2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

SEIL: Simulation-augmented Equivariant Imitation Learning

Mingxi Jia*, Dian Wang*, Guanang Su, David Klee, Xupeng Zhu, Robin Walters, Robert Platt
Khoury College of Computer Sciences
Northeastern University
{jia.ming, wang.dian, su.gu, klee.d, zhu.xup, r.walters, r.platt}@northeastern.edu

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161252

***Abstract*— In robotic manipulation, acquiring samples is extremely expensive because it often requires interacting with the real world. Traditional image-level data augmentation has shown the potential to improve sample efficiency in various machine learning tasks. However, image-level data augmentation is insufficient for an imitation learning agent to learn good manipulation policies in a reasonable amount of demonstrations. We propose Simulation-augmented Equivariant Imitation Learning (SEIL), a method that combines a novel data augmentation strategy of supplementing expert trajectories with simulated transitions and an equivariant model that exploits the $\mathrm{O}(2)$ symmetry in robotic manipulation. Experimental evaluations demonstrate that our method can learn non-trivial manipulation tasks within ten demonstrations and outperform the baselines by a significant margin.**

I. INTRODUCTION

Due to the high dimensionality of state and action spaces in robotic manipulation, learning effective policies often requires a significant amount of data. When policy learning is modeled using reinforcement learning [1], the agent needs to learn from trial and error while interacting with the environment. This process typically requires thousands or millions of time steps, which is extremely expensive even in a simulator. An alternative approach is to model policy learning as imitation learning [2], [3] by cloning an expert's behavior in a supervised way. Even then, the agent still needs hundreds of episodes of data to learn a simple task like object picking [4].

One way to combat the problem of high data requirements is to use data augmentation to improve sample efficiency, i.e., helping the neural network to learn more general features from a limited amount of data. Recently, data augmentation has led to many successes in both reinforcement learning [5] and imitation learning [6] because it not only creates extra training samples, but also acts like a regularizer to prevent neural networks from overfitting [7]. However, current data augmentation techniques are typically implemented through image transformation [8] including cropping, blurring, rotating, etc, limiting their potential in 3D robotic manipulation tasks.

In this work, we propose a novel expert data augmentation method that we call **T**ransition **S**imulation (TS). Our method simulates new expert state-action pairs within the vicinity of existing expert transitions by projecting observation images from the observed point cloud (Figure 1), resulting in more diverse augmentations compared with standard data augmentation techniques. Transition Simulation can also be combined with image-level data augmentation to further improve the sample efficiency. In addition, we combine Transition Simulation with equivariant policy learning, utilizing the existing $\mathrm{O}(2)$ (i.e., planar rotation and reflection) symmetry in robotic manipulation to make the policy automatically generalize over different $\mathrm{O}(2)$ transformed states. Our proposed algorithm, **S**imulation-augmented **E**quivariant **I**mitation **L**earning (SEIL), achieves few-shot imitation learning using ten or fewer demonstrations in non-trivial manipulation tasks.

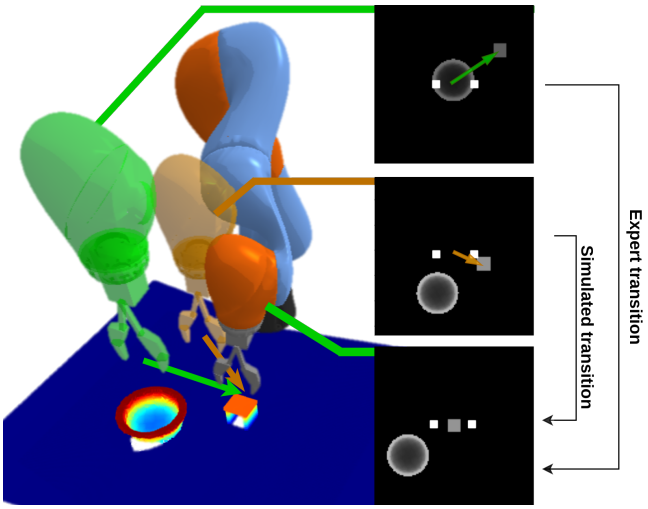


Fig. 1. **Overview of Transition Simulation.** Our method generates new expert transitions by projecting the observed point cloud (bottom) to an observation image (right) in a simulated arm pose (orange). Green shows the existing expert transition, and orange shows the simulated transition.

