Autonomous Robots (2023) 47:1175–1193
https://doi.org/10.1007/s10514-023-10112-w

On robot grasp learning using equivariant models

Xupeng Zhu[1] · Dian Wang[1] · Guanang Su[1] · Ondrej Biza[1] · Robin Walters[1] · Robert Platt[1]

Received: 7 February 2023 / Accepted: 29 May 2023 / Published online: 4 July 2023

Abstract
Real-world grasp detection is challenging due to the stochasticity in grasp dynamics and the noise in hardware. Ideally, the system would adapt to the real world by training directly on physical systems. However, this is generally difficult due to the large amount of training data required by most grasp learning models. In this paper, we note that the planar grasp function is SE(2)-equivariant and demonstrate that this structure can be used to constrain the neural network used during learning. This creates an inductive bias that can significantly improve the sample efficiency of grasp learning and enable end-to-end training from scratch on a physical robot with as few as 600 grasp attempts. We call this method Symmetric Grasp learning (SymGrasp) and show that it can learn to grasp "from scratch" in less that 1.5 h of physical robot time. This paper represents an expanded and revised version of the conference paper Zhu et al. (2022).

Keywords Grasping · Equivariant models · On robot learning · Sample efficiency · Reinforcement learning · Transparent object grasping

1 Introduction

Grasp detection detects good grasp poses in a scene directly from raw visual input (e.g., RGB or depth images) using machine learning. The learning-based method generalizes to novel objects. This is in contrast to classical model-based methods that attempt to reconstruct the geometry and the pose of objects in a scene and then reason geometrically about how to grasp those objects.

Most current grasp detection models are data-driven, i.e., they must be trained using large offline datasets. For example, Mousavian et al. (2019) trains on a dataset consisting of over 7 M simulated grasps, Breyer et al. (2021) trains on over 2 M simulated grasps, Mahler et al. (2017) trains on grasp data drawn from over 6.7M simulated point clouds, and ten Pas et al. (2017a) trains on over 700k simulated grasps. Some models are trained using datasets obtained via physical robotic grasp interactions. For example, Pinto and Gupta (2015) trains on a dataset created by performing 50k grasp attempts over 700 h, Kalashnikov et al. (2018) trains on over 580k grasp attempts collected over the course of 800 robot hours, and Berscheid et al. (2021) train on a dataset obtained by performing 27k grasps over 120 h.

Such reliance on collecting large datasets necessitates either learning in simulation or using significant amounts of robot time to generate data, motivating the desire for a more sample efficient grasp detection model, i.e., a model that can achieve good performance with a smaller dataset. In this paper, we propose a novel grasp detection strategy that improves sample efficiency significantly by incorporating the equivariant structure into the model. We term our strategy Symmetric Grasp learning as SymGrasp. Our key observation is that the target grasp function (from images onto grasp poses) is SE(2)-equivariant. That is, rotations and translations of the input image should correspond to the same rotations and translations of the detected grasp poses at the output of the function. In order to encode the SE(2)-equivariance in the target function, we constrain the layers of

✉ Xupeng Zhu
zhu.xup@northeastern.edu

Dian Wang
wang.dian@northeastern.edu

Guanang Su
su.gu@northeastern.edu

Ondrej Biza
biza.o@northeastern.edu

Robin Walters
r.walters@northeastern.edu

Robert Platt
r.platt@northeastern.edu

[1] Khoury College of Computer Science, Northeastern University, Huntington Ave, Boston, MA 02115, USA

our model to respect this symmetry. Compared with conventional grasp detection models that must be trained using tens of thousands of grasp experiences, the equivariant structure we encode into the model enables us to achieve good grasp performance after only a few hundred grasp attempts.

This paper makes several key contributions. First, we recognize that the grasp detection function from images to grasp poses is a SE(2)-equivariant function. Then, we propose a neural network model using equivariant layers to encode this property. Finally, we introduce several algorithmic optimizations that enable us to learn to grasp online using a contextual bandit framework. Ultimately, our model is able to learn to grasp opaque (using depth images) and transparent objects (using RGB images) with a good success rate after only approximately 600 grasp trials—1.5 h of robot time. Although the model we propose here is only for 2D grasping (i.e., we only detect top down grasps rather than all six dimensions as in 6-DOF grasp detection), the sample efficiency is still impressive and we believe the concepts could be extended to higher-DOF grasp detection models in the future.

These improvements in sample efficiency are important for several reasons. First, since our model can learn to grasp in only a few hundred grasp trials, it can be trained easily on a physical robotic system. This greatly reduces the need to train on large datasets created in simulation, and it therefore reduces our exposure to the risks associated with bridging the sim2real domain gap—we can simply do all our training on physical robotic systems. Second, since we are training on a small dataset, it is much easier to learn on-policy rather than off-policy, i.e., we can train using data generated by the policy being learned rather than with a fixed dataset. This focuses learning on areas of state space explored by the policy and makes the resulting policies more robust in those areas. Finally, since we can learn efficiently from a small number of experiences, our policy has the potential to adapt relatively quickly at run time to physical changes in the robot sensors and actuators.

2 Related work

2.1 Equivariant convolutional layers

Equivariant convolutional layers incorporate symmetries into the structure of convolutional layers, allowing them to generalize across a symmetry group automatically. This idea was first introduced as G-Convolution (Cohen & Welling, 2016a) and Steerable CNN (Cohen & Welling, 2016b). E2CNN is a generic framework for implementing E(2) Steerable CNN layers (Weiler & Cesa, 2019). In applications such as dynamics (Walters et al., 2020; Wang et al., 2020b) and reinforcement learning (van der Pol et al., 2020; Mondal et al., 2020; Wang et al., 2021, 2022) equivariant models demonstrate improvements over traditional approaches.

2.2 Sample efficient reinforcement learning

Recent work has shown that data augmentation using random crops and/or shifts can improve the sample efficiency of standard reinforcement learning algorithms (Laskin et al., 2020a; Kostrikov et al., 2020). It is possible to improve sample efficiency even further by incorporating contrastive learning (van den Oord et al., 2018), e.g. CURL (Laskin et al., 2020b). The contrastive loss enables the model to learn an internal latent representation that is invariant to the type of data augmentation used. The FERM framework (Zhan et al., 2020) applies this idea to robotic manipulation and is able to learn to perform simple manipulation tasks directly on physical robotic hardware. The equivariant models used in this paper are similar to data augmentation in that the goal is to leverage problem symmetries to accelerate learning. However, whereas data augmentation and contrastive approaches require the model to *learn* an invariant or equivariant encoding, the equivariant model layers used in this paper *enforce* equivariance as a prior encoded in the model. This simplifies the learning task and enables our model to learn faster (see Sect. 6).

2.3 Grasp detection

In grasp detection, the robot finds grasp configurations directly from visual or depth data. This is in contrast to classical methods which attempt to reconstruct object or scene geometry and then do grasp planning. See Platt (2023) for a review on this topic.

2D Grasping: Several methods are designed to detect grasps in 2D, i.e., to detect the planar position and orientation of grasps in a scene based on top-down images. A key early example of this was DexNet 2.0, which infers the quality of a grasp centered and aligned with an oriented image patch (Mahler et al., 2017). Subsequent work proposed fully convolutional architectures, thereby enabling the model to quickly infer the pose of *all* grasps in a (planar) scene (Morrison et al., 2018; Satish et al., 2019; Depierre et al., 2018; Kumra et al., 2019; Zhou et al., 2018) (some of these models infer the z coordinate of the grasp as well).

3D Grasping: There is much work in 3D grasp detection, i.e., detecting the full 6-DOF position and orientation of grasps based on truncated signed distance function (TSDF) or point cloud input. A key early example of this was GPD (ten Pas et al., 2017b) which inferred grasp pose based on point cloud input. Subsequent work has focused on improving grasp candidate generation in order to improve efficiency, accuracy, and coverage (Huang et al., 2022; Mousavian et al.,

2019; Sundermeyer et al., 2021; Jiang et al., 2021; Fang et al., 2020; Breyer et al., 2021; Berscheid et al., 2021).

On-robot Grasp Learning: Another important trend has been learning to grasp directly from physical robotic grasp experiences. Early examples of this include (Pinto & Gupta, 2015) who learn to grasp from 50k grasp experiences collected over 700 h of robot time and Levine et al. (2018) who learn a grasp policy from 800k grasp experiences collected over two months. QT-Opt (Kalashnikov et al., 2018) learns a grasp policy from 580k grasp experiences collected over 800 h and James et al. (2019) extends this work by learning from an additional 28k grasp experiences. Song et al. (2020) learns a grasp detection model from 8k grasp demonstrations collected via demonstration and Zeng et al. (2018) learns a online pushing/grasping policy from just 2.5k grasps.

Transparent objects grasping: Commercial depth sensors that are based on structured-light or time-of-flight techniques often fail to sense transparent objects accurately (Weng et al., 2020). Pixels in the depth image are often dropped due to specularities or the object is simply invisible to the sensor because it is transparent (Sajjan et al., 2020). To avoid this type of failure, RGB or RGB-D sensors are commonly used. Weng et al. (2020) infers grasp pose from RGBD images. They collect paired RGB and D images on opaque objects and utilize transfer learning from a trained D modality model to an RGBD modality model. Sajjan et al. (2020) reconstructs depth images from RGBD images using CNN and then performs grasp detection. Likewise, Ichnowski et al. (2021) and Kerr et al. (2022) infer grasp pose from reconstructed depth images, but using neural radiance field (NeRF) (Mildenhall et al., 2021) for reconstruction. These methods either rely on collecting paired images for training or require training a NeRF model per grasp during evaluation. In contrast, our method is trained directly on RGB images without requiring paired images or NeRF models.

Equivariance through canonicalization in grasping: An alternative to modeling rotational symmetry using equivariant neural network layers is an approach known as *canonicalization* where we learn a model over the non-equivariant variables assuming a single "canonical" group element (Wang et al., 2020b; Kofinas et al., 2021; Gao et al., 2020). Equivariance based on canonicalization is common in robotic grasping where it is not unusual to translate and rotate the input image so that it is expressed in the reference frame of the hand, e.g. Mahler et al. (2017), ten Pas et al. (2017b) and Mousavian et al. (2019). This way, the neural network model need only infer the quality of a grasp for a single canonical grasp pose rather than over arbitrary translations and orientations. In this paper, we compare our model-based approach to equivariance with VPG, a method that obtains rotational equivariance via canonicalization (Song et al., 2020). Our results in Sect. 6.2 suggest that the model-based approach has a significant advantage.

3 Background

3.1 Equivariant neural network models

In this paper, we use equivariant neural network layers defined with respect to a finite group (e.g. a finite group of rotations). Equivariant neural network (Weiler & Cesa, 2019; Cohen et al., 2018; Cohen & Welling, 2016b, a) encodes a function $f : X \to Y$ that satisfy the equivariance constraint: $gf(x) = f(gx)$, where $g \in G$ is an element of a finite group. gx is shorthand for the action of g on x, e.g. rotation of an image x. Similarly, $g(f(x))$ describes the action of g on $f(x)$. Below, we make these ideas more precise and summarize how the equivariance constraint is encoded into a neural network layer.

