How to Close Sim-Real Gap? Transfer with Segmentation!

Mengyuan Yan

School of Electrical Engineering Stanford University Qingyun Sun School of Mathematics Stanford University qysun@stanford.edu

mengyuan@stanford.edu Iuri Frosio Stepl

furi Frosio Stephen Tyree NVIDIA NVIDIA Jan Kautz NVIDIA

ifrosio@nvidia.com

styree@nvidia.com

jkautz@nvidia.com

Abstract

One fundamental difficulty in robotic learning is the sim-real gap problem. In simulation environment, we have infinite amount of data (providing enough computing resource), additionally, we could generate good supervisions if needed for imitation learning, therefore, it is relatively easy to train a model-free control policy such as deep neural network parametrized control policy for a complex control tasks with unknown dynamic model; in real environment, we have extremely limited data as the acquisition of real data is slow and costly, additionally, we rarely have supervisions. Therefore, one natural attempt is to use control policy learned from simulation environment in real environment. However, there is always a gap on such transfer, especially when the deep control policy is end-to-end, namely, taking pixel as input. This gap is called sim-real gap.

In this work, we propose to use segmentation as the interface between perception and control, as a domain-invariant state representation. We identify two sources of sim-real gap, one is dynamics sim-real gap, the other is visual sim-real gap. To close dynamics sim-real gap, we propose to use closed-loop control. For complex task with segmentation mask input, we further propose to learn a closed-loop model-free control policy with deep neural network using imitation learning. To close visual sim-real gap, we propose to learn a perception model in real environment using simulated target plus real background image, without using any real world supervision.

We demonstrate this methodology in eye-in-hand grasping task. We train a closed-loop control policy model that taking the segmentation as input using simulation. We show that this control policy is able to transfer from simulation to real environment. The closed-loop control policy is not only robust with respect to discrepancies between the dynamic model of the simulated and real robot, but also is able to generalize to unseen scenarios where the target is moving and even learns to recover from failures. We train the perception segmentation model using training data generated by composing real background images with simulated images of the target. Combining the control policy learned from simulation with the perception model, we achieve an impressive 88% success rate in grasping a tiny sphere with a real robot.

1 Introduction

Segmentation as the interface between perception and control In robotic learning, it is common practice to resort to robotic simulators for the generation of training data, due to its safety and





Figure 1: We use imitation learning in simulation to train a closed-loop DNN controller, allowing a robot arm to successfully grasp a tiny (1.37cm of diameter) sphere (left panel). The DNN controller's visual input is a binary segmentation mask of the target sphere, extracted from the RGB image captured by the end-effector camera (left inset). A separate DNN vision module processes the real RGB images (right inset) to produce the same segmentation mask as in simulation, abstracting away the appearance differences between domains. Combining the vision module and the controller module in a real environment (right panel), we achieve a grasping success rate of 88% for the real robot.

scalability. However strategies learned in simulation have to generalize well to the real world, which justifies the research effort in the space of domain transfer Tobin et al. [2017]. We take a slightly different approach in addressing the visual domain gap, which takes advantage of the asymmetric information between simulation and the real environment. Because all semantic and geometric information are freely available in simulation, a robot controller can be trained directly from this privileged information, faster and with a smaller neural network. Additional perception modules are trained to translate real sensor inputs into these representations, and they are not burdened with the need to understand the corresponding sensor measurements in simulation.

This work propose to decompose vision and control into separate network modules and use segmentation as their interface. Using segmentation as interface is beneficial in several ways. First, the segmentation mask eases human interpretation of robot behavior, and facilitates development and debugging. Second, perception and control modules can be trained separately or end-to-end. When training separately the vision module can use a variety of data sources collected offline, and from our experiments training separately is considerably faster. End-to-end fine-tuning can improve the performance further. Third, defining segmentation as the module interface effectively exploits the privileged information available in simulation as additional supervision. Finally, modular networks with a well-defined segmentation interface potentially allow the re-use of the same vision or controller module for different robots or environments.