Our contributions can be summarized as three-fold. First, we propose a novel data augmentation method, Transition Simulation, that augments the expert data in imitation learning by simulating new expert transitions. Second, we combine Transition Simulation with equivariant policy learning and propose the SEIL method that achieves extremely high sample efficiency in robotic manipulation imitation learning. Last, we evaluate SEIL on a set of challenging manipulation tasks both in simulation and in the real world, and demonstrate a significant improvement over competing baselines. Supplementary video and code are available at `https://saulbatman.github.io/project/seil/`.

*Equal contribution

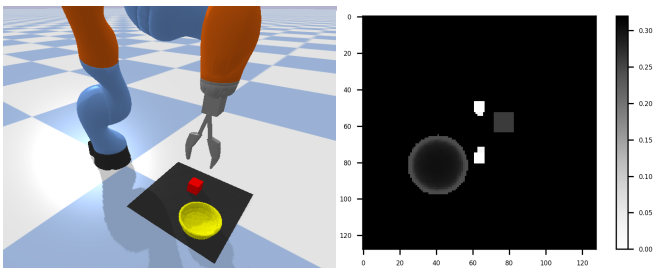
II. RELATED WORK

Imitation learning or Behavior Cloning (BC) has been widely used in robotics. The traditional BC algorithms typically learn an explicit model that maps from the state space to the action space in the framework of supervised learning. Such an approach has been applied to autonomous driving [9] and robotic manipulation [10], [11], [12], [13], [14]. Recently, implicit behavior cloning [3] is the first behavioral cloning algorithm to use an energy-based model (EBM), showing impressive results for learning discontinuous behaviors in robotic manipulation. However, implicit models need large amounts of data and careful hyperparameter tuning for their sampling strategy. Compared with prior works, this paper proposes a novel method to improve imitation learning through simulating new expert transitions.

Manipulation learning is a challenging problem because of the high dimensionality and complexity of the robot arm actions. One way to combat that challenge is to use open-loop control with pre-defined action primitives (e.g., pick, place, push, etc.) to reduce the action space [13], [15]. On the other hand, learning in a closed-loop manner [16], [17], [18], [19] is appealing because it allows the agents to behave more freely without being constrained by the pre-defined action choices. However, it also makes the policy harder to learn because of the continuous action space and the longer time horizon brought on by the nature of the closed-loop control. This paper solves such a problem though improving the expert data sample efficiency, thus providing more guidance for closed-loop policy learning.

Data augmentation is an indispensable technique in deep learning to improve sample efficiency. Traditionally, data augmentation means image-level augmentation like rotation, reflection [20], and translation [13]. In particular, random crops [5] have been shown to improve performance in image classification [20], reinforcement learning [21], and self-supervised learning [22]. However, image-level augmentation is not generalizable to the 3D space in which the manipulation agent operates. Some recent works perform novel data augmentation in the 3D space. [23] augments the offline dataset by learning a reverse dynamics model. [24] proposes to apply rigid body transformations for states and actions, but requires the ground-truth states of the objects, which makes it highly reliant on high-precision state estimation algorithms in real-world scenarios. [10] augments expert trajectory by connecting intermediate points to keyframes, but defines a keyframe decision function manually, which may possibly lead to inaccurate augmentations. In such cases, RL agents can recover from such failures due to their exploration strategy, but a behavioral cloning agent will suffer from sub-optimality. Compared with the prior works, our method implements data augmentation in the 3D space, does not depend on any state estimators, and ensures the quality of the augmented expert transitions by simulating them near the existing expert transitions.

Data efficiency is important in terms of the need to avoid labor-heavy data labeling. Besides data augmentation, another approach is to pre-train neural networks with self-supervised learning methods like contrastive learning [25] and auto-encoders [26]. Such approaches are widely adopted by manipulation works [27], [28], [29]. Recent works in geometric machine learning show that one can improve the data efficiency more directly by enforcing symmetries using equivariant models [30], [31]. Several works have applied equivariant models in robotic manipulation and show promising results [32], [33], [14], [34]. This works extends the prior works by applying equivariant models in conjunction with Transition Simulation to further improve sample efficiency.



(a) PyBullet environment (b) Observation in simulation

Fig. 2. **Environments and observations** (a) Block in Bowl Scene in PyBullet simulation; (b) Depth image observation.