3.1.1 The cyclic group $C_n \leq \mathrm{SO}(2)$

We are primarily interested in equivariance with respect to the group of planar rotations, SO(2). However, in practice, in order to make our models computationally tractable, we will use the cyclic subgroup C_n of SO(2), $C_n = \{2\pi k/n : 0 \leq k < n\}$. C_n is the group of discrete rotations by multiples of $2\pi/n$ radians.

3.1.2 Representation of a group

The way a group element $g \in G$ acts on x depends on how x is represented. If x is a point in the plane, then g acts on x via the *standard representation*, $\rho_1(g)x$, where $\rho_1(g)$ is the standard 2×2 rotation matrix corresponding to g. In the hidden layers of an equivariant neural network model, it is common to encode a separate feature map for each group element. For example, suppose G is the order n cyclic group and suppose x is a set of features $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{n\times\lambda}$ that maps the kth group element to a feature x_k. The *regular representation* of g acts on x by permuting its elements: $\rho_{reg}(g)x = (x_{n-m+1}, \ldots, x_n, x_1, x_2, \ldots, x_{n-m})$ where g is the mth element in C_n. Finally, it is sometimes the case that x is invariant to the action of the group elements. In this case, we have the *trivial representation*, $\rho_0(g)x = x$.

3.1.3 Feature maps of equivariant convolutional layers

An equivariant convolutional layer maps between feature maps which transform by specified representations ρ of the group. In the hidden layers of an equivariant model, generally, an extra dimension is added to the feature maps to encode group elements via the regular representation. So, whereas the feature map used by a standard convolutional layer is a tensor $\mathcal{F} \in \mathbb{R}^{m\times h\times w}$, an equivariant convolutional layer adds an extra dimension: $\mathcal{F} \in \mathbb{R}^{k\times m\times h\times w}$, where k denotes the

dimension of the group representation. This tensor associates each pixel $(u, v) \in \mathbb{R}^{h\times w}$ with a matrix $\mathcal{F}(u, v) \in \mathbb{R}^{k\times m}$.

3.1.4 Action of the group operator on the feature map

Given a feature map $\mathcal{F} \in \mathbb{R}^{k\times m\times h\times w}$ associated with group G and representation ρ, a group element $g \in G$ acts on $\mathcal{F}$ via:

$$(g\mathcal{F})(x) = \rho(g)\mathcal{F}(\rho_1(g)^{-1}x), \tag{1}$$

where $x \in \mathbb{R}^2$ denotes pixel position. The RHS of this equation applies the group operator in two ways. First, $\rho_1(g)^{-1}$ rotates the pixel position x using the standard representation. Second, ρ applies the rotation to the feature representation. If the feature is invariant to the rotation, then we use the trivial representation $\rho_0(g)$. However, if the feature vector changes according to rotation (e.g. the feature denotes grasp orientation), then it must be transformed as well. This is accomplished by setting ρ in Eq. 1 to be the regular representation that transforms the feature vector by a circular shift.

3.1.5 The equivariant convolutional layer

An equivariant convolutional layer is a function h from $\mathcal{F}_{in}$ to $\mathcal{F}_{out}$ that is constrained to represent only equivariant functions with respect to a chosen group G. The feature maps $\mathcal{F}_{in}$ and $\mathcal{F}_{out}$ are associated with representations ρ_{in} and ρ_{out} acting on feature spaces $\mathbb{R}^{k_{in}}$ and $\mathbb{R}^{k_{out}}$ respectively. Then the equivariant constraint for h is Cohen et al. (2018):

$$h(g\mathcal{F}_{in}) = gh(\mathcal{F}_{in}) = g\mathcal{F}_{out}. \tag{2}$$

This constraint can be implemented by tying kernel weights $K(y) \in \mathbb{R}^{k_{out}\times k_{in}}$ in such a way as to satisfy the following constraint (Cohen et al., 2018):

$$K(gy) = \rho_{out}(g)K(y)\rho_{in}(g)^{-1}. \tag{3}$$

Please see "Appendix Appendix B" for an example of equivariant convolustional layer.

When all hidden layers h in a neural network satisfy Eq. 2, then by induction the entire neural network is equivariant (Cohen et al., 2018).

3.2 Augmented state representation (ASR)

We will formulate SE(2) robotic grasping as the problem of learning a function from an m channel image, $s \in S = \mathbb{R}^{m\times h\times w}$, to a gripper pose $a \in A = \mathrm{SE}(2)$ from which an object may be grasped. Since we will use the contextual bandit framework, we need to be able to represent the

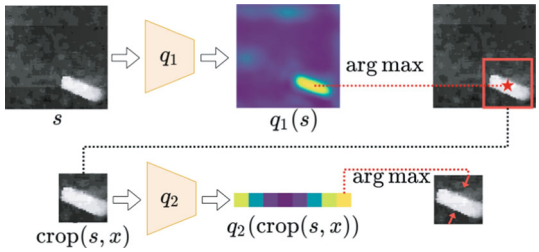


Fig. 1 Illustration of the ASR representation. Q_1 selects the translational component of an action, Q_2 selects the rotational component

Q-function, $Q : \mathbb{R}^{m\times h\times w} \times \mathrm{SE}(2) \to \mathbb{R}$. However, this is difficult to do using a single neural network due to the GPU memory limitation. To combat this, we will use the Augmented State Representation (ASR) (Sharma et al., 2017; Wang et al., 2020a) to model Q as a pair of functions, Q_1 and Q_2. Another advantage of using ASR is we can use different group order in Q_1, Q_2 , as explained in Sect. 5.1.3.

We follow the ASR framework that factors $\mathrm{SE}(2) = \mathbb{R}^2 \times \mathrm{SO}(2)$ into a translational component $X \subseteq \mathbb{R}^2$ and a rotational component $\Theta \subseteq \mathrm{SO}(2)$. The first function is a mapping $Q_1 : \mathbb{R}^{m\times h\times w} \times X \to \mathbb{R}$ which maps from the image s and the translational component of action X onto value. This function is defined to be: $Q_1(s, x) = \max_{\theta\in\Theta} Q(s, (x, \theta))$. The second function is a mapping $Q_2 : \mathbb{R}^{m\times h'\times w'} \times \Theta \to \mathbb{R}$ with $h' \leq h$ and $w' \leq w$ which maps from an image patch and an orientation onto value. This function takes as input a cropped version of s centered on a position x, $\mathrm{crop}(s, x)$, and an orientation, θ, and outputs the corresponding Q value: $Q_2(\mathrm{crop}(s, x), \theta) = Q(s, (x, \theta))$.

Inference is performed on the model by evaluating $x^* = \arg\max_{x\in X} Q_1(s, x)$ first and then evaluating $\theta^* = \arg\max_{\theta} Q_2(\mathrm{crop}(s, x^*), \theta)$. Since each of these two models, Q_1 and Q_2, are significantly smaller than Q would be, the inference is much faster. Figure 1 shows an illustration of this process. The top of the figure shows the action of Q_1 while the bottom shows Q_2. Notice that the semantics of Q_2 imply that the θ depends only on $\mathrm{crop}(s, x)$, a local neighborhood of x, rather than on the entire scene. This assumption is generally true for grasping because grasp orientation typically depends only on the object geometry near the target grasp point.

4 Problem statement

4.1 Planar grasp detection

The *planar grasp detection function* $\Gamma : \mathbb{R}^{m\times h\times w} \to \mathrm{SE}(2)$ maps from a top down image of a scene containing graspable objects, $s \in S = \mathbb{R}^{m\times h\times w}$, to a planar gripper pose, $a \in A = \mathrm{SE}(2)$, from which an object can be grasped. This is similar to

the formulations used by Mahler et al. (2017) and Morrison et al. (2018).

4.2 Formulation as a contextual bandit

We formulate grasp learning as a contextual bandit problem where the state is an image $s \in S = \mathbb{R}^{m \times h \times w}$ and the action $a \in A = \mathrm{SE}(2)$ is a grasp pose to which the robot hand will be moved and a grasp will be attempted, expressed in the reference frame of the image. After each grasp attempt, the agent receives a binary reward R drawn from a Bernoulli distribution with unknown probability $r(s, a)$. The true Q-function denotes the expected reward of taking action a from s. Since R is binary, we have that $Q(s, a) = r(s, a)$. This formulation of grasp learning as a bandit problem is similar to that used by, e.g. Danielczuk et al. (2020), Kalashnikov et al. (2018) and Zeng et al. (2018).

4.3 Invariance assumption

We assume that the (unknown) reward function $r(s, a)$ that denotes the probability of a successful grasp is invariant to translations and rotations $g \in \mathrm{SE}(2)$. Let gs denote the image s translated and rotated by g. Similarly, let ga denote the action translated and rotated by g. Therefore, our assumption is:

$$r(s, a) = r(gs, ga). \tag{4}$$

Intuitively, when the image of a scene transforms, the grasp poses (located with respect to the image) transform correspondingly.

5 SymGrasp: symmetric grasp learning

5.1 Equivariant learning

We use equivariant neural networks (Weiler & Cesa, 2019) to model the Q-function that enforces the invariance assumption in Sect. 4.3. Therefore once the Q function is fit to a data sample $r(s, a)$, it generalizes to any $g \in \mathrm{SE}(2)$ transformed data sample $r(gs, ga)$. This generalization could lead to a significant sample efficiency improvement during training.

5.1.1 Invariance properties of Q_1 and Q_2

The assumption that the reward function r is invariant to transformations $g \in \mathrm{SE}(2)$ implies that the optimal Q-function is also invariant to g, i.e., $Q(s, a) = Q(gs, ga)$. In the context of the augmented state representation (ASR, see Sect. 3.2), this implies separate invariance properties for Q_1 and Q_2:

$$Q_1(gs, gx) = Q_1(s, x) \tag{5}$$

$$Q_2(g_\theta(\mathrm{crop}(s, x)), g_\theta + \theta) = Q_2(\mathrm{crop}(s, x), \theta), \tag{6}$$

where $g_\theta \in \mathrm{SO}(2)$ denotes the rotational component of $g \in \mathrm{SE}(2)$, gx denotes the rotated and translated vector $x \in \mathbb{R}^2$, and $g_\theta(\mathrm{crop}(s, x))$ denotes the cropped image rotated by g_θ.

5.1.2 Discrete approximation of SE(2)

We implement the invariance constraints of Eqs. 5 and 6 using a discrete approximation to SE(2). We constrain the positional component of the action to be a discrete pair of positive integers $x \in \{1 \dots h\} \times \{1 \dots w\} \subset \mathbb{Z}^2$, corresponding to a pixel in s, and constrain the rotational component of the action to be an element of the finite cyclic group $C_n = \{2\pi k/n : 0 \leq k < n, i \in \mathbb{Z}\}$. This discretized action space will be written $\hat{\mathrm{SE}}(2) = \mathbb{Z}^2 \times C_n$.

