Sim-to-real via Image and Action Alignment

Andrew Kondrich*
Stanford University
Stanford, USA
andrewk1@stanford.edu

Jonathan Booher*
Stanford University
Stanford, USA
jonathanb@cs.stanford.edu

Matthew Ricks*
Stanford University
Stanford, USA
ricks@stanford.edu

***Abstract*—State-of-the-art reinforcement learning (RL) methods for robotic skill learning require large amounts of data collected via exploration, which can be prohibitively time-consuming and dangerous on real hardware. Thus, a common approach to solving robot learning tasks is to first train an agent in a computer simulation using a physics engine such as MuJoCo, and then transfer the learned policy to a physical robot. These methods take considerable time and compute, and do not effectively leverage existing data collected prior to any transfer attempt. In this work, we propose a novel demonstration-based algorithm that decouples visual and dynamics transfer from policy learning by learning correspondences between semantically matching states and actions in real and simulation. These correspondences can then be used to translate real states to sim and simulated actions to real such that a simulation policy can be rolled out in the real world. We compare performance against a number of baselines. Notably, we are able to show improvement over state-of-the-art visual transfer methods, particularly Randomized-To-Canonical Networks [5], with an increased reach success rate from 20% to 50%. We also show that this method can learn between different action spaces used in the simulator and real world with only a 10% reduction in performance in comparison to choosing the same action space.**

I. Introduction

Robot learning is an area of research that explores how deep learning can be used to make robots more efficient, intelligent, and robust to variations in input space and objective. Relying on deep reinforcement learning and an iterative trial-and-error approach, current methods for solving control tasks, such as teaching a robotic arm to lift a cube, require large amounts of data. When training on real robots, such deep approaches can be prohibitively expensive and time-consuming. In order to avoid this training overhead, a common approach to solving such robot learning tasks is to first train an agent in simulation using a physics engine such as MuJoCo, and then transfer this learned policy to a physical robot. Since these simulators do not perfectly recreate how objects respond to contact or how they look in the real world, achieving this policy transfer robustly is nontrivial and an active area of research.

Current state-of-the-art methods rely on heavy domain randomization or require rolling out real trajectories iteratively to adapt a simulation-based policy to the real world. These can be time-consuming approaches that require significant changes to the environment and simulation policy before any reasonable transfer can occur. Furthermore, these methods require the same action space to be used in the simulator and real world. By leveraging existing data, we can more effectively perform a zero-shot policy transfer without needing a policy trained with domain randomization or in-the-loop system identification for dynamics adaptation.

We first postulate that different controllers may be better suited for simulation or real control respectively, motivating the need for arbitrary action translation. For example, if a real-world policy may need to be interoperable with human control, then an end-effector action space would be an ideal choice of control. However, we have seen that end-effector space performs poorly for training reinforcement learning, and is outperformed by joint velocities for our choices of task. Indeed, existing work has demonstrated that a proper choice of action space can simplify the exploration needs of tasks and improve model robustness in reinforcement learning [9]. Thus, it is reasonable to assume that there can exist scenarios where having the capacity to transfer arbitrarily between action spaces is useful.

Our proposed method builds off of prior work and addresses the need for transfer of dynamics as well as the visual state space in order to achieve maximum policy transferability. The approach is theoretically policy independent as it can operate on trajectories. This is important as we can therefore decouple visual adaptation, dynamics adaptation, and policy learning, allowing for potentially more efficient learning of transferrable policies. By adapting images from real to simulation, we can utilize the policy trained using simulation images to get candidate actions which we can then adapt to actions in the real domain addressing potential dynamics mismatch.

In this work, we examine the task of lifting a cube with a Sawyer robotic arm with randomized cube location and orientation from RGB data (see Fig. 1 for examples of the domain in simulation and the real world) and proprioception. Overall, we note that all attempted approaches had trouble in successfully lifting the cube in the real world. However, our novel approach is capable of outperforming our baselines, achieving our best results of 50% successful reaches using joint velocities in both sim and real, and 40% successful reach with a learned transfer to operational space control. Notably, the addition of action adaptation improves on exclusively using visual adaptation via Randomized-To-Canonical Networks [5] or CycleGAN [12].

II. Related Work

Sim-To-Real transfer is a well-known problem with many attempts at solutions. We explore a number of related works to our approach, primarily methods for bridging the visual

sim-to-real gap, existing attempts to incorporating real data into improving transfer, and current work on Cross Domain Imitation Learning.

