning and requires a large dataset of pre-labeled grasps, and their pipeline is restricted only to 4-DoF grasping problem.

A method for 6-DoF grasping is presented in [8]. Using a large set of labeled grasps, they train a Variational Auto Encoder (VAE) to approximate the posterior distribution of successful 6-DoF grasps given a point cloud observation. However, similarly to related works [7], [9], their focus is mainly on the first two components of the grasping pipeline; perception and grasp prediction. Controlling the robot to realize a grasp is abstracted away, and the VAE is trained supervised and executed in open-loop. In contrast, we are interested in using RL to get a trajectory-level policy with a minimal level of supervision.

Another study [10] uses RL to learn a closed-loop, 6-DoF controller. Using a handheld gripper with a mounted camera that mirrors their robotic end-effector, they collect several thousand human demonstrations. The data is used to learn an action-value function, which is later fine-tuned on a physical robot. However, while their gripper is technically endowed with 6-DoF motion, their action space is not. Using heuristics, they generate a finite set of 35 action candidates and use the learned Q-function to select the most promising. Our approach differs because we train a policy capable of directly synthesizing full $\mathbb{R}^6$ 6-DoF actions. Additionally, we aim to limit the dependency on real-world data/demonstrations and do zero-shot transfer from simulation.

We present a vision-based, eye-in-hand, 6-DoF closed-loop grasping framework with reinforcement learning that brings the following specific contributions:

- An approach that does not need any real-world manipulation examples and that is trained completely in a simulated environment.
- An RL-based framework that learns the controller with minimal reliance on external supervision.
- A policy capable of directly synthetizing full $\mathbb{R}^6$ 6-DoF actions.
- An approach that transfers to the real world without supplementary training
- An 86% mean zero-shot success rate when grasping previously unseen objects in real-world validation.

II. PROBLEM DESCRIPTION

We consider a setting where a robot grasps an object from a table. Using only robot state readings and images from a camera, a learning agent is tasked with controlling the end-effector through cartesian space and deciding when to close the gripper. We model this problem as a Partially Observable Markov Decision Process (POMDP) [14]. At each point in time t, the agent finds itself in state s_t and must pick an action a_t, after which it will receive reward r_t and be in state s_{t+1}. The states s_t, the dynamics governing them, and the rewards are not known to the agent. Instead, it has to rely on a stream of incomplete observations o_t.

A. Observations

At each timestep, the agent is presented with an observation o_t consisting of the following: The current gripper pose $o_{t,p}$ given as an $\mathbb{R}^{3\times3}$ rotation matrix and an $\mathbb{R}^3$ translation vector; the (static) transform between the gripper and the camera $o_{t,c}$, also given as a ($\mathbb{R}^{3\times3}$, $\mathbb{R}^3$) tuple; A depth image $o_{t,d} \in \mathbb{R}^{H\times W}$ captured from the camera; and a binary segmentation image $o_{t,s} \in \{0,1\}^{H\times W}$ captured from the same camera frame. The segmentation mask is 1 wherever a pixel is occupied by the grasping object and zero otherwise. During training, we render $o_{t,s}$ directly within the simulator, but in the real world, we have to perform semantic segmentation on an RGB image (Section III-E).

B. Actions

The action a_t consists of two components: A relative change in gripper pose $a_{t,\Delta p} \in \mathbb{R}^6$ and a decision on whether to grasp or not $a_{t,g} \in \{0,1\}$. $a_{t,\Delta p}$ is a twist vector in the gripper frame where the first three components represent a translation vector, and the last three comprise a rotation vector. The grasping action $a_{t,g}$ is a binary decision variable. If set to false, nothing happens. If set to true, the episode terminates, the gripper is closed and the object lifted.

C. Rewards

We use a simple, binary reward that directly reflects our high-level objective; whether the object was successfully picked up or not. After a grasp has been requested and the gripper has been lifted, we check whether the prop was successfully lifted along. If so, a fixed positive reward will be credited to the last MDP transition (i.e. the step where the grasp action was set to true). If not, the reward is zero. Likewise, transitions where the grasp action was set to false will always have a zero reward.

