Self-Supervised Correspondence in Visuomotor Policy Learning

Peter Florence, Lucas Manuelli, Russ Tedrake
Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology
{peteflo,manuelli,russt}@csail.mit.edu

Abstract

In this paper we explore using self-supervised correspondence for improving the generalization performance and sample complexity of visuomotor policy learning. Prior work has primarily used approaches such as autoencoding, pose-based losses, and end-to-end policy optimization in order to train the visual portion of visuomotor policies. We instead propose an approach using self-supervised dense visual correspondence training, and show this enables visuomotor policy learning with surprisingly high generalization performance with modest amounts of data: using imitation learning, we demonstrate extensive hardware validation on challenging manipulation tasks with as few as 50 demonstrations. Our learned policies can generalize across classes of objects, react to deformable object configurations, and manipulate textureless symmetrical objects in a variety of backgrounds, all with closed-loop, real-time vision-based policies. Simulated imitation learning experiments suggest that correspondence training offers sample complexity and generalization benefits compared to autoencoding and end-to-end training. https://sites.google.com/view/correspondence-visuomotor

1 Introduction

Correspondence is fundamental in computer vision, and we believe it has fundamental usefulness for robots learning complex tasks requiring visual feedback. In this paper we introduce using self-supervised correspondence for visuomotor policies, and our results suggest this enables policy learning that is surprisingly capable. Our evaluations pair correspondence training with a simple imitation learning objective, and extensive hardware validation shows that learned policies can address challenging scenarios: manipulating deformable objects, generalizing across a class of objects, and visual challenges such as textureless objects, clutter, moderate occlusion, and lighting variation (Fig 3). Additionally our simulation-based comparisons empirically suggest that our method offers significant generalization and sample complexity advantages compared to existing methods for training visuomotor policies, while requiring no additional human supervision. To bound our method's scope: while spatial correspondence alone cannot suffice for all tasks (for example, it cannot discriminate when to be finished cooking eggs), there is a wide set of tasks for which dense spatial correspondence may be useful: essentially any task concerning the spatial relationships of objects.

2 Related Work

There have been three primary methods used in the robot learning literature to train the visual portion of visuomotor policies. Often these methods are used together – for example [1, 2] use pose-based losses together with end-to-end training. ***(i) End-to-End training.*** This approach can be applied to any learning signal that is formed as a consequence of a robot's actions, for example through imitation

NeurIPS 2019 Workshop on Robot Learning: Control and Interaction in the Real World, Vancouver, Canada

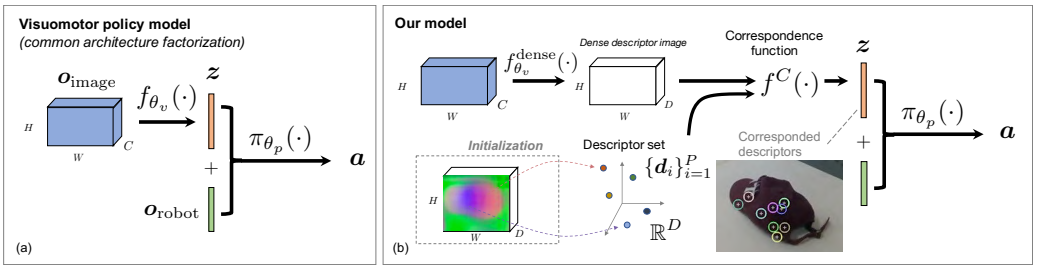


Figure 1: Diagram of common visuomotor policy factorization (a), and our proposed model (b) using visual models trained on correspondence.

learning or reinforcement learning. While often end-to-end training is complemented with other learning signals, other works use purely end-to-end training. ***(ii) Autoencoders.*** Autoencoding can be applied to any data with no supervision and is commonly used to aid visuomotor policy learning [3–8]. Sometimes polices are learned with a frozen encoder [3–5], other times in conjunction with end-to-end training [8]. ***(iii) Pose-based losses.*** In [1], for example, a separate dataset is collected of the robot holding objects, and assuming that the objects are rigid and graspable, then using the robot's encoders and forward kinematics the visual model can be trained to predict the object pose. In [2], pose-based auxiliary losses are used regardless of whether or not objects are held. Simulation-based works [9] have also used auxiliary losses for object and gripper positions.