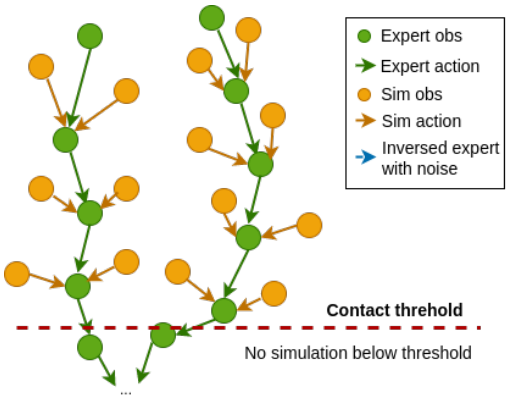
III. PROBLEM STATEMENT

We consider the problem of manipulating (e.g., pick & place, pushing, etc.) objects in a 3D space in a closed-loop manner. This problem can be formulated as learning a neural model mapping a state to the desired delta actions in a 3D workspace $\mathcal{W} \subseteq \mathbb{R}^3$. Let $\mathcal{T} \subseteq \mathcal{W}$ denote the subset of our workspace that is free of objects.

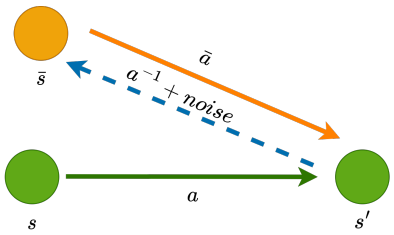
State space: State is expressed as a tuple $s = (C, p, r)$ where $C \in \mathbb{R}^{n\times 3}$ is a point cloud, $p \in \mathbb{R}^d \subseteq \mathbb{R}^6$ is the $d \leq 6$ degrees of freedom gripper pose, and r is the other robot state expressed arbitrarily. We assume that the robot arm is segmented out from the point cloud C.

Action space: The actions are represented by $a = (\lambda, \Delta p)$ where λ is the gripper open width, and Δp is the delta movement of the gripper with respect to the current pose p.

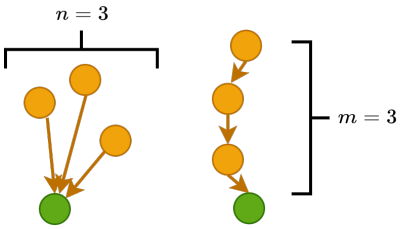
Five DoF imitation learning with an image observation: In this work, we focus on gripper poses with $d = 4$ degrees of freedom (i.e., $p = (x, y, z, \theta)$ where θ is the top-down rotation; $\Delta p = (\Delta x, \Delta y, \Delta z, \Delta\theta)$) and learning policies from an image observation. The total number of DoF in the action is five because the agent also needs to control the gripper open width λ. We define $\Pi : s \mapsto I \in \mathbb{R}^{h\times w}$ as an observation projection function that projects the point cloud C into an orthographic projection depth image I centered at the (x, y, z) position of the gripper. The gripper is located at the center of I with an orientation θ (Figure 2). Notice that the image I effectively encodes the current gripper pose p inside the image such that a planar rotation or reflection of the state will lead to a planar rotation or reflection on the image. Given an expert transition $T = (s, a, s')$, an observation-action pair can be created by $(I = \Pi(s), a) \in B$



(a) Transition Simulation on multiple trajectories



(b) Transition Simulation on one transition



(c) Breadth and depth simulation

Fig. 3. **Concept of Transition Simulation.** (a) shows how simulated transitions (orange) are distributed around expert trajectories (green). (b) shows how Transition Simulation is accomplished for one specific transition, which is being described in Algorithm 1. (c) shows the breadth and depth simulation of our method for $n = 3$ or $m = 3$.

where B is the buffer of all observation-action pairs. Let $\pi^* : s \mapsto a$ be the expert policy. Our goal is to learn a neural network $\pi : I \mapsto a$ from B such that $||\pi^*(s) - \pi(s)||_2 \to 0$.

IV. APPROACH

We propose a novel imitation learning algorithm, **S**imulation-augmented **E**quivariant **I**mitation **L**earning (SEIL). Our method has two main components: a novel augmentation strategy that we call Transition Simulation (TS) and an equivariant model that utilizes the geometric symmetry in the manipulation task.

A. Transition Simulation

Few-shot imitation learning often suffers from poor generalization due to insufficient demonstration data, which is not enough to cover all variations in the workspace. As a result, agents are usually stuck in situations with unseen and out-of-distribution observations. To solve this problem, we propose Transition Simulation that can create simulated expert transitions T^{sim} from existing pre-collected expert transition T, as shown in Figure 3.

Algorithm 1 Transition Simulation

Input: N steps of expert transitions $\{T_1, T_2, ..., T_N\}$, buffer B, valid checking function transition_valid, parameter n and m.