5.1.3 Equivariant *Q*-learning with ASR

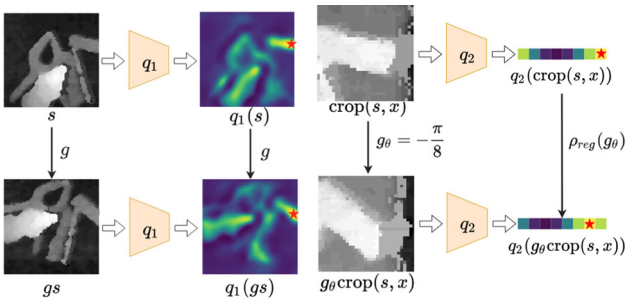
In Q-Learning with ASR, we model Q_1 and Q_2 as neural networks. We model Q_1 as a fully convolutional UNet (Ronneberger et al., 2015) $q_1 : \mathbb{R}^{m \times h \times w} \to \mathbb{R}^{1 \times h \times w}$ that generates Q value for each discretized translational actions from the input state image. We model Q_2 as a standard convolutional network $q_2 : \mathbb{R}^{m \times h' \times w'} \to \mathbb{R}^n$ that evaluates Q value for n discretized rotational actions based on the image patch. The networks q_1 and q_2 thus model the functions Q_1 and Q_2 by partially evaluating at the first argument and returning a function in the second. As a result, the invariance properties of Q_1 and Q_2 (Eqs. 5, 6) imply the equivairance of q_1 and q_2:

$$q_1(gs) = g q_1(s) \tag{7}$$

$$q_2(g_\theta \mathrm{crop}(s, x)) = \rho_{reg}(g_\theta) q_2(\mathrm{crop}(s, x)) \tag{8}$$

where $g \in \hat{\mathrm{SE}}(2)$ acts on the output of q_1 through rotating and translating the Q-map, and $g_\theta \in C_n$ acts on the output of q_2 by performing a circular shift of the output Q values via the regular representation ρ_{reg}.

This is illustrated in Fig. 2. In Fig. 2a we take an example of a depth image s in the upper left corner. If we rotate and translate this image by g (lower left of Fig. 2a) and then evaluate q_1, we arrive at $q_1(gs)$. This corresponds to the LHS of Eq. 7. However, because q_1 is an equivariant function, we can calculate the same result by first evaluating $q_1(s)$ and *then* applying the transformation g (RHS of Eq. 7). Figure 2b illustrates the same concept for Eq. 8. Here, the network takes the image patch $\mathrm{crop}(s, x)$ as input. If we rotate the image patch by g_θ and then evaluate q_2, we obtain the LHS of Eq. 8,



(a) Illustration of Equation 7 (b) Illustration of Equation 8.

Fig. 2 Equivariance relations expressed by Eqs. 7 and 8

$q_2(g_\theta \text{crop}(s,x))$. However, because q_2 is equivariant, we can obtain the same result by evaluating $q_2(\text{crop}(s,x))$ and circular shifting the resulting vector to denote the change in orientation by one group element.

5.1.4 Model architecture of equivariant q_1

As a fully convolutional network, q_1 inherits the translational equivariance property of standard convolutional layers. The challenge is to encode rotational equivariance so as to satisfy Eq. 7. We accomplish this using equivariant convolutional layers that satisfy the equivariance constraint of Eq. 2 where we assign $\mathcal{F}_{in} = s \in \mathbb{R}^{1\times m\times h\times w}$ to encode the input state s and $\mathcal{F}_{out} \in \mathbb{R}^{1\times 1\times h\times w}$ to encode the output Q-map. Both feature maps are associated with the trivial representation ρ_0 such that the rotation g operates on these feature maps by rotating pixels without changing their values. We use the regular representation ρ_{reg} for the hidden layers of the network to encode more comprehensive information in the intermediate layers. We found we achieved the best results when we defined q_1 using the dihedral group D_4 which expresses the group generated by rotations of multiples of $\pi/2$ in combination with vertical reflections.

5.1.5 Model architecture of equivariant q_2

Whereas the equivariance constraint in Eq. 7 is over $\hat{\text{SE}}(2)$, the constraint in Eq. 8 is over C_n only. We implement Eq. 8 using Eq. 2 with an input of $\mathcal{F}_{in} = \text{crop}(s,x) \in \mathbb{R}^{1\times m\times h'\times w'}$ as a trivial representation, and an output of $\mathcal{F}_{out} \in \mathbb{R}^{n\times 1\times 1\times 1}$ as a regular representation. q_2 is defined in terms of the group C_n, assuming the rotations in the action space are defined to be multiples of $2\pi/n$.

5.1.6 q_2 symmetry expressed as a quotient group

It turns out that additional symmetries exist when the gripper has a bilateral symmetry. In particular, it is often the case that rotating a grasp pose by π radians about its forward axis does not affect the probability of grasp success, i.e.,r is invariant to rotations of the action by π radians. When this symmetry is present, we can model it using the quotient group $C_n/C_2 \cong \{2\pi k/n : 0 \le k < n/2, k \in \mathbb{Z}, 0 \equiv \pi\}$ which pairs orientations separated by π radians into the same equivalence classes.

5.2 Other optimizations

While our use of equivariant models to encode the Q-function is responsible for most of our gains in sample efficiency (Sect. 6.3), there are several additional algorithmic details that, taken together, have a meaningful impact on performance.

5.2.1 Loss function

In the standard ASR loss function, given a one step reward $r(s,a)$, where $a = (x,\theta)$, Q_1 and Q_2 have targets (Wang et al., 2020a):

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 \tag{9}$$

$$\mathcal{L}_1 = \tfrac{1}{2}(Q_1(s,x) - \max_{\theta'} Q_2(\text{crop}(s,x),\theta'))^2 \tag{10}$$

$$\mathcal{L}_2 = \tfrac{1}{2}(Q_2(\text{crop}(s,x),\theta) - r(s,a))^2. \tag{11}$$

In $\mathcal{L}_1$ term, however, since the reward $r(s,a)$ is the ground truth return of $Q_2(\text{crop}(s,x),\theta)$, we correct Q_2 with $r(s,a)$. Denoted $\hat{Q}_2$ as the corrected Q_2

$$\hat{Q}_2(\text{crop}(s,x),\theta') = \begin{cases} Q_2(\text{crop}(s,x),\theta'), & \text{if } \theta' \neq \theta \\ r(s,a), & \text{if } \theta' = \theta \end{cases} \tag{12}$$

We then modify $\mathcal{L}_1$ to learn from $\hat{Q}_2$:

$$\mathcal{L}'_1 = \frac{1}{2}\Big(Q_1(s,x) - \max_{\theta'}\Big[\hat{Q}_2(\text{crop}(s,x),\theta')\Big]\Big)^2. \tag{13}$$

In addition to the above, we add an off-policy loss term $\mathcal{L}''_1$ that is evaluated with respect to an additional k grasp positions $\bar{X} \subset X$ sampled using a Boltzmann distribution from $Q_1(s)$:

$$\mathcal{L}''_1 = \frac{1}{2k}\sum_{x_i\in\bar{X}}\Big(Q_1(s,x_i) - \max_{\theta'}\Big[Q_2(\text{crop}(s,x_i),\theta')\Big]\Big)^2, \tag{14}$$

where Q_2 provide targets to train Q_1. This off-policy loss minimizes the gap between Q_1 and Q_2. Our combined loss function is therefore $\mathcal{L} = \mathcal{L}'_1 + \mathcal{L}''_1 + \mathcal{L}_2$.

5.2.2 Prioritizing failure experiences in minibatch sampling

In the contextual bandit setting, we want to avoid the situation where the agent repeats incorrect actions in a row. This can happen because some failure grasps left the scene intact, thus the image and the Q-map are intact. We address this problem by prioritizing failure experiences. When a grasp failed, the experience is included in the sampled minibatch on the next SGD step (Zeng et al., 2018), thereby updating the Q-function prior to reevaluating it on the next time step. The updates in Q reduce the chance of selecting the same (bad) action.

5.2.3 Boltzmann exploration

We find empirically Boltzmann exploration is better compares to ϵ-greedy exploration in our grasp setting. We use a temperature of τ_{training} during training and a lower temperature of τ_{test} during testing. Using a non-zero temperature at test time helped reduce the chances of repeatedly sampling a bad action.

5.2.4 Data augmentation

Even though we are using equivariant neural networks to encode the Q-function, it can still be helpful to perform data augmentation as well. This is because the granularity of the rotation group encoded in q_1 (D_4) is coarser than that of the action space (C_n/C_2). We address this problem by augmenting the data with translations and rotations sampled from $\hat{\text{SE}}(2)$. For each experienced transition, we add eight additional $\hat{\text{SE}}(2)$-transformed images to the replay buffer.

5.2.5 Softmax at the output of q_1 and q_2

Since we are using a contextual bandit with binary rewards and the reward function $r(s, a)$ denotes the parameter of a Bernoulli distribution at s, a, we know that Q_1 and Q_2 must each take values between zero and one. We encode this prior using an element-wise softmax layer at the output of each of the q_1 and q_2 networks.

5.2.6 Selection of the z coordinate

In order to execute a grasp, we must calculate a full x, y, θ, z goal position for the gripper. Since our model only infers a planar grasp pose, we must calculate a depth along the axis orthogonal to this plane (the z axis) using other means. In this paper, we calculate z by taking the average depth over a 5×5 pixel region centered on the grasp point in the input depth image. The commanded gripper height is set to an offset value from this calculated height. While executing the motion to this height, we monitor force feedback from the arm and halt the motion prematurely if a threshold is exceeded.

5.3 Optimizations for transparent object grasping using RGB input

In order to grasp transparent objects using our model, we found it was helpful to make a few small modifications to our model and setup. Most importantly, we found it was essential to use RGB rather than depth-only image input to the model. Bin color: We found that for transparent objects, our system performed much better using a black bin color rather than a white or transparent bin color. Figure 6 illustrates this difference in setup. We believe that the performance difference is due to the larger contrast between the transparent objects and the bin in the RGB spectrum.

Dihedral group in $q2$: Another optimization we used in our transparent object experiments was to implement Eq. 8 using dihedral group D_n which expresses the group of multiples of $2\pi/n$ and reflections, rather than C_n. As in Sect. 5.1.6, we use a quotient group that encodes gripper symmetries. Here, this quotient group becomes D_n/D_2, which pairs orientations separated by π and reflected orientations into the same equivalent class.

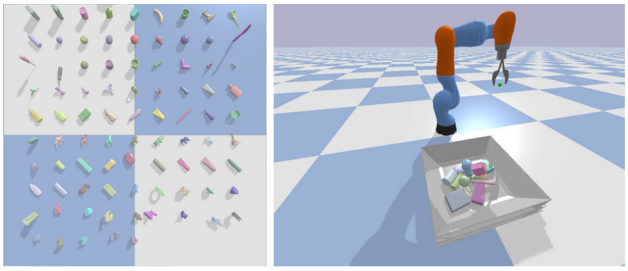
Collision penalty: In our experiments with transparent objects, we found that collision was a more significant problem than it was with opaque objects. We believe this was a result of the fact that since the transparent objects did not completely register in the depth image, standard collision checking between the object point cloud and the gripper did not suffice to prevent collisions. Therefore, in our transparent object experiments, we penalized successful grasps that produced collisions during grasping by awarding those grasps only 0.8 reward instead of the full 1.0 reward.