In our comparison experimentation, we include end-to-end training and autoencoding, but not pose-based losses, since they are not applicable to deformable or un-graspable objects. While the above are three of the most popular, other visual training methods include: training observation dynamics models [10, 11], using time-contrastive learning [12], or using no visual training and instead using only generic pre-trained visual features [13]. Relevant concurrent works include [14] which proposes autoencoder-style visual training but with a reference image and novel architecture, and [15] which proposes a graph-based reward function using a fixed set of correspondences.

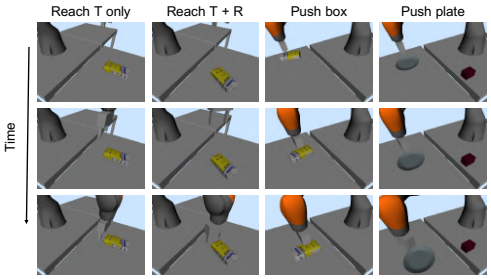
3 Formulation

Preliminary: Visuomotor Policies. We would like to have a policy $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{o}_{0:t})$, where $\boldsymbol{o}_{0:t} = (\boldsymbol{o}_0, \boldsymbol{o}_1, \ldots \boldsymbol{o}_t)$ is the full sequence of the robot's observations during some episode up until time t, with each $\boldsymbol{o}_i \in \mathcal{O}$, the robot's observation space. This sequence of observations is mapped by $\pi_\theta(\cdot)$, the robot's policy parameterized by θ, to the robot's actions $\boldsymbol{a} \in \mathcal{A}$. In particular, we are interested in visuomotor policies in which the observation space contains high-dimensional images $\mathcal{O}_{\text{image}} \subset \mathcal{O}$, for example $\mathcal{O}_{\text{image}} = \mathbb{R}^{W \times H \times C}$ for a C-channel, width W, and height H image. The visual data is perhaps complemented with additional lower-dimensional measurements $\mathcal{O}_{\text{robot}}$, such as produced from sensors like the robot's encoders, such that $\mathcal{O}_{\text{robot}} \times \mathcal{O}_{\text{image}} = \mathcal{O}$.

It is common for a visuomotor policy to have an architecture that can factored as shown in Fig. 1(a), $\boldsymbol{z} = f_{\theta_v}(\boldsymbol{o}_{\text{image}}) : \boldsymbol{o}_{\text{image}} \in \mathcal{O}_{\text{image}}, \boldsymbol{z} \in \mathbb{R}^Z$, $\boldsymbol{a} = \pi_{\theta_p}(\boldsymbol{z}, \boldsymbol{o}_{\text{robot}}) : \boldsymbol{o}_{\text{robot}} \in \mathcal{O}_{\text{robot}}, \boldsymbol{z} \in \mathbb{R}^Z, \boldsymbol{a} \in \mathcal{A}$, in which a visual model $f_{\theta_v}(\cdot)$, parameterized by θ_v, processes the high-dimensional $\boldsymbol{o}_{\text{image}}$ into a much smaller Z-dimensional representation $\boldsymbol{z}$. The policy model $\pi_{\theta_p}(\cdot)$ then combines the output of the visual model with other observations $\boldsymbol{o}_{\text{robot}}$. This is a practical modeling choice – images are extremely high dimensional, i.e. in this work we use images in $\mathbb{R}^{640 \times 480 \times 3} = \mathbb{R}^{921,600}$, whereas our $\mathcal{O}_{\text{robot}}$ is at most $\mathbb{R}^{13}$. A wide variety of works have employed a similar architecture to [1], consisting of convolutional networks extracting features from raw images into an approximately $Z = 32$ to 100 bottleneck representation of features, e.g. [2, 3, 6, 8, 16, 17]

Visual Correspondence Models for Visuomotor Policy Learning. The objective of the visual model is to produce a feature vector $\boldsymbol{z}$ which serves as a suitable input for policy learning. In particular, we are interested in deploying policies that can operate directly on RGB images. Our approach builds off the method of [18] which can in a self-supervised manner, learn pixel descriptors of objects that are effective in finding correspondences between RGB images.