Output: Observation-action pair Buffer B.

for $i = 1 \dots N$ **do**
  B.append$(\Pi(T_i.s), T_i.a)$
  for $j = 1 \dots n$ **do** # breadth expansion
    $T = T_i$
    for $k = 1 \dots m$ **do** # depth expansion
      $(s, a, s') = T$
      Create T^{sim} using Equation 1, 2, 3.
      if transition_valid(T^{sim}) **then**
        B.append$(\Pi(T^{sim}.s), T^{sim}.a)$
      else # data balancing
        B.append$(\Pi(T.s), T.a)$
      end if
      $T = ((C, \bar{p} + \bar{a}^{-1}, r), \bar{a}, \bar{s})$
    end for
  end for
end for

Our key idea is to simulate new expert transitions by creating new observations using the projection function Π and the rigid transformation of the robot gripper. Specifically, given a fixed point cloud C, one can generate a depth image at an arbitrary camera position. This means we can simulate the observation I as if the gripper is located at an arbitrary pose in the workspace (a similar method has been used by [35] to serve as a forward model for rendering action views). We choose to simulate new transitions that are nearby to the demonstrated trajectories; these simulated transitions serve as a funnel that guides the agent toward expert states. Given an expert transition $T = (s, a, s')$ where $s = (C, p, r), a = (\lambda, \Delta p), s' = (C', p', r')$, we first define the inverse of a as inverting the delta pose Δp: $a^{-1} = (\lambda, -\Delta p)$. $\bar{a}^{-1}$ is defined as adding a Gaussian noise to a^{-1}:

$$\bar{a}^{-1} = a^{-1} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma), \tag{1}$$

where σ is a hyper-parameter for the standard deviation of the Gaussian noise. Then we create a simulated position $\bar{p} = p' + \bar{a}^{-1}$ by applying the delta action in $\bar{a}^{-1}$ to p'. Since $\bar{a}^{-1}$ is close to a^{-1}, $\bar{p}$ is within the vicinity of p. Thus a simulated state $\bar{s}$ can be created by replacing p with $\bar{p}$ in s:

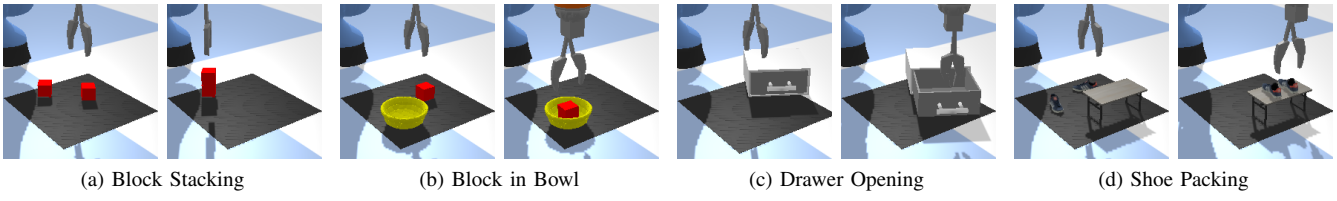
$$\bar{s} = (C, \bar{p}, r). \tag{2}$$

Next, we can take the inverse of $\bar{a}^{-1}$, $\bar{a}$, and form a simulated expert transition:

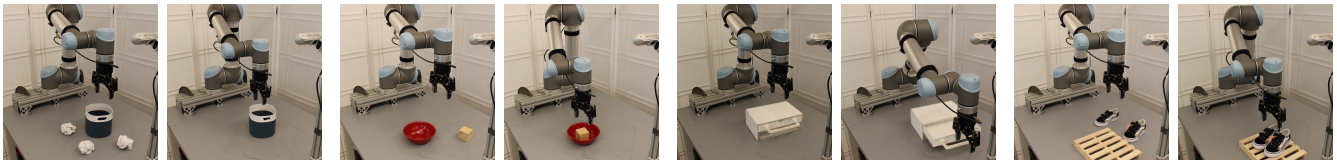
$$T^{sim} = (\bar{s}, \bar{a}, s'). \tag{3}$$

In the end, we can create a new expert observation-action pair by querying $(\Pi(C, \bar{p}, r), \bar{a})$. After this process, we can further simulate a state prior to $\bar{s}$ by applying $\bar{a}^{-1}$ to $\bar{p}$ again: $(C, \bar{p} + \bar{a}^{-1}, r)$.

Intuitively, our method views an expert trajectory as a tree with a single path where the initial state is a leaf and the



(a) Block Stacking (b) Block in Bowl (c) Drawer Opening (d) Shoe Packing



(e) Trash Tidying (f) Block in Bowl (g) Drawer Opening (h) Shoe Packing

Fig. 4. **Environments and goal states.** (a)-(d) Simulation environments in PyBullet [36]; (e)-(h) Real-world environments on a physical robot.

terminal state is the root. Our method creates new expert transitions by expanding the tree with new branches within the vicinity of the existing nodes in the tree. In contrast, standard data augmentation techniques (e.g., applying a rotation or translation to the state and action pair [13]) can only duplicate an expert trajectory path under a fixed transformation applied to all nodes and edges, rather than creating new connected paths. Moreover, our method can be used in conjunction with standard data augmentation techniques to further improve the sample efficiency.