RCAN [5] is a model that trains a conditional GAN to transform arbitrary styles of task scenes into a "canonical" visual style by leveraging domain randomization in simulation and expecting generalization to the real world. This model outperforms training a policy w/ domain randomization. Notably, it does not attempt any dynamics transfers and is a potential reason as to why it requires training on 5,000 additional real rollouts to improve from 70% raw transfer to 91%. We extend on RCAN by using a limited set of real demonstrations to improve dynamics transfer. This should hopefully improve the number of needed rollouts.

In domain randomization for both visual state and dynamics [11], the task is to train the RL policy directly on randomized scenes/dynamics and hope that it will generalize to real world. This is the current standard approach to getting sim-to-real to work, but makes tasks inherently harder in simulation, requiring significantly more samples and tuning per task. Furthermore domain randomization based approaches require some expertise in which randomizations are needed for particular tasks.

Sim-opt [1] interleaves RL in simulation with rollouts in the real world to efficiently adapt the parameters of the simulation to that of the real world, where a Bayesian Optimization framework is leveraged to efficiently explore the space of parameters. However, the test rollouts in the real world might not be safe or desirable, and requires choosing and modeling distributions of parameters (significant domain knowledge). Despite this, the approach is relatively successful and uses on the order of minutes of real robot interaction time to get to successful transfer. However, it only handles dynamics mismatch and not visual mismatch, demonstrating zero-shot transfer from sim to real based on ground-truth states only.

Self-Supervised Sim-to-Real Adaptation for Visual Robotic Manipulation [6] is similar to our proposed approach in that it uses sets of unaligned demonstration in sim and in real to assist in zero-shot transfer. This approach exploits the temporal nature of robot experience (by using an action-conditioned dynamics model) without assuming any alignment of underlying states in simulated and unlabeled real images. Their 2-step algorithm is split into first learning a generalizable latent space with domain randomization, then collecting rollouts of their domain-randomization policy in order to finetune the visual state encoder. Notably, their approach does not handle any dynamics transfer. In our work, we assume existing demonstrations of an optimal policy in the real world (human demonstrations), but are able to leverage this information for action adaptation as well.

Domain Adversarial Neural Networks [4] are a strong baseline for learning representations that are shared between domains. These representations can be learned for images from simulation and form real and then used as observation spaces for reinforcement learning. Kim et al. [6] demonstrates strong performance of sim-to-real generalization using just this classic domain adaptation technique. Notably, this approach also does not take into account differences in dynamics.

Lastly, the work we largely based our project off of is Cross Domain Imitation Learning [8], which proposes a new theoretical framework for "MDP alignment", a method which demonstrates that, given optimal policies for different MDPs with certain shared characteristics (such as identical reward), the policies can be "reduced" to a single optimal policy that is applicable in both domains. In their proposed GAMA method, two networks f and g are learned to map between the states and actions of two unaligned demonstration sets. These networks are trained to imitate the behavior in the target domain. This requires a dynamics model in the target domain (in our case the real world) in order to ensure that the translated actions are still performing the task. In addition to a discriminator that determines if the (s, a, s') tuple from the translated actions is from the same distribution as the source domain. The original works attempts transfer between inherently disjoint morphologies but all in simulation on simple tasks, such as transferring policies between 2-joint and 3-joint snake morphologies. This MDP alignment task is core to our proposed model. We notably remove the learned state adaptation network f and replace it with a pretrained RCAN model, as well as pretrain a dynamics model p. We hope to extend on this by showing transfer between more nontrivial tasks and dynamics gaps (sim-to-real) with possibly differing controllers. Unique contributions in our approach include:

1) Using visual observations to enable more expressive polices.
2) Learning controller transfer (instead of embodiment transfer) between simulation and real (e.g. Joint velocities to opspace).
3) Extending to Dexterous Manipulation tasks that are more rich in dynamics
4) Attempting actual sim-to-real transfer instead of sim-to-sim
5) Increasing training stability by pretraining elements of the model.

III. Data

We choose a reach and lift task with random cube location and orientation because it is difficult enough that most model-free RL algorithms are not sample efficient enough to solve the task on real hardware but is readily solved in simulation. We also examine two different action spaces and their effectiveness for transfer, the first being joint velocity control and the other being operational space control [7]. We examine the joint velocity action space as that space is shared between the simulated and real robots; however, operational space control is more desirable for use on the real robot due to its increased safety and accuracy, posture control, and can be more readily paired with a force sensor for increased tactile control.