III. METHODOLOGY

To train the RL agent in simulation, we use a lightly modified version of the Stochastic Latent Actor-Critic (SLAC) algorithm [17]. This is an off-policy algorithm designed for vision-oriented control problems. The algorithm consists of three components: A latent variable model M, a policy π, and an action-value function Q, all of which will be explained below. To facilitate sim-to-real transfer, we make use of domain randomization and data augmentation and integrate the algorithm with off-the-shelf segmentation tools.

A. Latent Variable Model

The latent variable model M learns useful representations for the Actor and Critic. It achieves this through the proxy objective of modeling the dynamics and reward function of the MDP. Given a history of actions $a_{0:t-1}$ and noisy observations $o_{0:t}$, the true state s_t is approximated with a latent variable z_t. This is done by estimating the posterior $q(z_{t+1} \mid f(o_{t+1}), z_t, a_t)$ using amortized variational inference with the following high-level components.

$$
\begin{aligned}
\text{Observation Encoder:} \quad & e_t = f_\psi(o_t) && (1)\\
\text{Latent Prior (init):} \quad & p_\psi(z_0) && (2)\\
\text{Latent Prior (next):} \quad & p_\psi(z_{t+1} \mid z_t, a_t) && (3)\\
\text{Latent Posterior (init):} \quad & q_\psi(z_0 \mid e_0) && (4)\\
\text{Latent Posterior (next):} \quad & q_\psi(z_{t+1} \mid e_{t+1}, z_t, a_t) && (5)\\
\text{Observation Decoder:} \quad & g_\psi(o_t^* \mid z_t) && (6)\\
\text{Reward Predictor:} \quad & h_\psi(r_t \mid z_t, a_t, z_{t+1}) && (7)
\end{aligned}
$$

All the components are parametrized mappings governed by ψ, but only the observation encoder f_ψ is a deterministic function. The rest are stochastic functions that output distributions. The prior p_ψ models how a system state z_t is initialized and transitions to a successor state z_{t+1} when action a_t is taken. The posterior q_ψ models the same, except conditioned on the most recent encoded observation $e_{t+1} = f_\psi(o_{t+1})$. g_ψ reconstructs the original observation o_t^*, and h_ψ predicts the reward r_t. The difference between o^* and o is that $g_\psi(o^* \mid z)$ evaluates the raw observation whereas the input encoder $f_\psi(o)$ gets an augmented input during training, leading to an implicit denoising objective. The model is trained to maximize the Evidence Lower Bound (ELBO) with the following minimization objective:

$$
\begin{aligned}
J_M(\psi) = \mathop{\mathbb{E}}_{z\sim q_\psi} \Bigg[\sum_{t=0}^{\tau} & - \log g_\psi(o_{t+1}^* \mid z_{t+1}) \\
& - \log h_\psi(r_{t+1} \mid z_{t+1}) \\
& + D_{KL}(q_\psi(z_{t+1} \mid \cdot) \mid\mid p_\psi(z_{t+1} \mid \cdot)) \Bigg]
\end{aligned} \tag{8}
$$

The architecture of M is similar to [17]. Multilayer Perceptrons (MLPs) with two hidden layers are used to model p_ψ, q_ψ, and h_ψ. f_ψ preprocesses concatenated images $(o_{t,s}, o_{t,d})$ with a CNN and the vector $(o_{t,p})$ with an MLP before mixing the output with a linear mapping to get e_t. Equivalently, g_ψ reconstructs images with a mirrored, transposed CNN and the vector part with an MLP.

B. Actor-Critic

The Actor and Critic of SLAC largely resembles the SAC-algorithm [18]. It consists of two components: A policy $\pi_\phi(a_t \mid s_t)$ that generates distributions over actions, and a state-action function $Q_\theta(s_t, a_t)$, here implemented as neural networks governed by parameters ϕ and θ. SAC optimizes a soft policy, by augmenting the raw environment reward r_t with an intrinsic entropy bonus $\mathcal{H}(\pi_\phi)$ weighted by coefficient α.