We introduce four different methods for how to employ dense correspondence models as the visual basis of visuomotor policy learning. The first three are based on the idea of a set of points on the object(s) that are localized either in image-space or 3D space from an RGB image. We represent



(a)

Method / Task	Reach T only	Reach T + R	Push box	Push plate
Ground truth 3D points	100.0	100.0	100.0	86.5
Ground truth 2D image coord.	94.1	95.6	100.0	92.0
Auto-encoder, frozen	8.1	61.1	–	53.0
Auto-encoder w/ mask, frozen	16.3	10.0	–	67.0
Auto-encoder, then End-to-End	40.7	38.9	–	16.0
End-to-End	43.0	32.2	**100.0**	5.5
End-to-End (34-layer ResNet)	–	3.3	–	–
DD 2D image coord. (ours)	**94.1**	**97.8**	**100.0**	**87.0**

(b)

Figure 2: (a) RGB images used for visuomotor control in each of the simulation tasks. T=translation, R=rotation, see Appendix for task descriptions. (b) Summary of simulation results (success rate, as %). See Appendix for task success criteria and additional details.

these points as a set $\{\boldsymbol{d}_i\}_{i=1}^{P}$ of P descriptors, with each $\boldsymbol{d}_i \in \mathbb{R}^D$ representing some vector in the D-dimensional descriptor space produced by a dense descriptor model $f_{\theta_v}^{\text{dense}}(\cdot)$. This $f_{\theta_v}^{\text{dense}}(\cdot)$ maps a full-resolution RGB image, $\mathbb{R}^{W\times H\times 3}$, to a full-resolution descriptor image, $\mathbb{R}^{W\times H\times D}$. Let us term $f^C(\cdot)$ to be the non-parametric correspondence function that, given one or more descriptors and a dense descriptor image $f_{\theta_v}^{\text{dense}}(\boldsymbol{o}_{\text{image}})$, provides the predicted location of the descriptor(s):

$$\boldsymbol{z} = f^C\left(f_{\theta_v}^{\text{dense}}(\boldsymbol{o}_{\text{image}}), \{\boldsymbol{d}_i\}_{i=1}^{P}\right) \tag{1}$$

Specifically $f^C : \mathbb{R}^{W\times H\times D} \times \mathbb{R}^{P\times D} \to \mathbb{R}^{P\times K}$, and $K = 2$ corresponds to the case where the descriptors are being localized in the pixel space of the image. Continuing the example of $K = 2$, $\boldsymbol{z}$ contains the (u,v) pixel location in the RGB image for each point $\boldsymbol{d}_i$. See Appendix for specific descriptions of the correspondence function. All methods optimize a generic policy-based loss function $\min_{\Theta} \mathcal{L}\left(\pi_{\theta_p}(\boldsymbol{z}, \boldsymbol{o}_{\text{robot}})\right)$ and vary only in the set of learnable parameters Θ and how $\boldsymbol{z}$ is acquired. Our preferred method is to optimize the descriptor set $\{\boldsymbol{d}_i\}_{i=1}^{P}$ along with the policy parameters θ_p while keeping the dense descriptor mapping $f_{\theta_v}^{dense}$ fixed. Intuitively $f_{\theta_v}^{dense}$ has already been trained to perform correspondence, and we are simply allowing the policy optimization to choose *what to correspond*. The Supplementary Material discusses alternatives and tradeoffs.

4 Results

Our experimentation sought to answer the following questions: (1) Is it possible to use self-supervised descriptors as successful input to learned visuomotor policies? (2) How does visual correspondence learning compare to various baselines in terms of enabling effective policy learning, as measured by generalization performance and sample complexity? All of our experiments use the approach from Section 3 paired with a simple imitation learning objective, more details are provided in Section S.3 of the Supplementary Material.