For our behavioral cloning baselines, we are using 50 demonstrations each for operational space control and joint velocity control collected on the real robot. In order to train a model that can accurately and robustly adapt actions between

domains and controllers, we need more interaction data than just the demonstrations that contain mappings between state, action, and next state tuples. In order to accomplish this, we created a script to move the robot randomly within the desired range of motion for about 20 min each using both controller types (velocity and operational space).

For training the visual adaptation models (RCAN and CycleGAN), we used the frames from an existing collection of 200 demonstrations in simulation and 50 demonstrations in real. The collections were resized to both 256×256 and 84×84 resolutions to compare model performance at different resolutions.

(a) Image of the sim domain. (b) Image of the real domain.

Fig. 1: Images from the task and domain being examined.

IV. Methods

A. Model-free RL with MDP Adaptation

Generative Adversarial MDP Alignment (GAMA) [8] is an algorithm that learns state and action correspondences between unpaired and unaligned datasets. Two MDPs are alignable if there exists a surjective mapping between the sets of optimal state-action pairs such that both MDPs can be "reduced" to a single MDP. In the sim-to-real case, we know that the states and the actions are semantically identical, where the primary mismatch is between dynamics and visual representation. We can thus learn correspondences between real and simulated images and actions and then perform a zero-shot sim-to-real transfer by rolling out a sim-trained RL policy in the real world. The full MDP Alignment algorithm is shown in Fig. 2. We assume we are given a set of real world cube lifting demonstrations D_x (collected by hand) and a trained cube lifting policy in simulation from image state π_y. First, we learn a dynamics model P_{θ_p} on real data to predict the next real (image) state s'_x given previous state-action (s_x, a_x) using a video prediction model like Visual MPC [3] or Contrastive Forward Dynamics [6]. We represent a function f for mapping between state correspondences with a Random-To-Canonical Network (RCAN) [5] that transforms real world images to the visual style of the simulated domain. Lastly, g is a neural network that learns to predict the corresponding real action a_x for a simulated action a_y from state s_y. We pretrain f and P_{θ_p} using the collected real world demonstrations. Next, we train g in the GAMA loop adversarially using a discriminator between the generated state-action tuples from f and g, $(\hat{s}_y, \hat{a}_y, \hat{s}'(y))$ and the real trajectories in simulation (s_y, a_y, s'_y), as well as with an MSE loss between $\hat{a}_x$ and a_x. At test time, we translate new real images s_x using the RCAN model f, receive an action $\hat{a}_y$ using the trained simulated policy, and then translate the new action back to the real domain with g.

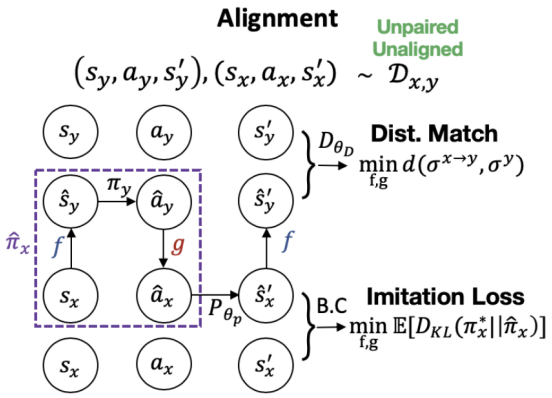


Fig. 2: MDP Alignment as in Kim et al. [8]

B. Visual Transfer Model

We attempted two methods to represent the visual transfer model, each with their own benefits and drawbacks.

1) CycleGAN: CycleGAN [12] is a model that learns correspondences between unpaired images in different domains using a cycle-consistency loss and adversarial training. This model benefits from training directly on the real data distribution, and was able to generate plausible simulated recreations of the real world with minimal stylistic artifacts. However, we found that this model is sensitive to the training data distribution, and can learn semantically wrong mappings for states that never appeared while training. This kind of behavior is the likely cause of it's poor performance, where rollouts using CycleGAN-adapted states were 100% unsuccessful in reaching the cube. An example of the semantically wrong transfer is presented in Fig. 3.

2) Random-To-Canonical Networks: As described in the Related Works, Random-To-Canonical Networks learn to transfer images of a task scene in arbitrary visual styles to a single canonical domain. This method enforces semantic consistencies via segmentation and depth loss terms. Assuming the model successfully can generalize to the real domain, these losses ensure that the cube and arm position stay relatively the same. Furthermore, by training with camera pose randomization, we are more able to ensure that a mismatch between the sim and real camera pose and intrinsics can be handled well by the model. We believe these robustness constraints allowed this method to outperform CycleGAN in our experiments.