The Q-networks estimate the expected sum of future reward, discounted by γ, when the agent is in state s_t and has committed to action a_t by minimizing the squared, 1-step Bellman residual J_Q. We train two Q-networks (θ^1, θ^2) to avoid overestimation and maintain polyak-averaged versions of them $(\bar{\theta}^2, \bar{\theta}^2)$ to provide stable bootstrap estimates.

$$
\begin{aligned}
J_Q(\theta^i) &= \frac{1}{2}\Big(Q_{\theta^i}(s_t, a_t) - (r_t + \gamma \bar{V}(s_{t+1}))\Big)^2 \\
\bar{V}(s_{t+1}) &= \mathop{\mathbb{E}}_{\pi}\left[\bar{Q}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1} \mid s_{t+1})\right] \\
\bar{Q}(s_{t+1}, a_{t+1}) &= \min_{j\in\{1,2\}} Q_{\bar{\theta}^j}(s_{t+1}, a_{t+1})
\end{aligned} \tag{9}
$$

Preliminary training runs sometimes diverged due to Q-function overestimation. To mitigate this, we incorporate a variation of the CQL [19] regularizer. This results in the following minimization objective being added to J_Q with mixing coefficient β.

$$
R_Q(\theta^i) = \beta\Big(\log \sum_{a\sim\pi_\phi} e^{Q_{\theta^i}(s_t, a)} - Q_{\theta^i}(s_t, a_t)\Big) \tag{10}
$$

The policy is trained to maximize the Q-function. Actions are sampled with gradient, fed into the differentiable Q-networks, and the policy climbs the gradient by minimizing:

$$
J_\pi(\phi) = \mathop{\mathbb{E}}_{\pi}\left[\alpha \log(\pi_\phi(a_t \mid s_t)) - \min_{i\in\{1,2\}} Q_{\theta^i}(s_t, a_t)\right] \tag{11}
$$

Similarly to [17], we approximate the unknown state value s_t with outputs of M so that $Q(s_t, a_t) \approx Q(z_t, a_t)$ and $\pi(a_t \mid s_t) \approx \pi(a_t \mid e_{t-k:t}, a_{t-k:t-1})$. Both π_ϕ and Q_θ are implemented as MLPs with two hidden layers. The Q-networks output scalars[1]. The policy network outputs an $\mathbb{R}^{13}$ vector that parametrizes two distributions. The distribution for $a_{t,\Delta p}$ is given as a tanh-transformed, isotropic $\mathbb{R}^6$ Gaussian with mean and standard deviation given by the first 12 outputs. The 13th output parametrizes a Bernoulli distribution from which we sample $a_{t,g}$. We minimize $J_Q + R_Q$ and J_π with gradient descent and the Adam [20] optimizer.

C. Scripted Seed Data

The binary reward complicates learning. An untrained policy is unlikely to stumble upon a successful grasp, and without the contrast of positive and negative examples, the critic cannot learn. We overcome this by bootstrapping learning with positive examples. Using a few props with known grasp poses, we roll out a hand-crafted policy to get seed data for the replay buffer ($\sim 85\%$ success rate).

D. Domain Randomization and Augmentation

To bridge the sim-to-real gap, we make extensive use of domain randomization. Table I provides an exhaustive list. Rows 3, 4, 9, 12, and 13 are randomized by drawing a scaling coefficient from a log-uniform distribution. The rest are sampled uniformely.

The simulated image observations are augmented (Fig. 2). Unlike domain randomization, which is only done when the data is collected, the augmentations are sampled before every network evaluation. To give the noise pattern structure, we rely on random Perlin noise.

[1]To handle the discrete nature of $a_{t,g}$, the Q-network first generates an $\mathbb{R}^2$ vector and then compute the inner product with $[1 - \pi(a_{t,g}), \pi(a_{t,g})]$.