Simulation Comparisons. Figure 2(b) contains the results of the simulation experiments. For more detail on the tasks and baselines, see the Supplementary Material. Interestingly we find that our method's visual representation is capable of enabling policy learning that is remarkably close in performance to what can be achieved if the policy has access to ground truth world state information. In contrast the performances of the end-to-end (E2E) and autoencoder (AE) methods vary much more across the different tasks. Since our method benefits from object mask information during visual training, we also experimented with letting the autoencoder use this information by applying the reconstruction loss on only the masked image. Additionally we tried training the autoencoder end-to-end during behavior cloning. Both of these yield mixed results, depending on the task. Since the vision network in our method is a 34-layer ResNet, we wanted to see if the end-to-end method would benefit from using the same, deeper vision backbone. The deeper network did not improve closed-loop performance (Figure 2 (b)). This suggests the advantage of our method comes from the correspondence training rather than the model capacity. For more analysis of our simluation comparisons, see the Supplementary Material.

Hardware Validation We validate both our visual learning method and its use in imitation learning in the real world. Figure 3 (b) displays visualizations of learned correspondences from demonstration

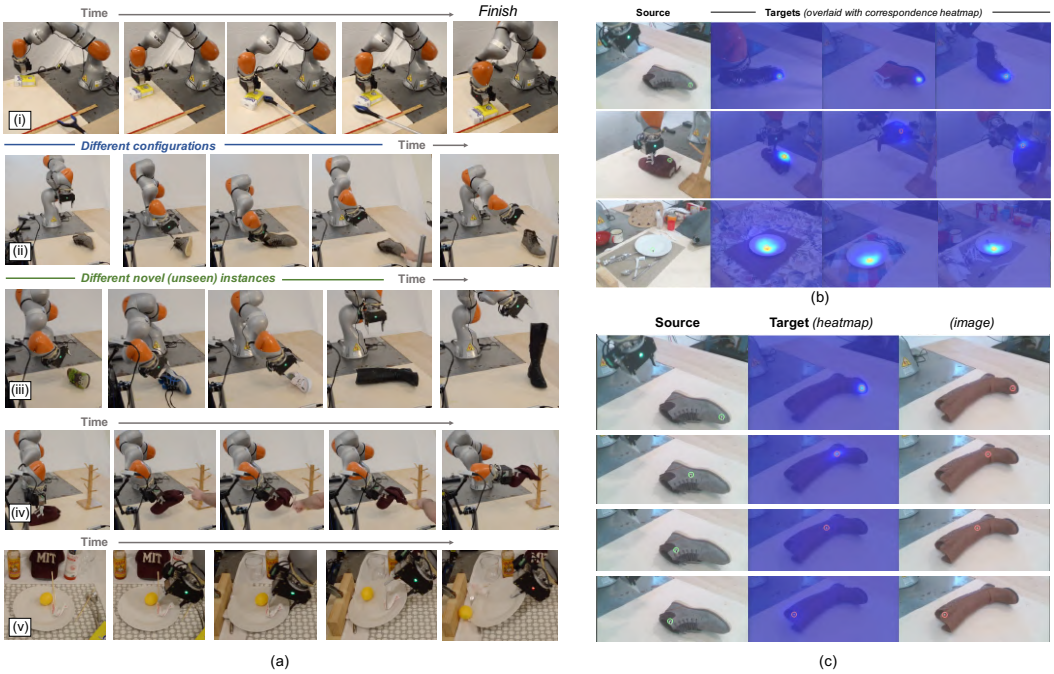


Figure 3: (a) Depictions of hardware tasks, including a variety of non-prehensile, class-general, and deformable manipulation. (b,c) Learned correspondences from demonstration data, depicted as correspondence heatmaps between a source pixel (left, with the green reticle) and target scenes (right, with red reticle as best predicted).

data. The results show that the learned visual models, despite imperfect depth sensor noise, calibration, and only time-synchronized image pairs, are capable of identifying correspondences across a class of objects, for an object in different deformable configurations, and for objects in a diversity of backgrounds. Figure 3(c) displays class-general correspondences for a particularly challenging instance with large shape variation.