6 Experiments in simulation

6.1 Setup

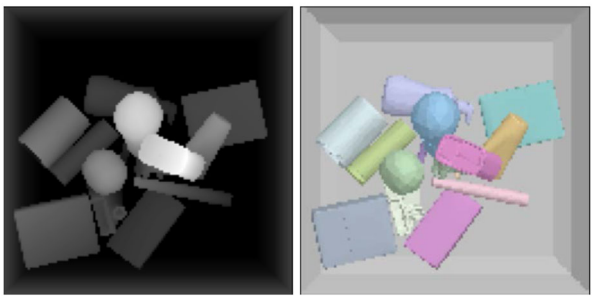
6.1.1 Object set

All simulation experiments are performed using objects drawn from the GraspNet-1Billion dataset (Fang et al., 2020). This includes 32 objects from the YCB dataset (Calli et al., 2017), 13 adversarial objects used in DexNet 2.0 (Mahler et al., 2017), and 43 additional objects unique to GraspNet-1Billion (Fang et al., 2020) (a total of 88 objects). Out of these 88 objects, we exclude two bowls because they can be stably placed in non-graspable orientations, i.e., they can be placed upside down and cannot be grasped in that orientation using standard grippers. Also, we scale these objects so that they



(a) 86 GraspNet-1B objects used

(b) Pybullet simulation



(c) Depth image

(d) RGB image

Fig. 3 **a** The 86 objects used in our simulation experiments are drawn from the GraspNet-1Billion dataset (Fang et al., 2020). **b** Phybullet simulation. **c** and **d** Top-down depth and RGB images of the grasping scene

are graspable from any stable object configuration. Lastly, objects are assigned with random RGB values drawn from uniform distribution $U((0.6, 0.6, 0.6), (1, 1, 1))$. We refer to these 86 mesh models as our simulation "object set", shown in Fig. 3a.

6.1.2 Simulation details

Our experiments are performed in Pybullet (Coumans & Bai, 2016). The environment includes a Kuka robot arm and a 0.3 m × 0.3 m tray with inclined walls (Fig. 3b). At the beginning of each episode, the environment is initialized with 15 objects drawn uniformly at random from our object set and dropped into the tray from a height of 40 cm so that they fall into a random configuration. The state is a depth image (depth modality) or RGB image (RGB modality) captured from a top-down camera (Fig. 3c, d). On each time step, the agent perceives a state and selects an action to execute which specifies the planar pose to which to move the gripper. A grasp is considered to have been successful if the robot is able to lift the object more than 0.1m above the table. The environment will be reinitialized when all objects have been removed from the tray or 30 grasp attempts have been made.

6.2 Comparison against baselines on depth modality

6.2.1 Baseline model architectures

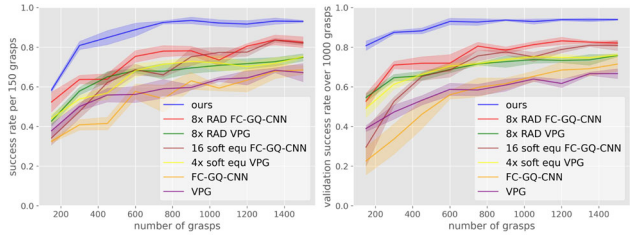
We compare our method against two different model architectures from the literature: VPG (Zeng et al., 2018) and FC-GQ-CNN (Satish et al., 2019). Each model is evaluated alone and then with two different data augmentation strategies (soft equ and RAD). In all cases, we use the contextual bandit formulation described in Sect. 4.2. The baseline model architectures are: VPG: Architecture used for grasping in (Zeng et al., 2018). This model is a fully convolutional network (FCN) with a single-channel output. The Q value of different gripper orientations is evaluated by rotating the input image. We ignore the pushing functionality of VPG. FC-GQ-CNN: Model architecture used in (Satish et al., 2019). This is an FCN with 8-channel output that associates each grasp rotation to a channel of the output. During training, our model uses Boltzmann exploration with a temperature of $\tau = 0.01$ while the baselines use ϵ-greedy exploration starting with $\epsilon = 50\%$ and ending with $\epsilon = 10\%$ over 500 grasps (this follows the original implementation in Zeng et al. (2018)).

6.2.2 Data augmentation strategies

The data augmentation strategies are: $n \times$ RAD: The method from Laskin et al. (2020a) that augments each sample in the mini-batch with respect to Eq. 4. Specifically, for each SGD step, we first draw bs (where bs is the batch size) samples from the replay buffer. Then for each sample, we augment both the observation and the action using a random SE(2) transformation, while the reward is unchanged. We perform n SGD steps on the RAD augmented mini-batch after each grasp sample. $n \times$ soft equ: similar to $n \times$ RAD except that we produce a mini-batch by drawing bs/n samples, randomly augment those samples n times with respect to Eq. 4, then perform a single SGD step. Details can be found in "Appendix Appendix C".

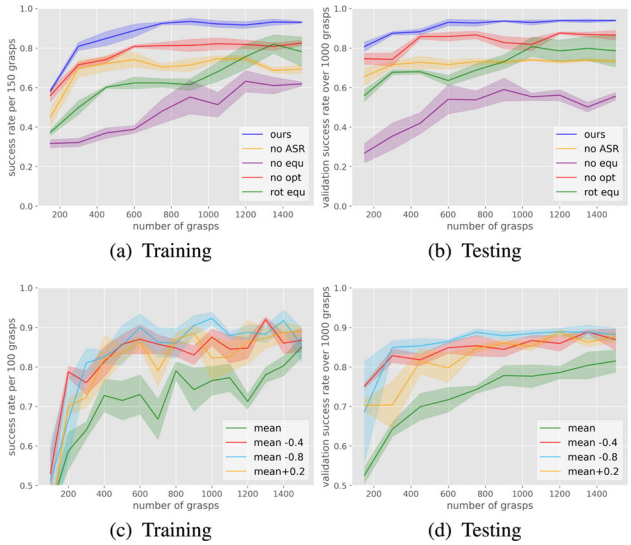
6.2.3 Results and discussion

The learning curves of Fig. 4 show the grasp success rate versus the number of grasping attempts on depth modality. Figure 4a shows online learning performance. Each data point is the average success rate over the last 150 grasps (therefore, the first data point occurs at 150). Figure 4b shows testing performance by stopping training every 150 grasp attempts and performing 1000 test grasps and reporting average performance over these 1000 test grasps. Our method tests at a lower test temperature of $\tau = 0.002$ while the baselines test pure greedy behavior.



(a) Training (b) Testing

Fig. 4 Comparison with baselines. All lines are an average of four runs. Shading denotes standard error. **a** Shows learning curves as a running average over the last 150 training grasps. **b** Shows the average near-greedy performance of 1000 validation grasps performed every 150 training steps



(a) Training (b) Testing

(c) Training (d) Testing

Fig. 5 Ablation study for depth observation. Lines are an average over 4 runs. Shading denotes standard error. In the left column, learning curves as a running average over the last 150 training grasps. In the right column is the average near-greedy performance of 1000 validation grasps performed every 150 training steps. The first row is in the depth modality while the second row is in the RGB modality

Generally, our proposed equivariant model convincingly outperforms the baseline methods and data augmentation strategies with depth modality. In particular, Fig. 4b shows that the testing success rate of the equivariant model after 150 grasp attempts has the same or better performance than all of the baseline methods after 1500 grasp attempts. Notice that each of the two data augmentation methods we consider (RAD and soft equ) has a positive effect on the baseline methods. However, after training for the full 1500 grasp attempts, our equivariant model converges to the highest grasp success rate ($93.9 \pm 0.4\%$). Please see "Appendix Appendix D" for a comparison with longer training horizon.

6.3 Ablation study

There are three main parts of SymGrasp as described in this paper: (1) the use of equivariant convolutional layers instead of standard convolution layers; (2) the use of the augmented state representation (ASR) instead of a single network; (3) the various optimizations described in Sect. 5.2. Here, we evaluate the performance of the method when ablating each of these three parts in the depth modality. For additional ablations, see "Appendix Appendix D".

6.3.1 Baselines

In no equ, we replace all equivariant layers with standard convolutional layers. In no ASR, we replace the equivariant q_1 and q_2 models described in Sect. 3.2 by a single equivariant network. In no opt, we remove the optimizations described in Sect. 5.2. In addition to the above, we also evaluated rot equ which is the same as no ASR except that we replace ASR with a U-net (Ronneberger et al., 2015) and apply $4\times$ RAD (Laskin et al., 2020a) augmentation. Detailed network architectures can be found in "Appendix Appendix A".

6.3.2 Results and discussion

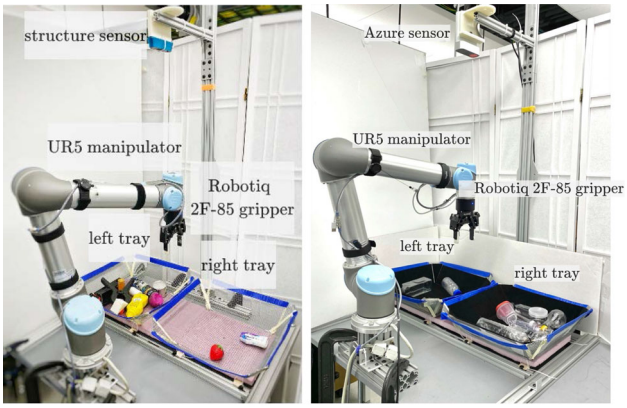
Figure 5a and b shows the results where they are reported exactly in the same manner as in Sect. 6.2. no equ does worst, suggesting that our equivariant model is critical. We can improve on this somewhat by adding data augmentation (rot equ), but this sill underperforms significantly. The other ablations, no ASR and no opt demonstrate that those parts of the method are also important.

6.4 Effect of background color

This experiment evaluates the effect of background color with RGB modality. We compare training with different tray colors which have an offset from the mean RGB value of the objects (0.8, 0.8, 0.8). For example, in mean − 0.8, the tray color would be black (0, 0, 0). We test four different colors from black (mean−0.8) to white (mean+0.2).

6.4.1 Results and discussion

Figure 5c and d show that the contrast between background and object color has a big impact on grasp learning. In particular, the model has the worst performance when the background color is the same as the mean of the object color (mean). The model performs best when the background color has the most significant contrast from the mean of the object color (mean − 0.8).