The full algorithm is shown in Algorithm 1. Notice that n and m are hyper-parameters that control the number of breadths and the number of depths for the simulated transitions, illustrated in Figure 3c. Line 8-11 is a data balancing (DB) technique such that when the simulated transition is not valid, a duplication of the original transition will be generated (we empirically find that this data balancing will improve the performance, see Section V-A.3). In our work, the $\mathrm{transition\_valid}$ function is defined as that the simulated position $\bar{p}$ is contact-free (i.e., $\bar{p} \in \mathcal{T}$) and the gripper is not holding any object. However, the method can be extended to placing transitions while holding objects to enhance placing behaviors for pick and place tasks. Line 13 creates a transition prior to $\bar{s}$ for depth expansion.

B. Equivariant Behavioral Cloning (Equi BC)

The second main component in our method is an equivariant model that improves sample efficiency through enforcing $\mathrm{O}(2)$ (i.e., the group of all planar rotation and reflection) symmetry in the network architecture. Inspired by prior work [33], [37] utilizing an equivariant actor function in an actor-critic algorithm, we apply a similar approach to behavior cloning. Specifically, the expert policy function π^* can be written as a $\mathrm{O}(2)$-equivariant function:

$$\pi^*(gs) = g\pi^*(s), \tag{4}$$

where $g \in \mathrm{O}(2)$. That is to say, when the manipulation scene is rotated and/or reflected by g, the action should be rotated and/or reflected accordingly. g acts on the state s through rotating and reflecting the point cloud C and the pose of the gripper p in the state; g acts on the action a through rotating and reflecting the (x, y) vector in Δp and reflecting the θ component in Δp. Formally, we define

$$gs = (gC, gp, r); ga = (\lambda, (R_g(\Delta x, \Delta y), \Delta z, f\Delta\theta)), \tag{5}$$

where R_g is the 2×2 transformation matrix with respect to g, and $f \in \{1, -1\}$ is the reflection component in g. Notice that our observation projection function $\Pi : s \mapsto I$ is an equivariant function, i.e., $\Pi(gs) = g\Pi(s) = gI$ because the image I encodes both the point cloud C and the gripper pose p. We can then define our policy network $\pi : I \mapsto a$ as an equivariant function:

$$\pi(gI) = g\pi(I). \tag{6}$$

We implement such an equivariant policy network using Steerable CNNs [31], [30]. Steerable CNNs share weights across different group elements, enabling the network to automatically generalize to all group transformations and, consequently, improve the sample efficiency. We use a similar network architecture as the prior work [33]. For a more comprehensive introduction on how to implement such an equivariant network, please refer to [33].

V. EXPERIMENTS

A. Simulation experiments

1) Tasks: In this experiment, we evaluate SEIL in four tasks in simulation using PyBullet [36]: Block Stacking, Block in Bowl, Drawer Opening, and Shoe Packing (Figure 4)[1]. For all environments, the maximum moving distances per step are limited as $\Delta x, \Delta y, \Delta z \in [-2cm, 2cm]$; $\Delta\theta \in [-\frac{\pi}{8}, \frac{\pi}{8}]$; $\lambda \in [0, 1]$ where 0 indicates fully close and 1 indicates fully open. We use a state-based planner to collect N episodes of expert demonstrations. For Transition Simulation, we set the breadth expansion parameter n to 4 and the depth expansion parameter m to 1. The Gaussian noise standard deviation σ is set to 0.4. In addition to TS, we also apply a rotational data augmentation where each expert observation-action pair (including the ones created using TS)

[1]The first three environments are from the BulletArm benchmark [38].

TABLE I

BASELINE COMPARISONS IN SIMULATION. TASK SUCCESS RATES AVERAGED OVER FOUR RUNS.