TABLE I

DOMAIN RANDOMIZATION

	Randomization	Min	Max	SI
1	Arm base position (x, y, z)	-2.00	2.00	cm
2	Arm base rotation (up-axis)	-0.05	0.05	rad
3	Arm max cartesian force norm	6.66	15.0	N
4	Arm max cartesian torque norm	16.6	37.5	Nm
5	Grippier init. position (x, y, z)	-5.00	5.00	cm
6	Grippier init. rotation (any axis)	-0.40	0.40	rad
7	Gripper-camera rel. position (x, y, z)	-1.00	1.00	cm
8	Gripper-camera rel. rotation (any axis)	-0.02	0.02	rad
9	Camera field of view	55.2	60.9	deg
10	Grasping object init. position (x, y, z)	-15.0	15.0	cm
11	†Grasping object init. rotation (any axis)	-3.14	3.14	rad
12	Grasping object friction (scale)	0.66	1.50	-
13	Grasping object size (scale)	0.83	1.20	-
14	Time between MDP steps	-180.0	220.0	ms
15	Control-to-observation delay	0.00	30.0	ms

†The initial orientation of the grasp object is only randomized half the time. In the other half, we leave the default mesh orientation unchanged.

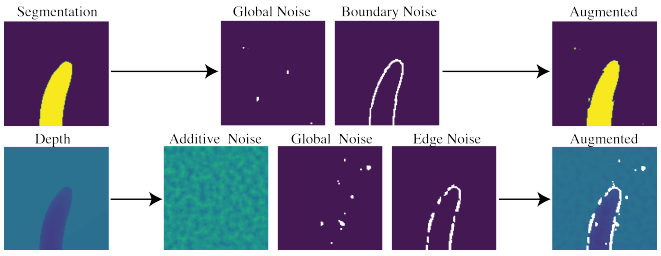


Fig. 2. Augmentations pipeline for simulated images. For the segmentation image, we swap the mask randomly across the entire image with a low probability and around the boundaries with a high probability. For the depth image, we emulate noisy readings with additive noise and mimic missing readings with thresholded noise. The threshold for missing readings is amplified in regions where the depth image has large gradients (edges).

E. Segmentation Pipeline

For real-world segmentation images, we use the filtered output of the Fast Segment Anything Model (FastSAM) [21]. This is a general-purpose segmentation model that maps an RGB input image to a set of candidate masks $\{o_{t,s}^{(i)}\}_{i=1}^{N_{FastSam}}$. FastSAM comes with several methods for narrowing down the selection, but we only found these sufficient for manually determining the initial segmentation mask $o_{o,s}$.

To track a segmentation mask across time, we leverage the dynamics model from SLAC. Given a posterior estimate of the current latent state z_t, we sample the prior dynamics model $p_\psi(z_{t+1} \mid z_t, a_t)$ to get a prediction of the impending latent state z_{t+1}. The prediction is then fed into the observation decoder $g_\psi(o_t \mid z_t) \in [0,1]^{H \times W}$ to get a predictive distribution over the segmentation mask for the next step. Once the next RGB image arrives, we generate candidate masks with FastSAM and select the one with the highest likelihood under g.

$$o_{t+1,s} = \underset{i \in N_{\text{FastSam}}}{\arg\max}\, g_\psi(o_{t+1,s}^{(i)} \mid \hat{z}_{t+1}),\ \hat{z}_{t+1} \sim p(z_{t+1} \mid z_t, a_t) \tag{12}$$

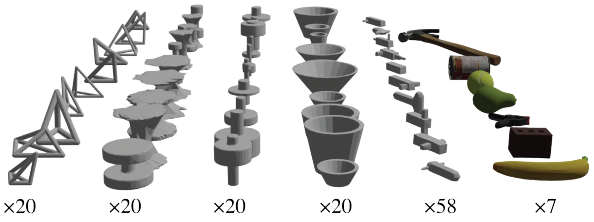


Fig. 3. Training grasp objects. We use 138 procedurally generated grasping objects and 7 objects from the YCB dataset [24] (right).