We focuses on grasping with eye-in-hand camera, which is simpler in perception side because the occlusion problem is mostly avoided, but harder in control side, as a close-loop control is required.

Training a closed-loop robot controller efficiently has its unique challenges. With closed-loop control, data collection depends on the policy network being trained, whether using reinforcement learning or imitation learning algorithms. If training with one particular network or hyper-parameters does not go well, the data collected cannot be easily reused to train the next improved version. Therefore, by choosing segmentation as the interface between perception and control, the perception modules can be trained as a prediction problem, where data is fully reusable and can be collected with massive parallel infrastructure; the controller module, taking this domain-agnostic mask as input and controlling robot joint angles, is trained efficiently in simulation using imitation learning and applied directly in real environments. Specifically, our vision module processes end-effector (eye-in-hand) camera images, and extracts the grasp target from the environment in the form of a segmentation mask. Training data for this module are generated by composing background images taken in the real environment with foreground objects from simulation.

Our real robot achieves 88% success in grasping, only using RGB images from the end-effector camera and a closed-loop DNN controller. The real robot is robust to clutter objects or a moving target, and has developed recovery strategies from failed grasp attempts.

Related Work We review different approaches to learning-based robotic grasping system design. Some previous works Mahler et al. [2016, 2017], Jang et al. [2017] use Convolutional Neural Networks (CNN) as grasp pose evaluators, and predict grasp success probability from visual observations and grasp poses. A separate grasp proposal generator is needed to work with the CNN evaluators. The CNNs observe one image and ranks the best grasp pose to be executed. Other works James et al. [2017] use end-to-end neural networks that directly map from visual input to robot joint commands, shifting the grasping system from open-loop to closed-loop. Although there are tools to visualize network filters and activations, it is often hard to decipher what the neural network has learned, or where the network is paying attention to for a specific decision. Devin et al. [2018], Singh et al. [2017] used network structures more similar to our work, where the network has a perception part and a controller part, with feature point coordinates as the connection between the two parts. The feature points are also internal state representations that humans can try to interpret, although their meaning is not always clear. In addition, because the internal state representations are only defined by format, not by content, the neural networks have to be trained end-to-end. In this paper, we also decompose the network into a perception part and a controller part, but we give specific meaning to the interface: it is defined as the segmentation of the target object. Combined with robotic simulation we are able to give direct supervision to the perception network and train the controller part independently from the perception part.

For the choice of the learning environment, robots can learn to grasp directly in the real world Levine et al. [2017], Singh et al. [2017], but training from scratch on real robots is costly, potentially unsafe, and requires very lengthy training time. Training in simulation is easier and cheaper, but transfer learning is needed to generalize from simulation to the real world.

Previous works had looked into the simulation-to-real transfer problem in grasping, for example Fang et al. [2018a,b]. The geometric configuration and appearance of the simulated environment can be extensively randomized, to create diverse training images that induce the neural network to have desired invariance to many visual aspects, and the neural network is shown to generalize also to unseen real environments James et al. [2017]. Inoue et al. [2017] used a variational autoencoder to change the style of the real images into that of the corresponding simulated images; although effective, this method requires coupled simulated and real image pairs to learn the style transfer mapping, thus it does not scale to complex environments. In this paper, we compromise the reality of the real images in exchange for scalable data collection and free annotation of correspondences, and incorporate the idea of randomization to boost generalization. It is worth mentioning that the domain transfer problem does not apply only to vision: since the dynamics of a simulated robot may differ from its real-world counterpart, randomization can also be applied in this domain to facilitate generalization of the trained controller Peng et al. [2018], Tobin et al. [2017].