Real-World Visuomotor Policies. Figure 3(a) displays examples of autonomous hardware results. Each of the different tasks present significant challenges, best appreciated in our video. Several of the tasks include non-prehensile manipulation, including pushing the box and plate, and flipping the shoes. In the *"Pick-then-hang hat on rack"* task, the robot autonomously reacts to the deformable configuration of the hat after disturbances. The *"Push-then-grab plate"* task as well is highly challenging given the visual clutter, the symmetry and lack of visual texture for the object, and requires using "extrinsic dexterity" [19] via the wood block to enable sliding the gripper into position to grasp the plate. To highlight a few quantitative results, several of the tasks achieve over 95% reliability, including the *"Push sugar box"* task with and without disturbances, and the *"Flip shoe, single instance"* and *"Push-then-grab plate"* tasks without disturbances. See Table 1 in the Supplementary for a quantitative overview.

5 Conclusion

Our experiments have validated that self-supervised correspondence training enables efficient policy learning in the real world, and our simulated imitation learning comparisons empirically suggest that our method outperforms two vision-based baselines in terms of generalization and sample complexity. While different hyperparameters, model architectures, and other changes to the baselines may increase their performance, our method is already near the upper bound of what can be expected in the used experimental setting: it achieves results comparable to baselines using ground truth information. One reason our approach may outperform the baselines is that it additionally uses a fundamentally different source of supervision, provided by visual correspondence training. Since our approach is self-supervised, it does not entail additional human supervision.

References

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[2] T. Zhang, Z. McCarthy, O. Jowl, 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. 1–8.

[3] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 512–519.

[4] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 397–403, 2016.

[5] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[6] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep predictive policy training using reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2351–2358.

[7] H. Van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3928–3934.

[8] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[9] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *Conference on Robot Learning (CORL)*, 2017.

[10] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, 2016, pp. 5074–5082.

[11] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[12] 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.

[13] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual rewards for imitation learning," *Robotics: Science and Systems (RSS)*, 2017.

[14] T. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih, "Unsupervised learning of object keypoints for perception and control," *arXiv preprint arXiv:1906.11883*, 2019.

[15] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, "Graph-structured visual imitation," *arXiv preprint arXiv:1907.05518*, 2019.

[16] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *Conference on Robot Learning (CoRL)*, 2017.

[17] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 79–86.

[18] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," *Conference on Robot Learning (CoRL)*, 2018.

[19] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1578–1585.

[20] P. Florence, "Dense visual learning for robot manipulation," in *PhD Thesis*, 2019.

[21] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[22] S. Ross, G. Gordon, and D. 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*, 2011, pp. 627–635.

[23] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," *arXiv preprint arXiv:1703.09327*, 2017.

[24] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[25] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 420–427, 2017.

[26] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 343–352.

[27] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[28] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3381–3388.

[29] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn, "Unsupervised visuomotor control through distributional planning networks," *arXiv preprint arXiv:1902.05542*, 2019.

[30] A. Singh, L. Yang, and S. Levine, "Gplac: Generalizing vision-based robotic skills using weakly labeled images," in *ICCV*, 2017.