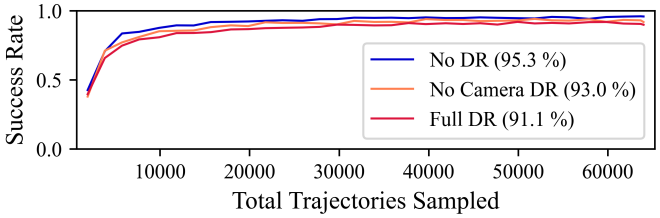


Fig. 4. Training results for different domain randomization (DR) configurations (3 seed average). "Full DR" uses all of Table I, "No camera DR" excludes rows (7, 8, 9), and "No DR" only employs rows (1, 2, 10, 11).

IV. EXPERIMENTAL SETUP

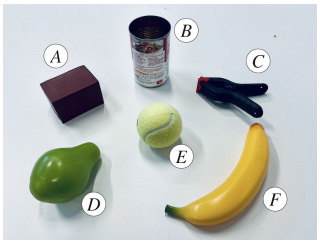
We train in simulation and evaluate in the lab. In both domains, the agent runs at 5Hz and the cartesian actions are realized with Operational Space Control (OCS) [22].

A. Simulation and Training

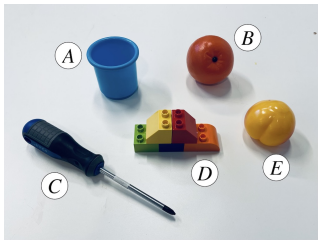
Our training environment simulates grasping with 145 different meshes (Fig. 3) in Nvidia Isaac Gym [23]. We sample 1024 seed episodes using scripted behavior to grasp 7 objects with manually labeled grasp poses. The seed data is first used to pretrain the latent variable model for 50,000 iterations, after which we begin sampling more data with the Actor's policy. Data is sampled from 32 parallel environments and stored in a replay buffer with a capacity of 32,000 trajectories. We cut off the episode if more than 50 steps have passed. For each trajectory collected, we train the latent variable model 10 times with a batch size of 32 and the Actor-Critic with a batch size of 256. Training lasts until 64,000 trajectories have been sampled. With the CQL regularization strength β set to 0.1, we found the learning algorithm to converge reliably. The seed data proved paramount, without it the algorithm failed to learn. The domain randomizations, in particular those interacting with the camera, substantially affected final performance (Fig. 4).

B. Lab Setup

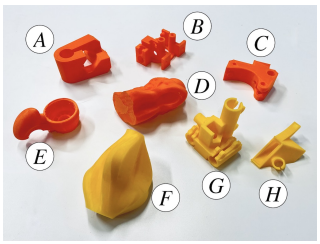
The real-world setup consists of the following elements: Panda 3 robot arm, generic Panda two-finger gripper, an Intel Realsense D405 RGB-D wrist-mounted camera, and a work surface on which the grasping objects are placed. Two desktop computers and the robot are wired together on a LAN. The first desktop runs real-time Linux and controls the robot. The second is equipped with an RTX 4080 GPU and runs the pytorch-based agent. Communication between



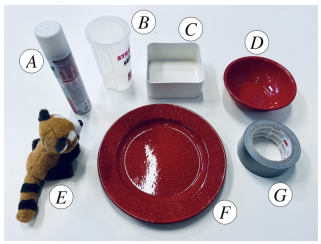
(a) Previously seen YCB objects



(b) Not previously seen YCB objects



(c) Adversarial objects



(d) Challenging objects

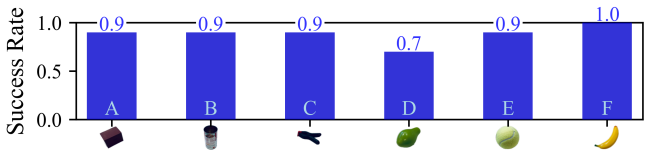
Fig. 5. Objects used for real-world evaluation

the two computers, as well as the OSC implementation, is provided by the Deoxys framework [25].