(a) Opaque objects grasping (b) Transparent objects grasping

Fig. 6 Setup for self-supervised training on the robot

7 Experiments in hardware

7.1 Setup

7.1.1 Robot environment

Our experimental platform is comprised of a Universal Robots UR5 manipulator equipped with a Robotiq 2F-85 parallel-jaw gripper. For depth modality, we use an Occipital Structure Sensor and for RGB modality we use a Kinect Azure Sensor paired with two black trays. The dual-tray grasping environment is shown in Fig. 6. The workstation is equipped with an Intel Core i7-7800X CPU and an NVIDIA GeForce GTX 1080 GPU.

7.1.2 Self-supervised training

The training begins with the 15 training objects (Fig. 9a for opaque object grasping and Fig. 10a for transparent object grasping) being dropped into one of the two trays by the human operator. Then, the robot attempts to pick objects from one tray and drop them on another. The drop location is sampled from a Gaussian distribution centered in the middle of the receiving tray. All grasp attempts are generated by the contextual bandit. When all 15 objects have been transported in this way, training switches to attempting to grasp from the other tray and drop into the first. Whether all objects are transported or not is decided by the heuristic. For opaque objects, we threshold the depth sensor reading. For transparent objects, we compare the RGB image of the current scene with an RGB image of an empty tray, similar to Kalashnikov et al. (2018). During training, the robot performs 600 grasp attempts in this way (that is 600 grasp *attempts*, not 600 successful grasps). A reward r will be set to 1 if the gripper was blocked by the grasped object, otherwise $r = 0$.

7.1.3 In-motion computation

We are able to nearly double the speed of robot training by doing all image processing and model learning while the robotic arm was in motion. This was implemented in Python as a producer-consumer process using mutexs. As a result, our robot is constantly in motion during training and the training speed for our equivariant algorithm is limited by the velocity of robot motion. This improvement enabled us to increase robot training speed from approximately 230 grasps per hour to roughly 400 grasps per hour.

7.1.4 Evaluation procedure

The evaluation begins with the human operator dropping objects into one tray, after this, no human interference is allowed. We evaluate the success rate of robot grasping objects. A key failure mode during testing is repeated failure grasps. To combat this, we use the procedure of Zeng et al. (2018) to reduce the chances of repeated grasp failures. The procedure is that after a grasp failure, we perform multiple SGD steps using that experience to "discourage" the model from selecting the same action and then use that updated model for the subsequent grasp. After a successful grasp, we discard these updates and reload the original network.

All runs are evaluated by freezing the corresponding model and executing 100 greedy (or near greedy) test grasps for each object set in the easy-object test set (Fig. 9b), the hard-object test set (Fig. 9c).

7.1.5 Model details

For all methods, prior to training on the robot, model weights are initialized randomly using an independent seed. No experiences from simulation are used, i.e.,we train from scratch. For our depth algorithm, the q_1 network is defined using D_4-equivariant layers and the q_2 network is defined using C_{16}/C_2-equivariant layers. For ours RGB algorithm, the q_2 network is defined using D_{16}/D_2-equivariant layers. During training, we use Boltzmann exploration with a temperature of 0.01. During testing, the temperature is reduced to 0.002 (near-greedy). For more details, see "Appendix Appendix G".

7.2 Opaque object grasping

7.2.1 Objects

For opaque objects grasping, all training happens using the 15 objects shown in Fig. 9a. After training, we evaluate grasp performance on both the "easy" test objects (Fig. 9b) and the "hard" test objects (Fig. 9c). Note that both test sets are novel with respect to the training set.

Table 1 Evaluation success rate (%), standard error, and training time per grasp t_{SGD} (in seconds) in the hardware experiments

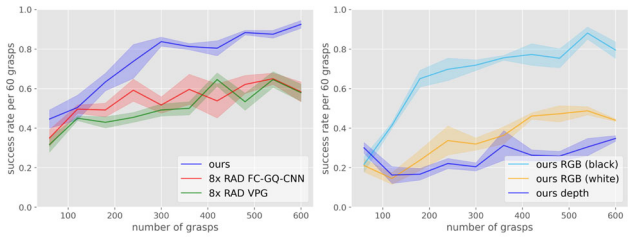
Baseline	Test set easy	Test set hard	t_{SGD}
8× RAD VPG	61.8 ±3.59	52.5 ±5.33	12.1s
8× RAD FC-GQ-CNN	82.8 ±1.65	74.3 ±3.04	1.55s
Ours	95.0 ±1.47	87.0 ±1.87	1.11s

Results are an average of 100 grasps per training run averaged over four runs, performed on the held out test objects shown in Fig. 9b and c

Table 2 Evaluation success rate (%), standard error, and capture time per grasp (in seconds) in the hardware experiments of transparent objects

Baseline	Trans. train set	Trans. test set	t_{capture}
Ours RGB	85.3 ±8.62	83.3 ±3.20	1.8s

Results are an average of 100 grasps per training run averaged over four runs, performed on the train and held out test objects shown in Fig. 10 a and b

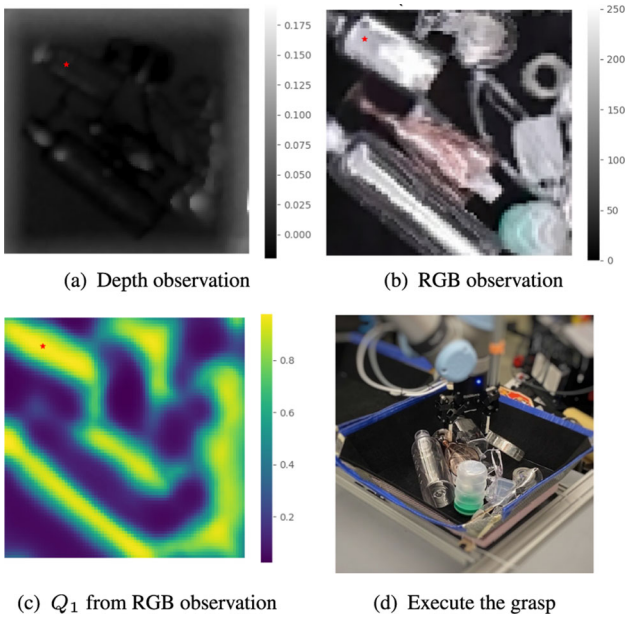


(a) Opaque object grasping (b) Transparent object grasping

Fig. 7 Learning curves for **a** the opaque object grasping and **b** the transparent object grasping hardware experiment, the parenthesis indicates the color of trays. All curves are averaged over 4 runs with different random seeds and random object placement. Each data point in the curve is the average grasp success over the last 60 grasp attempts. Shading denotes standard error

7.2.2 Baselines

In our robot experiments for opaque object grasping, we compare our method against 8 × RAD VPG (Zeng et al., 2018; Laskin et al., 2020a) and 8 × RAD FC-GQ-CNN (Satish et al., 2019; Laskin et al., 2020a), the two baselines we found to perform best in simulation. As before, 8 × RAD VPG, uses a fully convolutional network (FCN) with a single output channel. The Q-map for each gripper orientation is calculated by rotating the input image. After each grasp, we perform 8× RAD data augmentation (8 optimization steps with a mini-batch containing randomly translated and rotated image data). 8 × RAD FC-GQ-CNN also has an FCN backbone, but with eight output channels corresponding to each gripper orientation. It uses 8× RAD data augmentation as well. All exploration is the same as it was in simulation except that the ϵ-greedy schedule goes from 50 to 10% over 200 steps rather than over 500 steps.



(a) Depth observation (b) RGB observation

(c) Q_1 from RGB observation (d) Execute the grasp

Fig. 8 Illustrations for transparent object grasping. **a** Shows the depth observation, notice that most transparent objects are not sensed. **b** Shows the same scene with RGB modality, notice that objects are sensed up to orthographic projection error. **c** Shows the Q_1 map from the observation in (**b**) and the selected action in the red dot. **d** Shows the executed grasp

7.2.3 Results and discussion

Figure 7a shows the learning curves on opaque object grasping for the three methods during learning. An important observation is that the results from training on the physical robot Fig. 7a match the simulation training results Fig. 4a. Figure 7a shows for opaque object grasping, our method achieves a success rate of $>$ 90% after 600 grasp attempts while the baselines are near 70%. Table 1 shows the testing performance and also demonstrates our method significantly outperforms the baselines during testing. However, performance is lower on the "hard" test set. We hypothesize that the lower "hard" set performance is due to a lack of sufficient diversity in the training set. The other observation is that since each of these 600-grasp training runs takes approximately 1.5 h, suggests these methods could efficiently directly learn from real-world data thus avoiding the simulation-to-real-world gap and adapting to physical changes in the robot.

7.3 Transparent object grasping

7.3.1 Objects

For transparent object grasping, all training happens using the 15 objects shown in Fig. 10a. After training, we evaluate grasp performance on the in-distribution training set and out-of-distribution testing objects (Fig. 10a, b). Note that the test set is novel with respect to the training set.

(a) Training set (b) Testing set, easy (c) Testing set, hard

Fig. 9 Object sets used for training and testing. Both training and the test set easy include 15 objects while the test set hard has 20 objects. Objects were curated so that they were graspable by the Robotiq 2F-85 parallel jaw gripper from any configuration and visible to the Occipital Structure Sensor

7.3.2 RGB modality details

The top-down RGB images are orthographic projections of noisy RGB point clouds. RGB values are normalized to fit a Gaussian distribution $\mathcal{N}(0, 0.01)$. During training, each mini-batch of images is augmented in brightness with a (0.9, 1.1) range.

7.4 Baselines

For transparent object grasping, we compare Ours RGB (black), Ours RGB (white), and Ours D. Ours RGB (black) is the baseline in our proposed methods, Ours RGB (white) changes black trays to white trays, see Fig. 18. Ours D is the best baseline in physical opaque objects grasping experiments, here we train and test on transparent objects.

7.5 Results and discussion

Figure 7b shows the learning curves for transparent object grasping and Table 2 shows the grasping performance of ours RGB (black) on the transparent object set. Figure 8 shows a transparent object grasping process during evaluation.

Figure 7b shows that for transparent object grasping, ours RGB (black) produces a significant ($> 40\%$) improvement in success rate compared to ours depth, suggesting that the RGB information is needed to perform well in this setting. The background color is also important: changing the tray from black to white leads to a $> 30\%$ decrement in success rate. Table 2 summarizes that ours RGB achieves $> 80\%$ success rate on both the training set and testing set, indicating the method learns a good grasp function of in-distribution object set and generalizes to novel transparent objects. Also, notice that from a computational perspective, our method is significantly cheaper (1.8 s) than NeRF based approaches to transparent object grasping like EVO Nerf Kerr et al. (2022) and Dex Nerf (Ichnowski et al., 2021) which take 9.5 s and 16 s, respectively (Figs. 9, 10).

(a) Trans. training set (b) Trans. testing set

Fig. 10 Transparent object sets used for training and testing. The Training set includes 15 objects and the testing set includes 10 objects. Objects were curated so that they were graspable by the Robotiq 2F-85 parallel jaw gripper from any configuration

8 Conclusions and limitations

Our main contribution is to recognize that planar grasp detection is SE(2)-equivariant and to leverage this structure using an SO(2)-equivariant model architecture. This model is significantly more sample efficient than other grasp-learning methods and can learn a good grasp function with less than 600 grasp samples. This increase in sample efficiency enables us to learn to grasp on a physical robotic system in a practical amount of time (approximately and hour and a half) without relying on any sort of pretraining or semi-supervised learning. Training completely on physical systems is important for two main reasons. First, it eliminates the need to train in simulation, thereby avoiding the sim2real gap. Second, it enables our system to adapt to changes or idiosyncrasies of the robot hardware or the physical environment that would be hard to simulate.