Supplementary Material

S.1 Visual Training Options

Below we describe the variants for visual training, which vary based on the set of optimized paramters.

$$\min_{\Theta} \mathcal{L}\Big(\pi_{\theta_p}\big(\boldsymbol{z}, \boldsymbol{o}_{\text{robot}}\big)\Big) \tag{2}$$

Method	Θ
Fixed Descriptor Set	$\{\theta_p\}$
Descriptor Set Optimization	$\{\theta_p, \{\boldsymbol{d}_i\}_{i=1}^P\}$
End-to-End Dense Optimization	$\{\theta_p, \{\boldsymbol{d}_i\}_{i=1}^P, \theta_v\}$
End-to-End with Correspondence Pretraining	$\{\theta_p, \theta_v\}$ $\boldsymbol{z} = g\big(f_{\theta_v}^{\text{dense}}(\boldsymbol{o}_{\text{image}})\big)$

Fixed Descriptor Set. This method only optimizes the policy parameters θ_p. In particular the set of object points that are to be localized, represented by $\{\boldsymbol{d}_i\}_{i=1}^P$, is fixed. We use a simple initialization scheme of sampling $\{\boldsymbol{d}_i\}$ from a single masked reference descriptor image. While we have found this method to be surprisingly effective, it is unsatisfying that the visual model's representation is not optimized after the random descriptor set initialization process.

Descriptor Set Optimization. Our preferred method, this approach optimizes the descriptor set $\{\boldsymbol{d}_i\}_{i=1}^P$ along with the policy parameters θ_p while keeping the dense descriptor mapping $f_{\theta_v}^{dense}$ fixed. Intuitively $f_{\theta_v}^{dense}$ has already been trained to perform correspondence, and we are simply allowing the policy optimization to choose *what to correspond*. We have observed that Descriptor Set Optimization can improve validation error in some cases over a Fixed Descriptor Set, and introduces minimal additional computational cost and parameters.

End-to-End Dense Optimization. The third option is to train the full model architecture end-to-end by including θ_v in the optimization. While we may have expected this approach to allow the visual model to more precisely focus its modeling ability on task-critical parts of images, we so far have not observed a performance advantage of this approach over Descriptor Set Optimization.

End-to-End with Correspondence Pretraining. The fourth option is to directly apply a differentiable operation to a model which was previously trained on dense correspondence. We can apply any differentiable operation $g(\cdot)$ on top of $f^{\text{dense}}_{\theta_v}$ directly to produce a representation $\boldsymbol{z} = g\big(f^{dense}_{\theta_v}(\boldsymbol{o}_{\text{image}})\big)$. For example, we can apply non-parametric channel-wise spatial expectations to each of the D channels of the dense descriptor images. The optimization variables in this case are $\Theta = \{\theta_p, \theta_v\}$.

Using either a fixed- or optimized- descriptor set will significantly increase policy training speed. In particular, we do not require forward-backward optimizing through a very deep convolutional network in each step of policy training, which, with our architecture (34-layer ResNet), is 1 to 2 orders of magnitude faster.

S.2 Correspondence Function

The specific form of $f^C(\cdot)$ is defined by how the correspondence model was trained. If for example we train a correspondence model via pixelwise contrastive loss, then for some reference descriptor $\boldsymbol{d} \in \mathbb{R}^D$, and some target image $\boldsymbol{o}_{\text{image}}$, then $f^C(\cdot)$ just looks up the nearest pixel descriptor in $\boldsymbol{o}_{\text{image}}$ to d:

$$f^C(f^{\text{dense}}_{\theta_v}(\boldsymbol{o}_{\text{image}}), \boldsymbol{d}) = \underset{u,v}{\text{argmin}} \; ||f^{\text{dense}}_{\theta_v}(\boldsymbol{o}_{\text{image}})[u,v] - \boldsymbol{d}||_2 \tag{3}$$

In our preferred model we use a dense distributional loss in which we compute a spatial expectation using a correspondence kernel. In this case we have:

$$f^C(f^{\text{dense}}_{\theta_v}(\boldsymbol{o}_{\text{image}}), \boldsymbol{d}) = \{\mathbb{E}(u|\boldsymbol{o}_{\text{image}}, \boldsymbol{d}), \; \mathbb{E}(v|\boldsymbol{o}_{\text{image}}, \boldsymbol{d})\} \tag{4}$$

$$= \{\sum_{u,v} p(u,v|\boldsymbol{o}_{\text{image}}, \boldsymbol{d})u, \; \sum_{u,v} p(u,v|\boldsymbol{o}_{\text{image}}, \boldsymbol{d})v\} \tag{5}$$

Where $p(u, v|\boldsymbol{o}_{\text{image}}, \boldsymbol{d})$ is the probability that pixel u, v in $\boldsymbol{o}_{\text{image}}$ corresponds to the descriptor d. See [20], Chapter 4, for details. All results presented in this paper used dense distributional loss for training and used the spatial expectation for $f^C(\cdot)$.