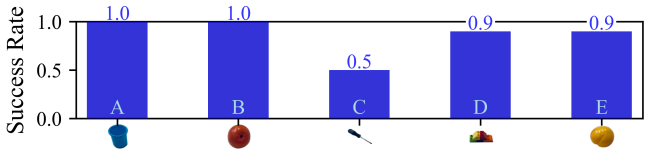
C. Evaluation

For real-world evaluation, we use four sets with a total of 26 grasping objects (Fig. 5). The first two sets were taken from [26] and used to evaluate zero-shot transfer for basic grasps. The "previously unseen YCB objects" (Fig. 5b) are analogous in shape and expected grasping strategy, but different enough to test whether the system can handle variations that can be expected in the wild. Similarly to [1], we also evaluate grasping on 8 3D-printed objects with adversarial geometry [27]. The "challenging objects" set (Fig. 5d) was curated by us to test the limits of the system. Most of the objects in the aforementioned sets can be adequately grasped by aiming at the center of mass and grasping along the major planar axis (analogously to the VS scheme in [3]). For most of the challenging objects, this is not a viable strategy. Objects C, D, F, and G are too wide in the plane and have to be grasped at any point along their rim. In addition, the soup bowl (D) and dinner plate (F) have to be grasped at an angle, which allows us to test the dexterity of full 6-DoF grasping. Similarly, objects A and B are cylinders with high horizontal grasp affordance (Fig. 7b). The plush toy (E) is compliant, fuzzy, and has many potential grasps. Moreover, many of the objects are much wider (F) and taller (A, B) than any of the training objects.

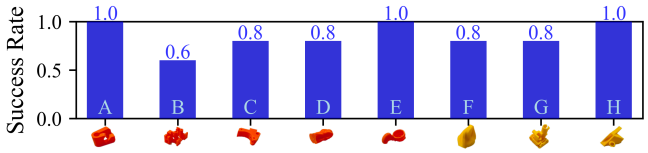
For each grasping object, we run 10 trials with the object starting in an arbitrary pose on the table. For the first three sets (Fig. 5a-c), we vary all stable orientations. For the last set (Fig. 5d), we only rotate around the up-axis. We let the agent run until the grasp action is set to true, after which the gripper is closed and raised. If the object is still grasped after lift-up, we consider it a success.



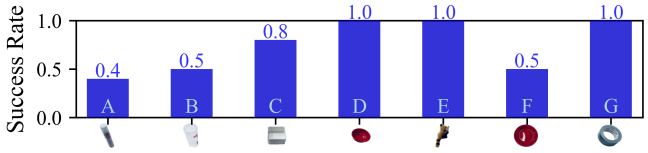
(a) Previously seen YCB objects (88.3% mean success rate)



(b) Not previously seen YCB objects (86.0% mean success rate)

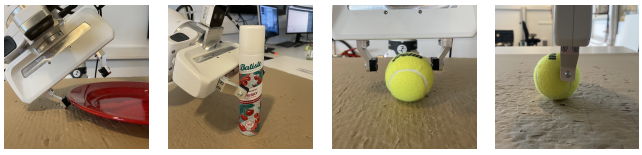


(c) Adversarial objects (85.0% mean success rate)



(d) Challenging objects (74.3% mean success rate)

Fig. 6. Results from grasping objects in the real world (10 trials per object).



(a) (b) (c) (d)

Fig. 7. (a) Sideways grasping. (b) Horizontal grasping, which the agent never carried out. (c, d) Illustration of observed positioning error. The robot would often grasp the object slightly to the side (c) and behind (d).