A key limitation in both simulation and the real world is that despite the fast learning rate, grasp success rates (after training) still seems to be limited to the low/mid 90% for opaque object and 80% for transparent objects. This is the same success rate seen in with other grasp detection methods (Mahler et al., 2017; ten Pas et al., 2017b; Mousavian et al., 2019), but it is disappointing here because one might expect faster adaptation to lead ultimately to better grasp performance. This could simply be an indication of the complexity of the grasp function to be learned or it could be a result of stochasticity in the simulator and on the real robot. Another limitation of our work is that it is limited to an open-loop control setting where the model infers only a goal pose. Nevertheless, we expect that the equivariant model structure leverage in this paper would also be useful in the closed loop setting, as suggested by the results of Wang et al. (2022).

Author Contributions XZ proposed the method in discussions with DW. XZ and GS conducted the robot experiment. All authors wrote and reviewed the manuscript.

Funding Open access funding provided by Northeastern University Library This work was supported in part by NSF 1724257, NSF

1724191, NSF 1763878, NSF 1750649, NASA 80NSSC19K1474, the Roux Institute, the Harold Alfond Foundation, NSF Grants 2107256, and 2134178.

Availability of data and materials Code is available at https://github.com/ZXP-S-works/SE2-equivariant-grasp-learning.

Declarations

Ethical approval Not applicable.

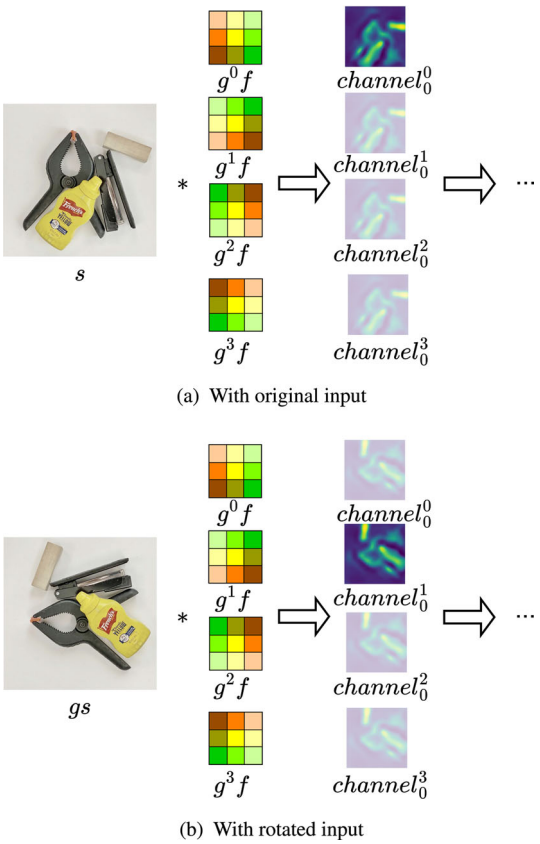
Conflict of interest The authors declare that they have no relevant financial or non-financial interests to disclose

Appendix A Neural network architecture

Our network architecture is shown in Fig. 12a. The $q1$ network is a fully convolutional UNet (Ronneberger et al., 2015). The $q2$ network is a residual neural network (He et al., 2015). These networks are implemented using PyTorch (Paszke et al., 2019), and the equivariant networks are implemented using the E2CNN library (Weiler & Cesa, 2019). Adam optimizer (Kingma & Ba, 2015) is used for the SGD step. The ablation no opt has the same architecture as above. The ablation no asr (Fig. 12b) ablated the $q2$ network and is defined with respect to group $C16$. The ablation no equ (Fig. 12c) has a similar network architecture as ours with approximately the same number of free weights. However, the equivariant network is replaced with an FCN. The ablation rot equ (Fig. 12d) has a similar network architecture as no asr method with approximately the same number of free weights. However, the equivariant network is replaced with an FCN.

Appendix B Examples of equivariant layer

Figure 11 shows an example of an equivariant layer with $C4$ group that maps trivial representation to regular representation. In Fig. 11a the equivariant layer will initialize a convolutional filter f, then rotates the filter by each group element to obtain f, gf, g^2f, g^3f. In Fig. 11b since each



(a) With original input

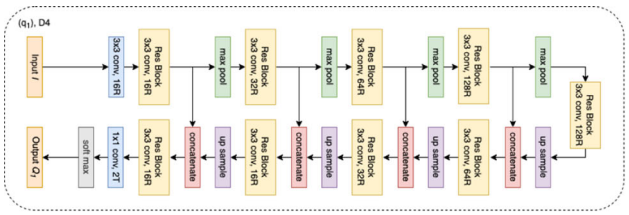
(b) With rotated input

Fig. 11 An example of $C4$ equivariant layer that maps 3 channels of trivial representation to 1 channel of regular representation. s: the 3 trivial channel RGB image. f: the 3 convolutional filter. $g \in C4$: the group action that rotates the signal 90 degrees. $channel_0^i$: one regular channel in the neural network with $i = 0, 1, 2, 3$ fiber channels for each group element in $C4$. $*$ stands for standard convolution in CNN. Rotating image s by g leads to rotating the $channel_0$ spatially by g and shifting the $channel_0$ along fiber channel by 1

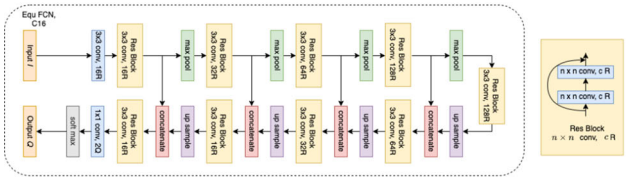
filter is a rotated copy of f and is correspond to a fiber channel in the regular representation, rotating image s by g leads to rotating the $channel_0$ spatially by g and shifting the $channel_0$ along the fiber channel by 1. There could be more layers following this layer, as denoted by arrows and ellipsis. When each layer is equivariant, the entire neural network is equivariant.

Appendix C Augmentation baseline choices

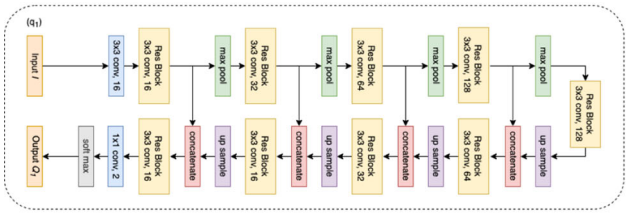
The data augmentation strategies are: $n \times$ RAD: The method from Laskin et al. (2020a) where we perform n SGD steps after each grasp sample, where each SGD step is taken over a mini-batch bs of samples that have been randomly translated and rotated by $g \in \mathrm{SE}(2)$ according to Eq. 4. Specifically, applying random SE(2) transformation g for both observation and action: (gs, ga, r). $n \times$ soft equ: a data augmentation method (Wang et al., 2021) that performs n



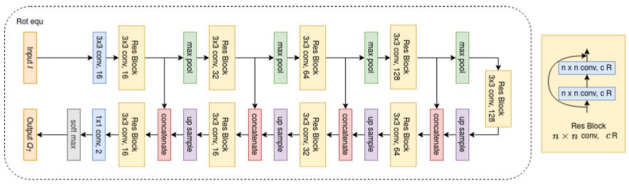
(a) ours and no opt



(b) no asr



(c) no equ



(d) Rot equ

Fig. 12 The neural network architecture for ours and ablations. R means regular representation, T means trivial representation, and Q means quotient representation

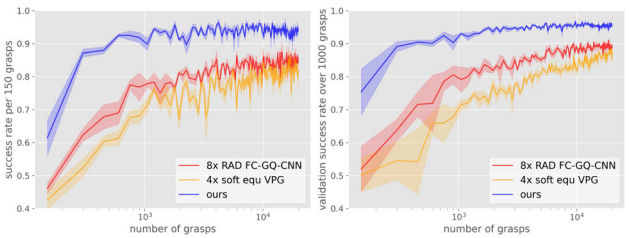
soft equivariant SGD steps per grasp, where each SGD step is taken over n times randomly $\hat{\mathrm{SE}}(2)$ augmented mini-batch. Specifically, we sample bs/n samples (bs is the batch size), augment it n times, and train on this mini-batch. We perform these SGD steps n times so that bs transitions are sampled. This augmentation aims at achieving equivariance in the mini-batch.

We apply $n\times$ RAD and $n\times$ soft equ data augmentation to both VPG and FC-GQ-CNN baselines, with $n = 2, 4, 8$. The best data augmentation parameters n are chosen for each baseline in the comparison in Fig. 4.

Appendix D Additional experiments

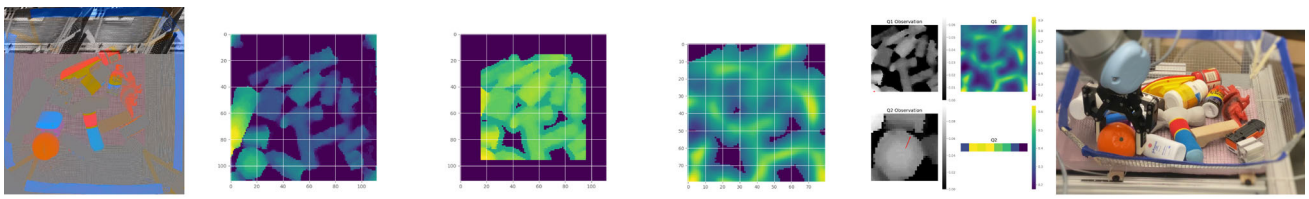
To investigate the sample efficiency of SymGrasp, we compare our method with the other two best methods in Fig. 4. We train all the methods with 20k grasps to illustrate the sample efficiency. As shown in Fig. 14, our method not only learns faster but also converges to a better success rate, compared with the other two baselines.