S.3 Imitation Learning Methodology

We now describe our used approach to perform imitation learning for robot manipulation.

Robot Observation and Action Spaces

At the lowest level our controller sends joint velocity commands to the robot. For ease of providing demonstrations via teleoperation, the operator commands relative-to-current desired end-effector poses $T_{\Delta,\text{cmd}}$. A low-level Jacobian based controller then tracks these end-effector pose setpoints. Our learned policies also output $T_{\Delta,\text{cmd}}$. The teleoperator also commands a gripper width setpoint which again is tracked by a low-level controller. Thus the action space is $\boldsymbol{a} = (T_{\Delta,\text{cmd}}, w_{\text{gripper}}) \in \mathcal{A} = SE(3) \times \mathbb{R}^+$.

Our $\boldsymbol{o}_{\text{robot}} \in \mathbb{R}^{13}$ is (i) three 3D points on the hand as in [1], (ii) an axis-angle rotation relative to the task's starting pose, and (iii) the gripper width. As noted previously, $\boldsymbol{o}_{\text{image}} \in \mathbb{R}^{921,600}$.

Imitation Learning Visuomotor Policies

To evaluate visual learning strategies for enabling visuomotor policy learning, we use imitation learning via a simple behavioral cloning [21] strategy, which a few recent works have demonstrated to be viable for learning visuomotor manipulation policies [2, 8]. Optimizing a policy with parameters Θ on the

behavioral cloning objective, given a dataset of N_{train} trajectories of observation-sequence-to-action pairs $\{(\boldsymbol{o}_t, \boldsymbol{a}_t^*)\}_{t=0}^{T_i}$ can be written as:

$$\min_{\Theta} \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \sum_{t=0}^{T_i} \mathcal{L}_{\text{BC}}\left(\boldsymbol{a}_t^*, \pi_\Theta(\boldsymbol{o}_t)\right) \tag{6}$$

For our loss function we use a simple weighted sum of l_1 and l_2 loss, $\mathcal{L}_{\text{BC}}(\cdot) = ||\boldsymbol{a}^* - \pi(\cdot)||_2^2 + \lambda||\boldsymbol{a}^* - \pi(\cdot)||_1$ where we use $\lambda = 0.1$. We scale $\boldsymbol{a}^*$ to equalize 1.0m end-effector translation, 0.1 radians end-effector rotation, and 1.0m gripper translation.

Training for Feedback through Data Augmentation

We introduce a simple technique which we have found to be effective in at least partially addressing a primary issue in imitation learning: the issue of cascading errors [22]. While other works have shown that injecting noise into the dynamics either during imitation learning [23] or sim-to-real transfer [24] can alleviate cascading errors, we provide a simple method based only on data augmentation. This method does not address recovering from discrete changes in the environment, but can address local feedback stabilization.

Consider the output of our policy in a global frame, $\boldsymbol{a} = (T_{\text{cmd}}, w_{\text{gripper}})$, which we can acquire from $T_{\Delta,\text{cmd}}$ since we know the end-effector pose. As previously mentioned a low-level controller tracks these setpoints, thus our learned policies can stabilize a trajectory by commanding the same global-frame setpoint $\boldsymbol{a}$ in the face of small disturbances to the robot state. If we want our policy to command the same setpoint in the face of a slightly perturbed robot state $\tilde{\boldsymbol{o}}_{\text{robot}}$ we can simply use $((\boldsymbol{o}_{\text{image}}, \tilde{\boldsymbol{o}}_{\text{robot}}), \boldsymbol{a})$ as an observation-action pair. These noise-augmented observation-action pairs are generated on-the-fly during training. A remaining question, of course, is what scale of noise is appropriate. In practice given our robot's scale and typical speeds we find $\tilde{\boldsymbol{o}}_{\text{robot}} \sim \mathcal{N}(\boldsymbol{o}_{\text{robot}}, I\boldsymbol{\sigma})$ with σ_i of 1mm, 1 degree, and 1cm works well respectively for translational, rotational, and gripper components.