V. Results and Discussion

A. Grasping in the Real World

1) Previously Seen YCB Objects (88.3% success): The results are shown in Fig. 6a. The grasping agent achieved a perfect score for the banana (F) and an almost perfect score for the rest, with the exception of the pear (D). The three registered failures for the pear were caused by a slip due to the robot grasping it at the slightly tapered side of the bulb. The real-world tomato soup can (B) is slightly different from its simulated counterpart. In simulation, it is completely closed, whereas our real-world version is open at one of the ends. Nevertheless, the agent adapted its strategy, grasping it by the rim when the open side was facing up, and by the entire body otherwise.

2) Previously unseen YCB Objects (86.0% success): In general, we observed good results for these objects (Fig. 6b). While the success rate for the screwdriver (C) is low, we do not believe it to be a novel-object generalization issue. All

five failures were highly similar to those of the in-sample pear object; the robot grasped the handle at the tapering part of its center and it slipped during liftup. The one failure on the Duplo bricks (D) was caused by another slip, and the one for the peach (E) was caused by a slight positioning error.

3) Adversarial Objects (85.0% success): Results on the adversarial objects (Fig. 6c) are similar to those reported by [1] (81% vs our 85%). In contrast to [1], who use an in-built compliant gripper, we use the Panda gripper without any finger modifications. A particularly hard case was object B. It is highly dissimilar to anything seen during training and its many small protrusions make for easy accidental collisions with the gripper. Object F was dropped once due to slip and once due to poor positioning. Other failures were highly dependent on object orientation; two for objects C and D occurred whilst placed on the table in an upright, tall-and-narrow pose.

4) Challenging Objects (74.3% success): For these objects, we got a mixed bag of results. All grasps on the soup bowl (D), plush toy (E), and tape roll went successfully, and the plastic box (C) was lifted up 8/10 times. For objects C and G, the robot always went for the rim. For the bowl (D) and dinner plate (F), it significantly tilted the gripper sideways before grasping the rim, demonstrating the power of 6-DoF grasping. For several objects (B, C, D, F, G) we observed that the robot went for grasps on the right side, which turned detrimental for trials with the large dinner plate (F); attempting and failing a difficult grasp on the right side, even when a better opportunity was available on the left.

Performance was lower on the tall, cylindrically shaped objects. No horizontal grasp (Fig. 7b) around the main axis was ever observed. The five successful grasps on the protein shaker (B) were all on the thin rim. Among the four successful grasps on the hair spray canister (A), two were top-down on the cap and two involved the robot tipping it over and then picking it up from the ground. While the tipping behavior was too rare to seem deliberate, it showcases a benefit of the closed-loop RL approach: A single horizontal grasp would have been preferable, but it managed to recover after significantly disturbing the object.

Although methodologically different from existing 6-DoF grasping approaches, our results (74.3%) show a higher success rate than [7], who report a score of 62.4%. They are also comparable to [8], who report a score of 80.3%. It is worth noting that [8] achieves this success rate for a mix of YCB and challenging objects without specifying the individual accuracy, making a direct comparison difficult.

B. Failure Modes and Limitations

1) Positioning Errors: A significant portion of the failed grasps were close to successful, yet small positioning errors led to the object slipping. This can be attributed to a myriad of noisy factors, such as imperfections in neural network representations, synchronization discrepancies between camera captures and robot state readings, calibration errors between the gripper and camera, and inaccuracies in the OSC controller. However, we observed a slight trend in the positioning errors. Particularly evident when grasping spherical objects, the gripper was consistently positioned slightly to the right and a bit behind the optimal grasp position (Fig. 7c, 7d). This error was amplified for objects farther away from the robot.

2) Gripper-Object "Wrestling": Most of the observed grasps were swift and clean. Sometimes, the robot made small adjustments before committing to closing the gripper, which is a desired consequence of our closed-loop approach. Other times it would keep adjusting while pushing down on the object, making it look like it was wrestling it. A long-winded struggle is unlikely to be fruitful because vision is poor up close, and the policy is only conditioned on the last 8 observations (1.6 seconds). A rational agent with the same limitations would retreat and reassess before going for another grasp, but we did not observe this behavior. Future work may want to use a longer observation context or draw on the rich literature of POMDPs [28] to design agents that factor the value of information gathering in their decisions.