We then ablate each components in Sect. 5.2. Baselines are: In ASR loss, both q_1 and q_2 minimize $l2$ loss between prediction and the reward r. In No prioritizing, the mini-batch is uniformly sampled from the replay buffer. In e greedy, we replace Boltzmann exploration with e greedy exploration that linearly anneals from 0.5 to 0.1 in 500



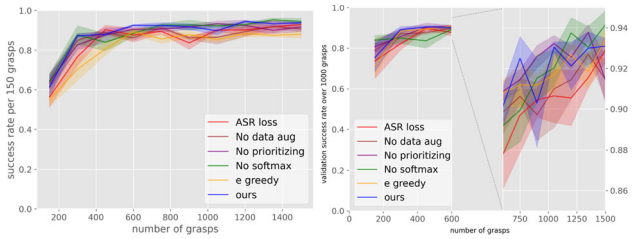
(a) Training (b) Testing

Fig. 14 Additional comparison with baselines for Sect. 6.2 with depth modality. Lines are an average over 4 runs. Shading denotes standard error. **a** Learning curves as a running average over the last 150 training grasps. **b** Average near-greedy performance of 1000 validation grasps performed every 150 training steps



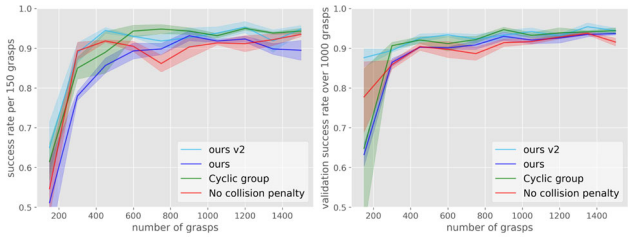
(a) (b) (c) (d) (e) (f)

Fig. 13 Action space constraint for action selection. **a** The test set easy cluttered scene. **b** The state s. **c** The action space x_{positive}, overlays the binary mask x_{positive} with the state s for visualization. **d** The Q-values within the action space. **e** Selecting an action. **f** Executing a grasp



(a) Training (b) Testing

Fig. 15 Additional ablation study for Sect. 5.2 with depth modality. Lines are an average over 4 runs. Shading denotes standard error. **a** Learning curves as a running average over the last 150 training grasps. **b** Average near-greedy performance of 1000 validation grasps performed every 150 training steps



(a) Training (b) Testing

Fig. 16 Additional ablation study for Sect. 5.3 with depth modality. Lines are an average over 4 runs. Shading denotes standard error. **a** Learning curves as a running average over the last 150 training grasps. **b** Average near-greedy performance of 1000 validation grasps performed every 150 training steps

grasps. In No data aug, no data augmentation is performed. In No softmax the pixel-wise softmax in the last layer of q_1 and q_2 is removed.

Figure 15a, b, shows the learning results. Figure 15b shows that ASR loss affects the performance the most, indicating the importance of the loss function in Sect. 5.2.1. Other components No data aug, e greedy slightly reduces the performance indicating the marginal benefits of data augmentation and Boltzmann exploration. Lastly, No softmax performs badly at the 600 grasps and No prioritizing performs badly at the 1500 grasps, these suggest softmax and prioritizing failure grasp stabilizes learning.

We then ablate each component in Sect. 5.3. Baselines are: In Cyclic group, we replace $D16$ group with $C16$ group in $q2$. In No collision penalty, we replace collision penalty by using binary grasp success reward.

Figure 16a, b, shows the learning results. Cyclic group shows Cyclic group in q_2 leads to slower learning before 300 grasp, compares to Dihedral group in q_2; No collision penalty shows collision penalty encourages quicker and better convergence.

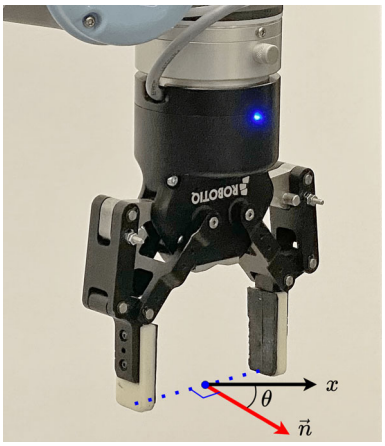
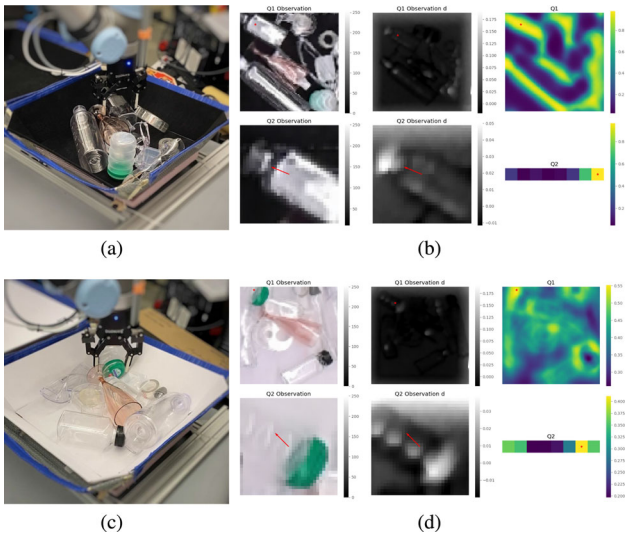


Fig. 17 The definition of θ. x is the x-axle of the workspace while $\vec{n}$ is the normal of the gripper



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

Fig. 18 Transparent object grasping scenes. First row: using black trays. Second row: using white trays

Appendix E Action space details

The action θ is defined as the angle between the normal vector $\vec{n}$ of the gripper and the x-axle, see Fig. 17.

While the grasping height is selected by heuristic, the grasping aperture is fixed with the maximum aperture of the gripper.

To prevent the grasps in the empty space where there is no object in s, we constrain the action space to $x_{\text{positive}} \in X$ to exclude the empty space, see Fig. 13c. The constrain x_{positive} is achieved by first thresholding the depth image: $s_{\text{positive}} = s > s_{\text{threshold}}$ ($s_{\text{threshold}}$ is 0.5 cm in simulation and 1.5 cm in hardware), then dilating this binary map s_{positive} by radius $d_{\text{dilation}} = 4$ pixels. The parameter $s_{\text{threshold}}$ is selected according to sensor noise where d_{dilation} is related to the half

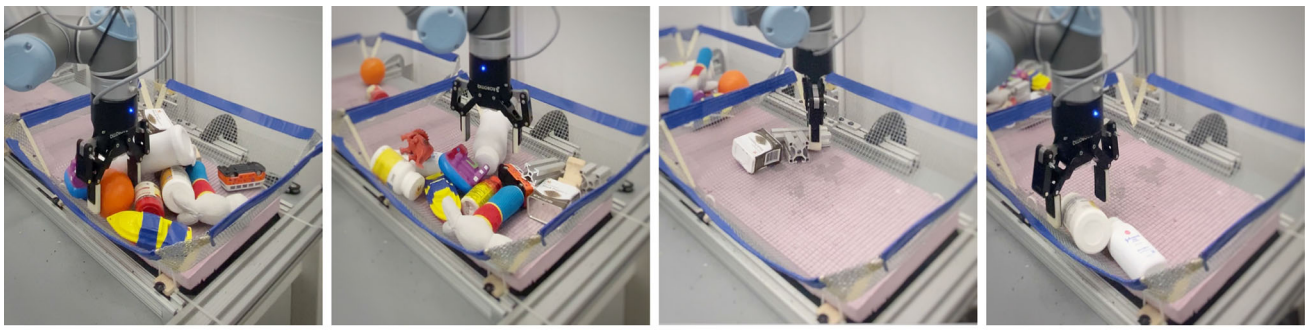
of the gripper aperture. Moreover, we constrain the action space within the tray to prevent collisions.

Appendix F Success and failure modes

We list typical success and failure modes to evaluate the performance of SymGrasp.

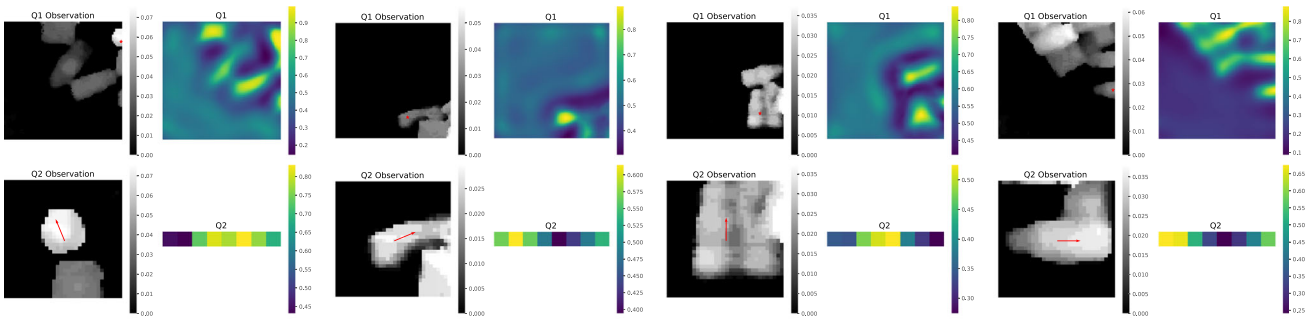
For success modes, the trained policy of SymGrasp showcases its capability. At the densely cluttered scene, SymGrasp prefers to grasp the relatively isolated part of the objects, see Fig. 19a, b. At the scene where the objects are close to each other, SymGrasp can find the grasp pose that is collision free with other objects, see Fig. 19c, d.

For failure modes, we identify several typical scenarios: Bad grasp pose, either wrong translation or orientation grasp pose (Fig. 20a, b, e) indicates that there is a clear gap between our method and optimal policy. This might be caused by the biased dataset collected by the algorithm. Reasonable grasps failure (Fig. 20d, f) means that the agent selects a reasonable grasp, but it fails due to the stochasticity of the real world, i.e., limited sensor resolution, contact dynamics, hardware calibration error, etc. Challenging scenes (Fig. 20c, g) is the nature of densely cluttered objects, it can be alleviated by

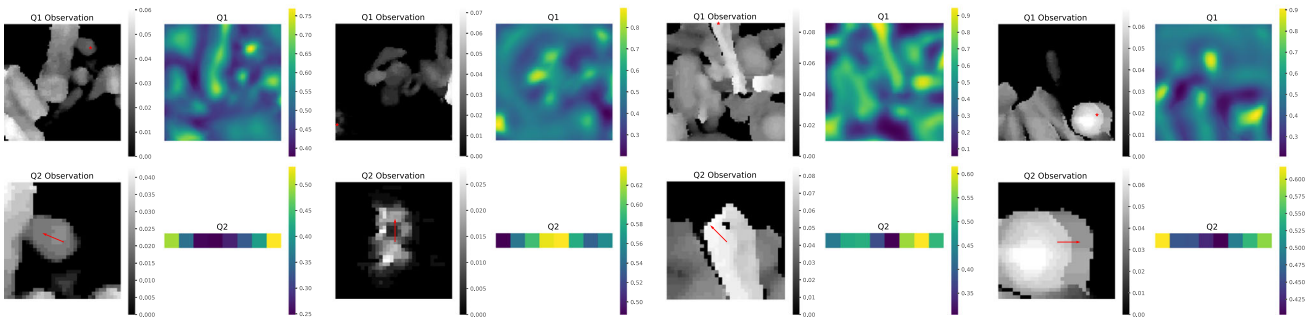


(a) (b) (c) (d)

Fig. 19 Success modes in the test set easy, which has 15 hold out objects



(a) Wrong q_1 (10/20) (b) Wrong q_2 (3/20) (c) Challenging scenes (3/20) (d) Reasonable grasps (3/20)



(e) Wrong q_1 (17/52) (f) Reasonable grasps (13/52) (g) Challenging scenes (11/52) (h) Sensor distortion (8/52)

Fig. 20 Failure modes. The brackets show the failure times divided by the total number of failures in all four runs. The first row is test in the test set easy while the second row is test in the test set hard

learning an optimal policy or synthesize higher DoFs grasps, e.x., predicting grasping high instead of using heuristic. The sensor distortion (Fig. 20h) is caused by an low quality sensor or occlusion. Among all failure modes, wrong action selection takes the most part (65% failure in the test set easy and 33% failure in the test set hard). It is followed by reasonable grasps, challenging scenes, and then sensor distortion.