C. Dynamics Model

In this work, we consider modeling P as a latent space dynamics model. We use an autoencoder with encoder E and

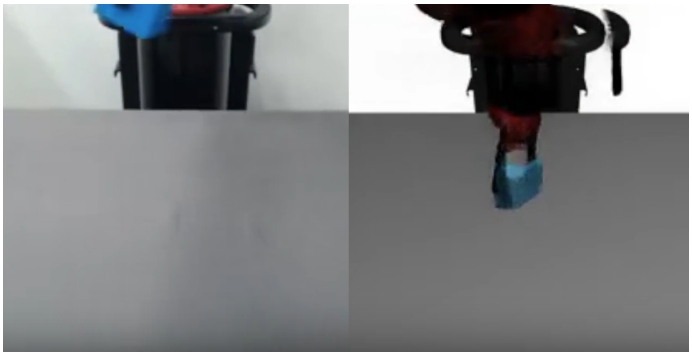


Fig. 3: CycleGAN out of distribution sample. Note that the real cube is almost out of the frame, a state not encountered in the training simulated data. This creates a wrong "mode-collapse" state as on the right.

decoder D to learn a latent space for the images that is rich in information about the state of the world. Specifically, we choose a latent space of size 50 which we found to enable high quality reconstructions. The transition model is then learned using an MLP (M) that takes as input the concatenation of the embedding of the current state and the action, and produces another latent vector which can be decoded using the autoencoder. We train the model using reconstruction error on the current image and the next image in the dataset while training the transition model using the following loss function

$$L_{latent} = ||E(s') - M(E(s), a)||_2^2$$
$$L_{reconstruction} = ||D(E(s')) - D(M(E(s), a))||_2^2$$
$$L_{dynamics} = \lambda L_{latent} + (1 - \lambda) L_{reconstruction}$$

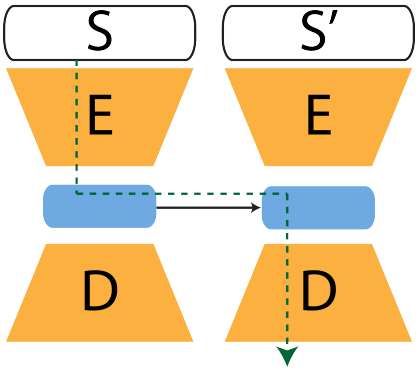


Fig. 4: The dynamics model architecture. Orange is the autoencoder and, black arrow illustrates the transition model and green arrow shows the gradient flow for training the transition model for forward dynamics ($L_{reconstruction}$).

D. Action Translation

We define g as model that maps from current image state to action. We share the same encoder from the learned dynamics model to provide a meaningful embedding of the image that we know captures meaningful information about the dynamics. We freeze this network and concatenate the image embedding with the action proposed by the policy and apply a MLP to produce the action to be applied in the target domain. In this way, the model g can learn transfer between arbitrary action spaces (ie joint velocities and delta positions).

E. Baselines

We attempt some common baselines to assess the performance of the MDP Alignment approach.

1) Real-To-Real Behavioral Cloning: The first approach is to use the real world demonstrations in behavioral cloning on a limited set of demonstrations.

2) Model-free RL with RCAN: We try a naive transfer of a dPPO policy trained in simulation using RCAN to convert real images to the style of the simulator. No dynamics adaptation is used in this transfer.

V. Experiments and Network Architectures

A. Real-to-real Behavioral Cloning

As a preliminary study we tried training a policy using the real world demonstrations we collected, both using the operational space controller, and the joint velocity controller. The joint velocity policy did not come close to the cube, but the opspace policy was able to reach the cube occasionally, though not lift it. Control itself was also much smoother on the opspace policy, which is another motivation for switching from joint velocity in sim to opspace on the real robot.

We tried several things to increase the performance of these behavioral cloning policies. We increased the observations to include proprioception, used frame stacking, and tried perturbing the images by multiplying the image by a random value between .8 and 1.2. None of these tricks improved performance too much except the last one, likely because it helps account for lighting differences from the day data was collected and the day of the rollout.