Another design choice regards the learning algorithm. The training speed depends on the cost of generating training data, the sample efficiency of the learning algorithm Levine and Finn [2017], and the balance of the available computational resources Babaeizadeh et al. [2017]. Deep RL algorithms have been successfully used to play Go and Atari games at superhuman level Mnih et al. [2015], Silver et al. [2016, 2017], Babaeizadeh et al. [2017]. A3C is also employed for robotics in Rusu et al. [2017], although its low data efficiency is posing strong constraints. More sample efficient RL algorithms, like DDPG Lillicrap et al. [2015], explore the solution space more effectively and consequently move some of the computational demand from the simulation to the training procedure, but still require a huge amount of data to reach convergence. The most sample efficient learning procedures are instead based on imitation learning: in this case the trained agent receives supervision from human demonstrations or from an oracle policy, thus the need for policy exploration is minimal and sample efficiency is maximized James et al. [2017], Singh et al. [2017]. Many imitation learning variants have been proposed to improve test-time performance and prevent exploding error Ross and Bagnell [2010], Ross et al. [2011]. We used DAGGER Ross et al. [2011] to train our DNN controller in simulation, with an expert designed as a finite state machine.

Compared to the existing literature, our approach is different in a few ways. In grasping setting, we use the end-effector RGB camera, thus not requiring a fixed camera in a controlled pose; Our close-loop DNN controller allows correcting for pose estimation errors and dynamic changes of the environment in real time; our results also show that, although existing, the "reality gap" Peng et al. [2018] between the dynamic model of the simulated and real robot is only a minor issue in the case of our close-loop controller.

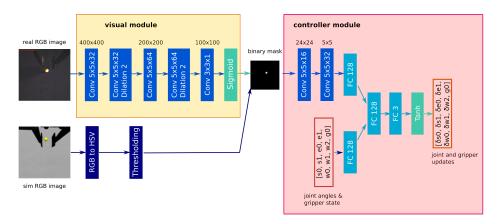


Figure 2: The DNNs for the vision module (left) and closed-loop controller module (right). The vision module takes RGB images from the end-effector camera and labels the sphere pixels. The controller module takes as input the segmentation mask and the current configuration of the robot arm (7 joint angles plus the gripper status), and it outputs an update for the robot configuration.

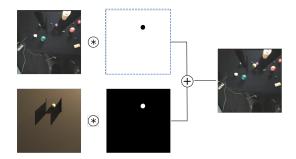


Figure 3: Composing Povray rendered sphere image (lower left) with real background image (upper left). The two images are blended according to the segmentation mask rendered from Povray (lower middle column), and the resulting image is on the right. All images are enhanced for viewing

2 Method

Deep closed-loop controller architecture and imitation learning Our closed-loop DNN controller processes the input information along two different pathways, one for the visual data and the other for the robot state. The visual input is a 100×100 segmentation mask, obtained directly from simulation or from the output of the vision module, segmenting the target object from background. The resulting field of view is approximately 80 degrees. The segmentation mask is processed by 2 convolutional layers with 16 and 32 filters respectively, each of kernel size 5×5 and with stride 4, and a fully connected layer with 128 elements; ReLU activations are applied after each layer (see Fig. 2). The robot state pathway has one fully connected layer (followed by ReLU) that expands the 8 dimensional vector of joint angles and gripper state into a 128 dimensional vector. The outputs of the two pathways are concatenated and fed to 2 additional fully connected layers to output the action command, *i.e.* the changes of joint angles and gripper status $[\delta s_0, \delta s_1, \delta e_0, \delta e_1, \delta w_0, \delta w_1, \delta w_2, g_0]$. A *tanh* activation function limits the absolute value of each command. Contrary to James et al. [2017], we do not use an LSTM module and do not observe a negative impact on the final result, although a memory module could help recovering when the sphere is occluded from the view of the end-effector camera.