Appendix G Evaluation details in hardware

The evaluation policy and environment are different from that of training in the following aspects. First, for all methods, the robot arm moves slower than that during training in the environment. This helps form stable grasps. Second, for our method, the evaluation policy uses a lower temperature ($\tau_{\text{test}} = 0.002$) than training. After a failure grasp, ours performs 2 SGD steps on this failure experience. The network weight will be reloaded after recovery from the failure (Zeng et al., 2018). For the baselines, the evaluation policy uses a greedy policy. After a failure grasp, baselines perform 8 RAD SGD steps on this failure experience. The network weight will be reloaded after recovery from the failure (Zeng et al., 2018).

References

Berscheid, L., Friedrich, C., & Kröger, T. (2021). Robot learning of 6 dof grasping using model-based adaptive primitives. arXiv preprint arXiv:2103.12810

Breyer, M., Chung, J. J., Ott, L., Siegwart, R., & Nieto, J. (2021). Volumetric grasping network: Real-time 6 dof grasp detection in clutter. arXiv preprint arXiv:2101.01132

Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., & Dollar, A. M. (2017). Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research, 36*(3), 261–268. https://doi.org/10.1177/0278364917700714

Cohen, T., Geiger, M., & Weiler, M. (2018). A general theory of equivariant cnns on homogeneous spaces. arXiv preprint arXiv:1811.02017

Cohen, T., & Welling, M. (2016a). Group equivariant convolutional networks. In *International conference on machine learning* (pp. 2990–2999). PMLR.

Cohen, T. S, & Welling, M. (2016b). Steerable cnns. arXiv preprint arXiv:1612.08498

Coumans, E., & Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*.

Danielczuk, M., Balakrishna, A., Brown, D. S, Devgon, S., & Goldberg, K. (2020). Exploratory grasping: Asymptotically optimal algorithms for grasping challenging polyhedral objects. arXiv preprint arXiv:2011.05632

Depierre, A., Dellandréa, E., & Chen, L. (2018). Jacquard: A large scale dataset for robotic grasp detection. *CoRR*, arXiv:1803.11469

Fang, H.-S., Wang, C., Gou, M., & Lu, C. (2020). Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11444–11453).

Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., & Schmid, C. (2020). Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11525–11533).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition.

Huang, H., Wang, D., Zhu, X., Walters, R., & Platt, R. (2022). Edge grasp network: A graph-based se (3)-invariant approach to grasp detection. arXiv preprint arXiv:2211.00191

Ichnowski, J., Avigal, Y., Kerr, J., & Goldberg, K. (2021). Dex-nerf: Using a neural radiance field to grasp transparent objects. arXiv preprint arXiv:2110.14217

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., & Bousmalis, K. (2019). Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12627–12637).

Jiang, Z., Zhu, Y., Svetlik, M., Fang, K., & Zhu, Y. (2021). Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. arXiv preprint arXiv:2104.01542

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., & Levine, S. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, arXiv:1806.10293

Kerr, J., Fu, L., Huang, H., Avigal, Y., Tancik, M., Ichnowski, J., Kanazawa, A., & Goldberg, K. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th annual conference on robot learning*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, arXiv:1412.6980

Kofinas, M., Nagaraja, N., & Gavves, E. (2021). Roto-translated local coordinate frames for interacting dynamical systems. *Advances in Neural Information Processing Systems, 34*, 6417–6429.

Kostrikov, I., Yarats, D., & Fergus, R. (2020). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *CoRR*, arXiv:2004.13649

Kumra, S., Joshi, S., & Sahin, F. (2019). Antipodal robotic grasping using generative residual convolutional neural network. *CoRR*, arXiv:1909.04810

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., & Srinivas, A. (2020a). Reinforcement learning with augmented data. *CoRR*, arXiv:2004.14990

Laskin, M., Srinivas, A., & Abbeel, P. (2020b). Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning* (pp. 5639–5650). PMLR.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research, 37*(4–5), 421–436.

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

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM, 65*(1), 99–106.

Mondal, A. K., Nair, P., & Siddiqi, K. (2020). Group equivariant deep reinforcement learning. arXiv preprint arXiv:2007.03437

Morrison, D., Corke, P., & Leitner, J. (2018). Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. arXiv preprint arXiv:1804.05172

Mousavian, A., Eppner, C., & Fox, D. (2019). 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, October.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates Inc.

Pinto, L., & Gupta, A. (2015). Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, arXiv:1509.06825

Platt, R. (2023). Grasp learning: Models, methods, and performance. *Annual Review of Control, Robotics, and Autonomous Systems, 6*, 363–389.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, arXiv:1505.04597

Sajjan, S., Moore, M., Pan, M., Nagaraja, G., Lee, J., Zeng, A., & Song, S. (2020). Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 3634–3642). IEEE.

Satish, V., Mahler, J., & Goldberg, K. (2019). On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters, 4*(2), 1357–1364. https://doi.org/10.1109/LRA.2019.2895878

Sharma, S., Suresh, A., Ramesh, R., & Ravindran, B. (2017). Learning to factor policies and action-value functions: Factored action space representations for deep reinforcement learning. arXiv preprint arXiv:1705.07269

Song, S., Zeng, A., Lee, J., & Funkhouser, T. (2020). Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters, 5*, 4978–4985.

Sundermeyer, M., Mousavian, A., Triebel, R., & Fox, D. (2021). Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. arXiv preprint arXiv:2103.14127

ten Pas, A., Gualtieri, M., Saenko, K., & Platt, R. Jr. (2017a). Grasp pose detection in point clouds. *CoRR*, arXiv:1706.09911

ten Pas, A., Gualtieri, M., Saenko, K., & Platt, R. (2017b). Grasp pose detection in point clouds. *The International Journal of Robotics Research, 36*(13–14), 1455–1473.

van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., & Welling, M. (2020). Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems, 33*, 4199–4210.

van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748

Walters, R., Li, J., & Yu, R. (2020). Trajectory prediction using equivariant continuous convolution. arXiv preprint arXiv:2010.11344

Wang, D., Kohler, C., & Platt, R. Jr. (2020a). Policy learning in SE(3) action spaces. *CoRR*, arXiv:2010.02798

Wang, D., Walters, R., & Platt, R. (2022). SO(2)-equivariant reinforcement learning. In *Int'l conference on representation learning*. https://openreview.net/forum?id=7F9cOhdvfk

Wang, R., Walters, R., Yu, R. (2020b). Incorporating symmetry into deep dynamics models for improved generalization. arXiv preprint arXiv:2002.03061

Wang, D., Walters, R., Zhu, X., & Platt, R. (2021). Equivariant q learning in spatial action spaces. In *5th annual conference on robot learning*. https://openreview.net/forum?id=IScz42A3iCI

Weiler, M., & Cesa, G. (2019). General e(2)-equivariant steerable cnns. *CoRR*, arXiv:1911.08251

Weng, T., Pallankize, A., Tang, Y., Kroemer, O., & Held, D. (2020). Multi-modal transfer learning for grasping transparent and specular objects. *IEEE Robotics and Automation Letters, 5*(3), 3791–3798.

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., & Funkhouser, T. A. (2018). Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *CoRR*, arXiv:1803.09956

Zhan, A., Zhao, P., Pinto, L., Abbeel, P., & Laskin, M. (2020). A framework for efficient robotic manipulation. *CoRR*, arXiv:2012.07975

Zhou, X., Lan, X., Zhang, H., Tian, Z., Zhang, Y., & Zheng, N. (2018). Fully convolutional grasp detection network with oriented anchor box. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 7223–7230). IEEE.

Zhu, X, Wang., Dian, B., Ondrej, S., Guanang, W., Robin, P., Robert. (2022). *Proceedings of Robotics: Science and Systems (RSS)*.

Xupeng Zhu is a Ph.D. candidate at Northeastern University in the Helping Hands lab, advised Prof. Robert Platt. His research focuses on reinforcement learning, geometric deep learning, and manipulation policy learning. Prior to pursuing his Ph.D., he obtained an M.S. in Electrical and Electronics Engineering from Northeastern University in Boston, USA, and a B.E. in Automation from South China University of Technology in Guangzhou, China. He serves as reviewer for Conference of Robot Learning, Transactions on Robotics, CVPR 3D Vision and Robotics Workshop, and RSS Symmetry Workshop.

Dian Wang is a Ph.D. candidate in Computer Science at the Khoury College of Computer Sciences, Northeastern University, advised by Professor Robert Platt and Professor Robin Walters. Prior to his Ph.D., Dian received his Master's degree in Computer Science from Northeastern University in 2019 and his Bachelor's degree in Computer Science and Technology from Sichuan University, Chengdu, China, in 2017. His research interests include Machine Learning and Robotics. Recently, his research has focused on applying equivariant machine learning methods to robotic manipulation to improve learning efficiency. He regularly served as a reviewer for robotic and machine learning conferences and organized the workshop on Symmetries in Robot Learning at Robotics: Science and Systems (RSS) 2023.

Guanang Su is a master student in Robotics at Northeastern University. She joined Helping Hands Lab as a research assistant in 2021. Her current research includes topics like reinforcement learning, imitation learning, and symmetric features. Before that, Guanang received her Bachelor's degree in Computer Engineering at Virginia Tech. She will continue her research journey in RPM Lab at the University of Minnesota starting in Fall 2023.

Ondrej Biza is a Ph.D. candidate in Computer Science at the Khoury College of Computer Sciences, Northeastern University, advised by Professors Robert Platt, Lawson L.S. Wong and Jan-Willem van de Meent. Prior to his Ph.D., Ondrej received his Master's degree in Computer Science from Northeastern University in 2022 and his Bachelor's degree in Information Technology from the Czech Technical University in Prague, in 2019. Ondrej's research focuses on learning structured representations for robotic manipulation, equivariance and few-shot learning.

Robin Walters is an assistant professor in the Khoury College of Computer Sciences at Northeastern University. Robin's research seeks to develop a fundamental understanding of the role symmetry plays in deep learning and to exploit this to improve the generalization and data efficiency of deep learning methods. He has applied these methods to improve models in domains with complex dynamics including climate science, transportation, and robotics.

Robert Platt is an Associate Professor in the Khoury College of Computer Sciences at Northeastern University. He works at the intersection of machine learning and robotic manipulation with a focus on reliability and adaptability in unstructured environments. Prior to coming to Northeastern, he was a Research Scientist at MIT and a technical lead at NASA Johnson Space Center.