	Block Stacking				Block in Bowl				Drawer Opening				Shoe Packing			
Method	1	5	10	100	1	5	10	100	1	5	10	100	1	5	10	100
CNN BC	18.5	73.5	79.0	90.5	19.0	93.5	99.0	**100**	31.0	66.5	76.0	88.5	2.0	4.7	12.0	20.0
Implicit BC	11.0	9.5	51.0	80.5	13.0	**99.5**	**100**	**100**	31.0	63.5	71.5	81.5	0.5	5.5	12.0	13.0
CNN BC + TS	41.0	75.0	87.0	92.0	52.2	91.0	96.5	98.5	53.5	76.0	75.0	84.5	7.5	13.0	22.0	26.0
Equi BC (Ours)	33.5	87.5	93.0	**100**	46.5	**99.5**	99.5	**100**	62.5	88.5	91.0	**100**	1.5	22.5	39.5	**75.0**
SEIL (Ours)	**71.5**	**99.5**	**98.5**	**100**	**75.0**	98.0	**100**	**100**	**78.5**	87.5	**93.5**	96.5	**16.5**	**57.3**	**68.0**	72.0

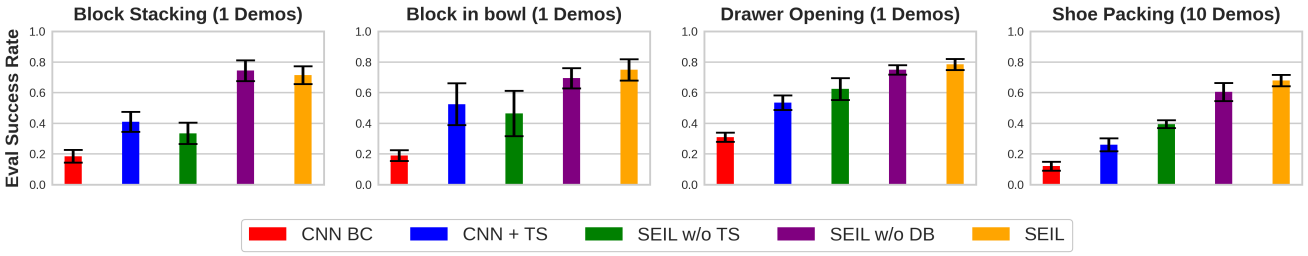


Fig. 5. **Ablation study.** The figure shows how every component of our method improves performance. Results averaged over four runs.

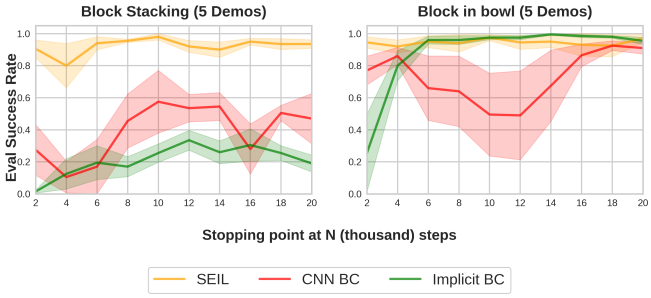


Fig. 6. **Convergence.** The plot shows the performance of the learned policy at different numbers of training steps. Results averaged over four runs. Shading represents standard error.

is augmented 64 times using random $\mathrm{SO}(2)$ rotations. The models are trained with MSE loss for 20,000 training steps. For each run, we evaluate the model for 50 episodes every 2000 training steps (eight evaluations in total). We take the highest success rate of all eight evaluations as the result of one run. Final results are averaged over four runs.

2) Baseline Comparison: We compare SEIL with the following baselines: 1) CNN BC: standard Behavior Cloning method implemented using conventional CNNs; 2) CNN BC + TS: CNN BC equipped with our Transition Simulation; 3) implicit BC [3]: an energy-based model (EBM) that takes observation and actions as input, outputting corresponding scalar energies (implicit BC uses infoNCE loss instead of MSE loss); 4) Equi BC: Behavior Cloning implemented using our equivariant policy network. We run each method in each environment with $N = 1, 5, 10, 100$ demonstration episodes. Note that all baselines without TS are given more $\mathrm{SO}(2)$ augmentations to ensure that the size of the training data is the same for all methods.

Table I shows the results. First, notice that our SEIL algorithm has the best overall performance. It allows the agent to directly generalize to unseen scenarios with a few expert demonstrations. In particular, given only one episode of expert data, SEIL shows a large performance gap compared with the baselines. When the agent is provided with more data, our method still outperforms the baselines with a high success rate. Second, comparing CNN BC + TS and Equi BC with CNN BC, both TS and the equivariant policy function generally improve the performance of CNN BC, especially with fewer demonstrations. This indicates that both TS and the equivariant policy can be used as a plug-in in a standard BC method to boost performance. Another interesting finding is that equivariant models converge faster and is more stable than conventional CNN models due to data efficiency. Figure 6 shows the evaluation curve of SEIL compared with CNN BC and implicit BC in two environments. CNN models converge slower and the performance is not stable at different training steps. Consequently, frequent evaluations in different training steps are necessary for CNN models to find the best performance. In contrast, Equivariant models are able to converge within 2k training steps and keep their robustness during eight evaluations, which is very useful for on-robot learning where frequent evaluation is time-consuming.