We use DAGGER Ross et al. [2011], an iterative algorithm for imitation learning, to learn a deterministic policy that allows a robot to grasp a 1.37cm diameter, yellow sphere. Here we give a brief overview of DAGGER. We gave more details about how to generate the supervision for imitation learning from expert in the appendix in paragraph "Design of the imitation expert".

Given an environment E with state space s and transition model $T(s, a) \to s'$, we want to find a policy $a = \pi(s)$ that reacts to every observed state s in the same manner as an expert policy π_E .

During the first iteration of DAGGER, we gather a dataset of state-action pairs by executing the expert policy π_E and use supervised learning to train a policy π_1 to reproduce the expert actions. At iteration n, the learned policy π_{n-1} is used to interact with the environment and gather observations, while the expert is queried for the optimal actions on the states observed and new state-action pairs are added to the dataset. The policy π_n is initialized from π_{n-1} and trained to predict expert actions on the entire dataset. Ross et al. [2011] also discussed blending the learned policy π_{n-1} with the expert policy when interacting with the environment, i.e. at each state, the action is calculated as $\beta_n \pi_E + (1 - \beta_n) \pi_{n-1}$. We choose $\beta_n = \delta(n = 1)$, i.e. not blending expert policy with learned policies, since we do not see advantage of using $\beta_n = p^{n-1}$, 0 . At each iteration <math>n > 1 the newly gathered state distribution is induced by the evaluated policy π_{n-1} , and over time the training data distribution will converge to the induced state distribution of the final trained policy.

To train this DNN with DAGGER, we collect 1000 frames at each iteration, roughly corresponding to 10 grasping attempts. At each iteration we run 200 epochs of ADAM on the accumulated dataset, with a learning rate of 0.001 and batch size 64. The training loss in DAGGER is the squared L2 norm of the difference between the output of the DNN and the ground truth actions provided by the expert agent, thus defined at iteration n as:

$$\mathcal{L} = \sum_{\mathbf{s}} ||\pi_n(\mathbf{s}) - \pi_E(\mathbf{s})||^2$$
 (1)

Perception training using simulated target plus real background The vision module takes real images from the robot end-effector camera, and predicts segmentation masks of the target object. It is a visual domain translator from real environments to the simulated environment. However, it does not need to imagine the nuance appearance style of the particular simulator, such as the lighting condition, object color and texture. It only needs to keep the essential geometric information in the form of segmentation. The segmentation mask predicted by the vision module has to match the one obtained from the simulator, if the underlying robot pose and sphere position matches.

As shown in Fig. 2, the vision module has 5 convolution layers interleaved with 2 pooling layers. It takes 400×400 RGB images and produces 100×100 segmentation masks. We use kernel size 5×5 and dilated convolution on the second and fourth convolution layers to have a large enough receptive field for each pixel prediction.

To ensure that the geometric information captured by the simulated and real cameras are coherent with each other, we calibrate the internal parameters (focal length, principal points) of the end-effector camera on the real robot and apply the same parameters in the Gazebo simulation. We do not apply any distortion correction as we assume that a good policy executed with a closed-loop control tends to see the sphere in the center of the image, where distortions are minimal.

To collect training data for the vision module, we need to have images as similar to real images as possible, while having cheap annotation of segmentation masks. We can obtain segmentation masks easily from simulated images. We blend the rendered image patches of the sphere with random background images taken by the real robot camera. Fig. 3 demonstrates the composition process, see appendix in paragraph "Image data generation" for the details of our image domain randomization method to generate simulated target plus real background.

Dynamics domain transfer The "reality gap" between the dynamic response of the simulated and real robots may also require domain adaptation. Several issues may contribute to the generation of this gap, including an inaccurate robot model or model parameter setting in simulation, hysteresis, joint frictions, delays in the transmission of the control signals, and noisy measurements of the state in the real robot Tobin et al. [2017]. Fig. 4 shows how the responses of the real and simulated robots can differ with an open-loop controller: starting from the same configuration, the execution of the same sequence of commands at the same frequency leads to two different robot configurations. While small differences accumulate over time in the case of an open-loop controller, our choice of a closed-loop controller corrects execution errors online, leading to a stable and accurate system as shown in Section 3, without requiring any dynamic domain adaptation as in Peng et al. [2018].