The final behavioral cloning network architecture was a standard resnet for the images with a spatial softmax on the last layer. In parallel we input the last 5 endpoint states of the robot, xyz position and a quaternion as well as the gripper position. This input went through a three layer network with relu activation. The outputs of these two networks were then concatenated and run through three more fully connected layers to get the final action output.

B. Model-free RL with RCAN

As an ablation of the final model, we run the model without the action adaptation network g using a policy that was trained in simulation and the same action space on the real robot. We still perform the observation translation using the model f. This experiment tests the ability for zero-shot policy transfer using only observation translation.

1) RL policy in simulation: We train a policy $\pi(o_i)$ in simulation that maps a down sampled observation image ($84 \times 84 \times 3$ pixels) to joint velocities which can be directly applied on both simulated and real robots. The simulation is not tuned to match the real-world sawyers other than applying

the correct joint limits as well as minimal damping on the joints leaving open the possibility for significant dynamics mismatch even on the joint velocity action space. Specifically, the policy π is trained using distributed Proximal Policy Optimization (dPPO) with a shaped reward function using the environment `SawyerLift` in Robosuite [2]. The visual appearance of the environment matches that of the "canonical" domain from training RCAN. We choose to use dPPO for its efficiency and robust convergence properties. Training took place with 16 agents running in parallel over the course with batch size 64 over 12 hours to get a robust policy that will also serve as the π in the proposed approach.

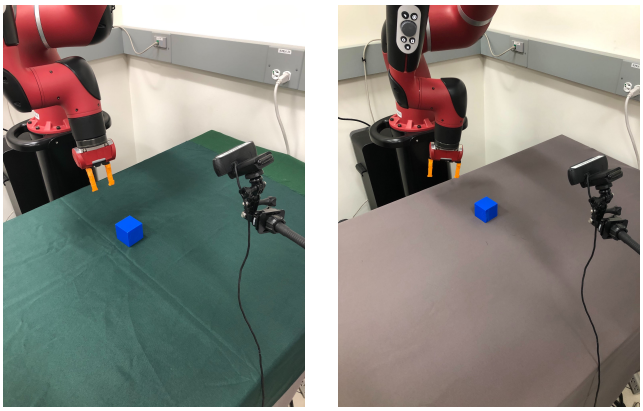
2) RCAN Image-to-Image Transfer: We represent f with a Random-To-Canonical Network [5] that can transform real images ($256 \times 256 \times 3$ pixels) to the style of the simulated domain. The model is trained to reconstruct a "canonical" simulation colorway given randomized data such that it would then be able to generalize to real world samples. The model is trained on 200 cube-lifting trajectories in simulation, where for each trajectory we change the visual randomization every 100 frames. We apply this procedure 3 times such that every frame appears in 3 different randomizations. The randomizations are generated by randomly applying colors, noise and gradients to the different geometries in the scene as well as perturbing the position of light sources and camera. We further randomize the table and skybox with real world images collected from the Zurich Buildings Database [10] to somewhat match the distribution of real world camera data. For each frame of the trajectories, we also record the segmentation and depth map. We train RCAN on a total of 289250 training pairs for 11 epochs. See Fig. 6 for samples from a final trained model, evaluated on a sim-to-sim transfer as well as real-to-sim. We experiment with validating the model on real world data in two domains: a domain that is close to the canonical domain, and another that is "perturbed" (see fig. 5). This evaluation showed that RCAN performance on the perturbed domain was no worse than the performance on the closely aligned domain which illustrates the robustness of RCAN to translating unseen environments to the canonical one.

VI. Results and Discussion

Each policy was tested 10 times and evaluated based on whether the robot was able to reach the cube, reach and lift it, or if it failed to do either.

TABLE I: Final Results

	Failed	Reached	Lifted
Jvel BC	100%	0%	0%
Opspace BC	80%	20%	0%
Sim RL	0%	100%	100%
RL + RCAN	80%	20%	0%
CDIL+CycleGAN* (Jvel)	100%	0%	0%
CDIL+CycleGAN* (OSC)	100%	0%	0%
CDIL+RCAN (Jvel)	50%	50%	0%
CDIL+RCAN (OSC)*	60%	40%	0%



(a) Image of a "perturbed" real environment.

(b) Image of the most "aligned" environment between real and sim

Fig. 5: images of the two domains experimented with in real. One is the best attempt at an aligned image between real and canonical sim domain, and the other is a version that is better translated by RCAN.