3) Ablation Study: In this experiment, we study the importance of each component of our method in an ablation study. We consider the following baselines: 1) CNN BC: a baseline without any of our contribution; 2) SEIL w/o Equi: our method using a standard CNN network instead of an equivariant network (i.e., only having the TS in Section IV-A. This is equivalent to equipping CNN BC with TS); 3) SEIL w/o DB: our method without data balancing (line 8-11 in Algorithm 1); 4) SEIL w/o TS: our method without Transition Simulation completely (i.e., only having the equivariant policy in Section IV-B); 5) SEIL: our full algorithm. Figure 5 shows the result. First, notice that removing TS (green) or removing the equivariant policy (blue) both lead

TABLE II

BASELINE COMPARISONS IN REAL-WORLD EXPERIMENTS. TASK SUCCESS RATES AVERAGED OVER 30 EPISODES ACROSS THREE SEEDS.

	Trash Tidying			Block in Bowl			Drawer Opening			Shoe Packing		
Method	1	5	10	1	5	10	1	5	10	1	5	10
CNN BC	0.0	3.3	20.0	0.0	50.0	76.7	23.3	60.0	70.0	10.0	46.7	70.0
SEIL (Ours)	**30.0**	**60.0**	**93.3**	**36.7**	**90.0**	**100.0**	**86.7**	**96.7**	**100.0**	**13.3**	**60.0**	**90.0**

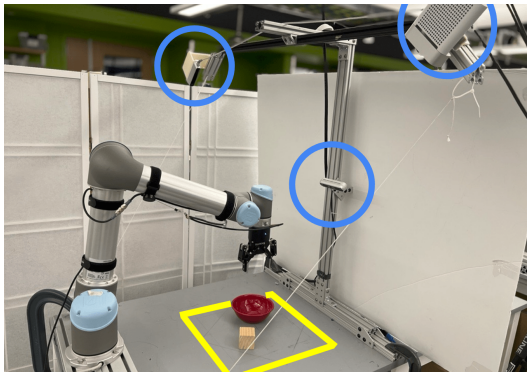


Fig. 7. **Real-world environment and settings.** Three cameras are marked in blue circles. The workspace is highlighted in yellow.

to a significant performance drop compared with our full algorithm (yellow), showing that they are equally important in our method. Meanwhile, both green and blue are significantly better than the baseline standard BC method (red). Second, comparing the purple and the yellow, removing the data balancing slightly impedes the performance of our method.

B. Real-world experiments

1) System setup: Our real-world experimental platform consists of a Universal Robots UR5 robotic arm equipped with a Robotiq 2F-85 parallel-jaw gripper and three depth cameras (one Intel Realsense D455, one Occipital structure sensor, and one Microsoft Azure Kinect DK), as is shown in Figure 7. The point clouds from three cameras are fused into the point cloud C in our state space. Our workstation has an Intel Core i7-9700k CPU (3.60GHz) and an Nvidia RTX 2080Ti GPU. We experiment with four tasks in the real world: Trash Tidying, Block in Bowl, Drawer Opening, and Shoe Packing (Figure 4 e-f). The expert demonstrations are provided by a person using a PS4 controller. In each task, we train the model for 20k training steps and evaluate the success rate over 10 episodes. In Table II, we report the performance averaged over three random seeds (30 episodes in total). Other experimental parameters match the simulation experiment in Section V-A.

2) Results: As shown in Table II our SEIL constantly outperforms the baseline CNN BC, especially when there are fewer expert demonstrations. In Trash Tidying, SEIL achieves 93.3% success rate with ten expert demonstrations, whereas CNN BC can only reach 20%. The common failure mode for CNN BC is that it fails to learn a policy to pick up any trash. Our SEIL algorithm, on the other hand, generalizes well from the limited demonstrations. In Block in Bowl with ten demos, SEIL succeeds in all 30 evaluations, whereas

(a) CNN pick (b) CNN place (c) SEIL pick (d) SEIL place

Fig. 8. **Comparison of SEIL and CNN BC in Shoe Packing.** SEIL is able to pack the shoes with reasonable orientations, but CNN cannot.

CNN often fails to accurately pick up the block (e.g., in 2 of the failures, it picks up the bowl instead of the block) and only achieves 76.7% success rate. With only five demons, SEIL can still achieve a 90% success rate whereas CNN BC can only reach 50%. In Drawer Opening with ten demos, SEIL reaches 100% success rate. CNN BC often fails to stick the finger into the gap between the drawer and the handle (in all failures), and only reaches 70% success rate. In Shoe Packing with ten demons, SEIL not only outperforms the baseline by 20%, but also packs the shoes with the proper orientations (Figure 8).