3 Results and Discussion



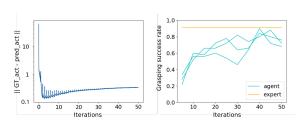


Figure 4: Starting from the same initial configuration (left-most panels), the simulated (top row) and real (bottom row) robots execute the same sequence of 150 commands at 10Hz. Because of differences in the dynamic model of the simulated and real robot, the final configurations of the two robots are different.

Figure 5: Left: DAGGER loss function \mathcal{L} during training. Overfitting occurs in early iterations when the dataset is small; then the loss stabilizes around its optimal value. Right: grasping success rate of the DNN controller in simulation, evaluated three times every five iterations of DAGGER.

Grasping in simulation We evaluate our DNN controller module in simulation at the end of each iteration of DAGGER. Evaluation is performed by measuring the number of successful grasps for a sphere located at 50 positions regularly spaced on a rectangular grid. For each position, the trial ends with a successful grasp or after 150 steps. To account for uncertainties in the simulator, we run the evaluation three times. Fig. 5 shows the grasping success rate as training progresses: 50K training frames are sufficient to achieve a 92% success rate, matching the performance of the expert. Compared to Rusu et al. [2017], Popov et al. [2017], James et al. [2017] that take 0.3M, 50M, and 1M frames respectively to solve similar tasks, we see the superior data efficiency of DAGGER, relative to other reinforcement or imitation learning algorithms.

Visual inspection of failed attempts reveals that on rare occasions the grippers push against the table and cannot be closed; such cases could be solved if force sensors are available on the robot end-effector. However, in most failure cases, the robot's end-effector reaches the sphere but touches it causing the sphere to roll away quickly out of the camera field of view, too far to be reached even for the expert. Such excessive speed of the sphere only appears in the simulator, probably due to the contrast in mass between the robot arm and the sphere, and a lack of good friction model. In the real environment, the sphere only moves slightly when the robot gripper touches it, and the robot has learned to recover by slightly raising its arm.

Transferring to a real robot There are two sim-real gaps, one is dynamics sim-real gap, another is visual sim-real gap. The dynamics sim-real gap is naturally small, thanks to the closed-loop approach we take for the controller: since the controller corrects previous position errors, the "reality gap" between the dynamics of the simulated and real robots does not represent a critical issue, at least for a robot moving at limited speed.

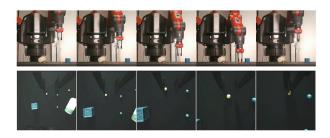


Figure 6: Snapshots of the learned agent grasping in a real environment. The only visual input of the DNN closed-loop controller is the endeffector camera image shown in the bottom row. We have modified the brightness and contrast of the images for ease of viewing.

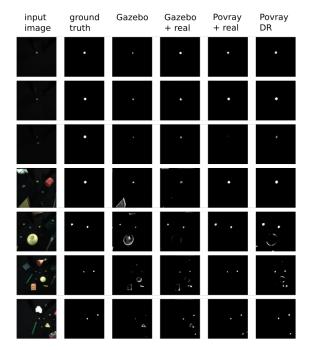


Figure 7: From left to right: input RGB images, ground truth segmentation masks, predicted segmentations from network trained with Gazebo images (at epoch 10), predicted segmentations from network trained with Gazebo + real images (at epoch 10), predicted segmentations from network trained with Povray + real images (our proposed method), and predicted segmentations from network trained with domain randomization implemented in Povray.