3) Unimodal Grasping: The agent showed a strong preference for a single type of "right-sided" grasps, even when left-sided grasps were available and more suitable. Within our RL agent architecture, the culprit is likely to be the policy network. The Gaussian-based policy head is, by definition, unimodal. Variations in initial conditions may lead to different outcomes, but for any given observation-action history, it can never concentrate likelihood on two or more substantially different actions. For some objects, this is not an issue as there is an obvious single best grasp. For others, it may be advantageous to have a portfolio of potential grasps and use the one most appropriate. A more diverse distribution of grasps may also be a boon to training as it aids in exploration. Future work may want to consider policies based on a richer family of distributions.

VI. CONCLUSIONS

In this paper, we presented a novel, grasping framework based on RL. Using only a handful of simulated demonstrations and a binary reward, we train an agent capable of eye-in-hand, closed-loop grasping. The trained policy is transferred straight from simulation and evaluated on a real robot without any fine-tuning. We demonstrate that minimally supervised RL is competitive with less supervised alternatives, achieving 86% mean zero-shot success rate when grasping previously unseen objects. In contrast to previous approaches, our approach is closed-loop, eye-in-hand, generates continuous 6-DoF actions, and decides autonomously when to close the gripper. Salient failure cases are identified and discussed, and we provide suggestions on how they can be addressed. For future work, we intend to enrich our framework to tackle static and dynamic clutters of challenging objects within domains such as food, marine, and manufacturing.

ACKNOWLEDGMENT

The work is supported by the GentleMAN (299757) and BIFROST (313870) projects (RCN Norway). We thank Elling Ruud Øye for providing invaluable help in the lab.

REFERENCES

[1] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *Robotics: Science and Systems XIV*, pp. 1–10, 2018.

[2] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, vol. 39, pp. 183–201, Mar. 2020. Publisher: SAGE Publications Ltd STM.

[3] O.-M. Pedersen, E. Misimi, and F. Chaumette, "Grasping Unknown Objects by Coupling Deep Reinforcement Learning, Generative Adversarial Networks, and Visual Servoing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, (Paris, France), pp. 5655–5662, IEEE, May 2020.

[4] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*, pp. 651–673, PMLR, 2018.

[5] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4238–4245, IEEE, 2018.

[6] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *The International Journal of Robotics Research*, vol. 41, no. 7, pp. 690–705, 2022.

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

[8] 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)*, pp. 6232–6238, IEEE, 2020.

[9] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13438–13444, IEEE, 2021.

[10] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4978–4985, 2020.

[11] Y. Hou and J. Li, "Learning 6-dof grasping with dual-agent deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 166, p. 104451, 2023.

[12] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4368–4375, July 2020.

[13] H. Zhang, J. Peeters, E. Demeester, and K. Kellens, "Deep Learning Reactive Robotic Grasping With a Versatile Vacuum Gripper," *IEEE Transactions on Robotics*, vol. 39, pp. 1244–1259, Apr. 2023. Conference Name: IEEE Transactions on Robotics.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] Y. Song, A. Romero, M. limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, 2023.

[16] H.-n. Wang, N. Liu, Y.-y. Zhang, D.-w. Feng, F. Huang, D.-s. Li, and Y.-m. Zhang, "Deep reinforcement learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 12, pp. 1726–1744, 2020.

[17] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," *Advances in Neural Information Processing Systems*, vol. 33, pp. 741–752, 2020.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[19] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv preprint arXiv:2306.12156*, 2023.

[22] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[23] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[24] 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.

[25] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *6th Annual Conference on Robot Learning*, 2022.

[26] K. Sumskiy, E. Misimi, and F. Chaumette, "A general framework for grasping unknown objects by coupling vision-based drl, real-to-sim adaptation, and ibvs," *International Journal of Robotics Research (IJRR)*, 2023. Under Review.

[27] 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," *Robotics: Science and Systems (RSS)*, 2017.

[28] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, 2022.