VI. CONCLUSIONS AND LIMITATIONS

This paper proposes Transition Simulation, a novel trajectory generation algorithm that can augment the existing expert demonstration data. We further integrate Transition Simulation with equivariant policy learning to introduce SEIL, a sample-efficient imitation learning algorithm that can solve robotic manipulation tasks with ten or fewer expert demonstrations. The major limitation of our work is that Transition Simulation can only simulate transitions where the gripper does not come into contact with the environment. This can be improved by learning an environment model to predict the outcome of the transition with contact. Another limitation is that our observations are based on depth images without color information, preventing the agent from solving more complex tasks like color-based sorting. In future work, we plan to include RGB channels in the point cloud and in the projection image. Another direction for future work is to extend our Transition Simulation to simulate on-policy transitions for reinforcement learning.

VII. ACKNOWLEDGMENTS

The authors would like to thank Chenghao Wang for insightful suggestions. This work is supported in part by NSF 1724257, NSF 1724191, NSF 1763878, NSF 1750649, NSF 2107256, and NASA 80NSSC19K1474. R. Walters is supported by the Roux Institute and the Harold Alfond Foundation and NSF 2134178.

REFERENCES

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

[2] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 362–369, 2019.

[3] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

[4] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[5] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[6] Z. Mandi, F. Liu, K. Lee, and P. Abbeel, "Towards more generalizable one-shot visual imitation learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2434–2444.

[7] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv preprint arXiv:2004.13649*, 2020.

[8] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.

[9] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[10] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1612–1619, 2022, publisher: IEEE.

[11] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 739–13 748.

[12] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *arXiv preprint arXiv:2012.06733*, 2020.

[13] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[14] H. Huang, D. Wang, R. Walter, and R. Platt, "Equivariant transporter network," *arXiv preprint arXiv:2202.09400*, 2022.

[15] D. Wang, C. Kohler, and R. Platt, "Policy learning in se (3) action spaces," *arXiv preprint arXiv:2010.02798*, 2020.

[16] U. Viereck, K. Saenko, and R. Platt, "Learning visual servo policies via planner cloning," in *International Symposium on Experimental Robotics*. Springer, 2020, pp. 285–295.

[17] O. Kilinc and G. Montana, "Reinforcement learning for robotic manipulation using simulated locomotion demonstrations," *Machine Learning*, vol. 111, no. 2, pp. 465–486, 2022.

[18] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *arXiv preprint arXiv:1909.12200*, 2019.

[19] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, "A framework for efficient robotic manipulation," *arXiv preprint arXiv:2012.07975*, 2020.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[21] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems*, vol. 33, pp. 19 884–19 895, 2020.

[22] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[23] J. Wang, W. Li, H. Jiang, G. Zhu, S. Li, and C. Zhang, "Offline reinforcement learning with reverse model-based imagination," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 420–29 432, 2021.

[24] P. Mitrano and D. Berenson, "Data augmentation for manipulation," *arXiv preprint arXiv:2205.02886*, 2022.

[25] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[26] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[27] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba, "3d neural scene representations for visuomotor control," in *Conference on Robot Learning*. PMLR, 2022, pp. 112–123.

[28] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1134–1141.

[29] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8943–8950.

[30] M. Weiler and G. Cesa, "General e (2)-equivariant steerable cnns," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[31] T. S. Cohen and M. Welling, "Steerable CNNs," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rJQKYt5ll

[32] D. Wang, R. Walters, X. Zhu, and R. Platt, "Equivariant q learning in spatial action spaces," in *Conference on Robot Learning*. PMLR, 2022, pp. 1713–1723.

[33] D. Wang, R. Walters, and R. Platt, "SO (2) equivariant reinforcement learning," in *International conference on learning representations (ICLR)*, 2022.

[34] X. Zhu, D. Wang, O. Biza, G. Su, R. Walters, and R. Platt, "Sample efficient grasp learning using equivariant models," *arXiv preprint arXiv:2202.09468*, 2022.

[35] 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, publisher: IEEE.

[36] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[37] D. Wang, M. Jia, X. Zhu, R. Walters, and R. Platt, "On-robot learning with equivariant models," *arXiv preprint arXiv:2203.04923*, 2022.

[38] D. Wang, C. Kohler, X. Zhu, M. Jia, and R. Platt, "Bulletarm: An open-source robotic manipulation benchmark and learning framework," *arXiv preprint arXiv:2205.14292*, 2022.