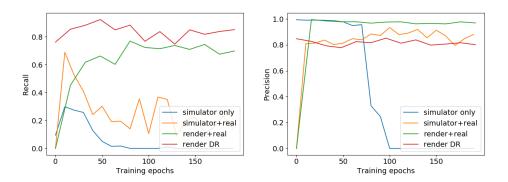


Figure 8: Precision (right) and recall (left) of the vision module on the test set, when training on different datasets. Simulator only (blue line) refers to images collected from Gazebo. Simulator + real (orange line) refers to images composed of Gazebo images and real backgrounds. Render + real (green line) refers to images composed of Povray rendered spheres and real backgrounds. Render DR (red line) refers to Povray rendered images with domain randomization implemented.

Another benefit of close-loop controller on the real robot is that we observe the emergence of recovery strategies from failed attempts, when the controller raises the end-effector slightly above the table to relocate the target sphere. Such recovery behavior can only be scripted in the case of an open-loop formulation, as shown in Levine et al. [2017], but it is learned automatically as an effect of the combined choices of a closed-loop controller, learning through DAGGER, and the design of the expert as a finite state machine. See paragraph "Close-loop controller learns to recover from failures" in appendix.

An agent that is more robust to changes in the environment, *e.g.* lighting conditions, is obtained with the introduction of the DNN vision module in Fig. 2. We compare the effectiveness of our proposed training dataset with other methods. As the first baseline we collected 5000 images from Gazebo while the expert is in action, and used the images to train the same network. Second, we compose the above 5000 images with the 1600 real background images to train the same network. Third,

we re-implemented domain randomization in Povray, generating geometric primitives with random shape, size, position, color, and solid or procedural texture. We also randomized lighting condition and camera viewpoints. We rendered 20000 images to train the same network. For our own method, we rendered 5000 images of the sphere in Povray, with randomized lighting condition and camera viewpoints, and composed them with the same 1600 real background images.

We evaluate the DNN vision module on a set of 2140 images from the real robot, collected by running the DNN controller in simulation and copying the sphere positions and robot trajectories to the real environment. Ground truth segmentation is annotated manually with the help of Maximally Stable Extremal Regions (MSER) detector Matas et al. [2004].

Precision and recall on the test set is evaluated every 10 epochs during training and plotted in Fig. 8. Because the lighting model in Gazebo renderer is rough, and lighting cannot be easily randomized, networks trained with Gazebo images, whether composed with real backgrounds, cannot generalize well to real images. Training the network using Povray rendered images achieves good results. Composing Povray rendered spheres and real backgrounds performs comparable to pure Povray images with domain randomization, with slightly lower recall but higher precision.

Visual comparisons (Fig. 7) confirms the clear advantage of using Povray rendered images compared to Gazebo images, due to more accurate lighting and the ability to depict shadows. The network trained by composing Povray images with real backgrounds occasionally fails to recognize the sphere from the image, while the network trained with domain randomization produces larger segmentation of the spheres, but also produces false positive predictions on the edges of other yellow objects.

When the DNN controller uses the output of the DNN vision module, we evaluate the success rate of the real robot to grasp the sphere. We choose 5 fixed positions to put the sphere, spanning the space used to training the DNN controller, and repeat grasping 5 times at each position. Because the real robot moves slower than the simulated robot, a maximum of 250 steps is allowed for each grasp.

When no clutter objects are present, the real robot can achieve 84% success rate while using the vision module trained with Povray-real composed images. Because the segmentation is not perfect especially when the gripper is close to the sphere, the robot will attempt to grasp several times before a final successful grasp. Using the vision module trained with domain randomization achieves a higher success rate of 88%.

When clutter objects present near the sphere, a segmentation mask with false positives can drive the robot out of the desired trajectory, or even drive it out of the training distribution of the DNN controller, causing self-colliding movements. With the vision module trained with domain randomization, the robot fails to grasp every time a yellow or orange object is present near the sphere. Since the network trained with Povray-real composed images suppress clutter objects much better, the robot succeeds in 2 out of 5 trials.