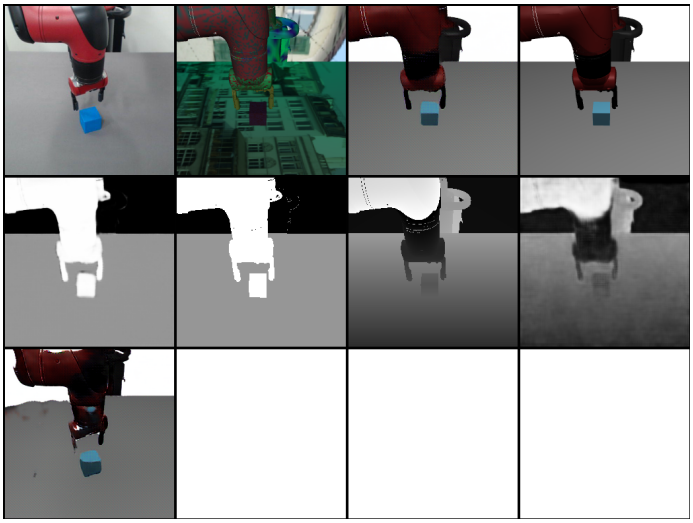


Fig. 6: From left to right, top to bottom: Real Image, Sampled Randomized Simulation Image, Generated Canonical Prediction from Simulation Image, Ground Truth Canonical Image, Predicted Segmentation, True Segmentation, True Depth, Predicted Depth, Generated Canonical Prediction from Real Image

A. Real-to-Real Behavioral Cloning

These results are not too unexpected for behavioral cloning. The distribution of cube placement was very large compared to how many samples we had, so it was challenging for the robot to learn a robust controller to the cube in any location. This issue would be mitigated with scale, which is why leveraging sim data is so important.

B. Zero-Shot Using RCAN

The result for RL + RCAN shows that that idea of visual adaptation from real to simulation enables the use of RL

policies right out of the box to achieve results comparable to behavioral cloning. This adaptation does not take into account differences in physics and as a result, real-robot rollouts tended to drift away from the cube over time.

C. CDIL + CycleGAN

We note here that CycleGAN was not the best model for preserving the location of the cube and the gripper resulting in poor transfer performance. Overall CycleGAN struggled to preserve the semantic structure of the environment, but produced believable translations. This is largely because CycleGAN uses unpaired data that provides weak supervision for semantics while the objective provides strong supervision for style.

D. CDIL + RCAN

These results were the best for the proposed approaches, and shows positive results for transferring to a new controller. While we were unable to get the arm to successfully grasp, we were able to see reaching performance that was robust to perturbations in the cube location during rollouts. We observed the common failure mode that the output from RCAN would have an imagined cube location within the gripper resulting in the policy reaching toward and grasping the imagined cube. This is likely an issue with the data distribution used to train RCAN. We see similar failure modes when training the CDIL approach in simulation.

VII. Future Work

Some of the most common failure modes that were observed were related to visual transfer from the real domain to the simulation domain. By adding a discriminator that learns to discriminate between simulation transitions and transitions with transferred images using RCAN might allow for additional finetuning of the visual adaptation model leading to improved results. Additionally, the current method assumes that the policy π is performing the same behaviors as the demonstrations in the real world, this is often not going to be the case. One possible extension would be to add another dynamics model (in simulation) that will take the translated images, and policy output and produce a plausible next state which can be used for the behavioral cloning loss instead of the next state in the demonstration. This would allow for direct cloning of behaviors in simulation.

References

[1] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, abs/1810.05687, 2018. URL http://arxiv.org/abs/1810.05687.

[2] Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.

[3] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. *CoRR*, abs/1610.00696, 2016. URL http://arxiv.org/abs/1610.00696.

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2015.

[5] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *CoRR*, abs/1812.07252, 2018. URL http://arxiv.org/abs/1812.07252.

[6] Rae Jeong, Yusuf Aytar, David Khosid, Yuxiang Zhou, Jackie Kay, Thomas Lampe, Konstantinos Bousmalis, and Francesco Nori. Self-supervised sim-to-real adaptation for visual robotic manipulation, 2019.

[7] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[8] Kun Ho Kim, Yihong Gu, Jiaming Song, Shengjia Zhao, and Stefano Ermon. Cross domain imitation learning. *ArXiv*, abs/1910.00105, 2019.

[9] Roberto Martín-Martín, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks, 2019.

[10] Hao Shao, Tomas Svoboda, and Luc Van Gool. Zubud — zurich buildings database for image based recognition.

[11] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

[12] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.