What if we training the policy end-to-end taking the image pixel space input? In the appendix of paragraph "Comparisons with end-to-end approach", we show that this method is significantly better than end-to-end approaches.

Snapshots from one successful grasp are shown in Fig. 6, while a video of the robot acting in the real environment can be seen in supplementary.

4 Conclusion

In this work, we propose a few principles to close the sim-real gap.

- First, we propose to use segmentation as the interface between perception and control.
- Second, to close the visual sim-real gap, we propose to learn a perception segmentation model in real environment using simulated target plus real background image, without using any real world supervision.
- Third, to close the dynamics sim-real gap, we prose to use closed-loop controller.
- Fourth, we show that imitation learning is a practical and model-agnostic way to learn a deep closed-loop model-free controller that takes segmentation mask as input.

The resulting system achieves a very significant 88% success rate in grasping a tiny sphere on the real robot without any supervision or fine tuning from real environment. The system is robust to moving targets and background clutter and is often able to recover from failed grasp attempts.

References

- Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. In *International Conference on Learning Representations (ICLR) 2017*, 2017. URL https://arxiv.org/abs/1611.06256.
- Coline Devin, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Deep object-centric representations for generalizable robot learning. In *Robotics and Automation (ICRA)*, 2018 IEEE International Conference on. IEEE, 2018. URL https://arxiv.org/abs/1708.04225.
- Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. Multitask domain adaptation for deep learning of instance grasping from simulation. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 3516–3523. IEEE, 2018a.
- Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *arXiv* preprint arXiv:1806.09266, 2018b.
- Tadanobu Inoue, Subhajit Chaudhury, Giovanni De Magistris, and Sakyasingha Dasgupta. Transfer learning from synthetic to real images using variational autoencoders for robotic applications. arXiv preprint arXiv:1709.06762, 2017. URL https://arxiv.org/abs/1709.06762.
- Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 334–343. PMLR, Nov 2017. URL http://proceedings.mlr.press/v78/james17a.html.
- Eric Jang, Sudheendra Vijayanarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 119–132. PMLR, 13–15 Nov 2017. URL http://proceedings.mlr.press/v78/jang17a.html.
- Sergey Levine and Chelsea Finn. Deep reinforcement learning, decision making, and control. 2017. URL https://sites.google.com/view/icml17deeprl.
- Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning handeye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 2017. URL https://doi.org/10.1177/0278364917710318.
- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv* preprint arXiv:1509.02971, 2015. URL http://arxiv.org/abs/1509.02971.
- Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *Robotics and Automation (ICRA)*, 2016 IEEE International Conference on, pages 1957–1964. IEEE, 2016. URL http://ieeexplore.ieee.org/document/7487342/.
- Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. arXiv preprint arXiv:1703.09312, 2017. URL https://arxiv.org/abs/1703.09312.
- Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. URL https://www.sciencedirect.com/science/article/pii/S0262885604000435.

- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. URL https://www.nature.com/articles/nature14236.
- Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Robotics and Automation (ICRA)*, 2018 IEEE International Conference on. IEEE, 2018. URL https://arxiv.org/abs/1710.06537.
- Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017. URL https://arxiv.org/abs/1704.03073.
- Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668. PMLR, May 2010. URL http://proceedings.mlr.press/v9/ross10a.html.
- Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635. PMLR, Apr 2011. URL http://proceedings.mlr.press/v15/ross11a.html.
- Andrei A. Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 262–270. PMLR, Nov 2017. URL http://proceedings.mlr.press/v78/rusu17a.html.
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. URL https://www.nature.com/articles/nature16961.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. URL https://www.nature.com/articles/nature24270.
- Avi Singh, Larry Yang, and Sergey Levine. Gplac: Generalizing vision-based robotic skills using weakly labeled images. *arXiv preprint arXiv:1708.02313*, 2017. URL https://arxiv.org/abs/1708.02313.
- 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. In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pages 23–30, 2017.