Policy Models

We use two standard classes of policy models, Multi-Layer Perceptrons (MLP) and Long Short-Term Memory (LSTM) recurrent networks, which are familiar model classes to many different types of machine learning problems and in particular have been demonstrated to be viable for real-world visuomotor control [1, 2, 8]. In our evaluations the MLPs are only provided current observations, $\pi(\boldsymbol{o}_t)$, whereas through recurrence the LSTMs use the full observation sequence. The Appendix provides more model details.

S.4 Multi-View Time-Synchronized Correspondence Training

Unlike in previous work which trained robotic-supervised correspondence models only for static environments [18], we now would like to train correspondence models with dynamic environments. Other prior work [25] has used dynamic non-rigid reconstruction [26] to address dynamic scenes. The approach we demonstrate here instead is to correspond pixels between two camera views with images that are approximately synchronized in time, similar to the full-image-embedding training in [12], but here for pixel-to-pixel correspondence.

For training, like the static-scene case, finding pixel correspondences between images requires only depth images, camera poses, camera intrinsics and relative camera intrinsics. Autonomous object masking can, similar to [18], be performed using 3D-based background subtraction, using only the live depth sensors' point clouds. Since both (a) the time-synchronized technique can only correspond between time-synchronized images rather than many different static-scene views ([18] used approximately 400]), and (b) the time-synchronized technique does not have access to highly accurate many-view-fused 3D geometry as used in [18], it was unclear that the multi-view time-synchronized method would provide good results. To encourage generalization despite having using only two static views, we add rotation, scale, and shear image augmentations, and to help alleviate incorrect correspondences due to noisy depth images, we add photometric-error-based rejection of correspondences.

S.5 Simulation Experiments

We perform simulated imitation learning experiments on the tasks shown in Figure 2 (a). The first two tasks involve reaching to an object whose configuration varies between trials either in translation only, or rotation as well. The additional two tasks are both pushing tasks, which require feedback due to simulated external disturbances. For expert demonstrations in simulation, we use simple hand-designed policies using ground truth object state information. In all tasks both cameras were used for correspondence training, but we only use one RGB camera for policy input. Our simulation environment was configured to closely match our real hardware experimentation. Using Drake [27], we simulate the 7-DOF robot arm, gripper, objects, and multi-view RGBD sensing with two cameras as shown in Fig. 2.

Simulation Tasks: ***Reach T only***: goal is to move the end-effector to a given target position relative to the sugar box object in the scene; success is within 1.2cm of target. The box pose only varies in translation, not rotation. ***Reach T + R***: same as *Reach T only* but now the box pose varies in rotation as well; success is within 1.2cm and 2 degrees. ***Push box***: goal is to push the box object across the table, and the box is subject to random external disturbances; success if translated across table and final box orientation is within 2 degrees of target. ***Push plate***: goal is to push a plate across a table to a specific goal location, and the plate is subject to external disturbances; success if plate center is within 1cm of target position.

We use these tasks to compare the generalization performance of behavior-cloned policies where the only difference is how the "visual representation" $\boldsymbol{z}$ is acquired. The compared methods are:

1. *Ground truth 3D points (GT-3D):* $\boldsymbol{z}$ is ground truth world-frame 3D locations of points on the object.
2. *Ground truth 2D image coordinates (GT-2D):* $\boldsymbol{z}$ is similar to the previous baseline, but the points are projected into the camera using the ground truth camera parameters.
3. *Autoencoder (AE):* $\boldsymbol{z}$ is the encoding of a pre-trained autoencoder, similar to the visual training in [3, 6].
4. *End-to-End (E2E):* $\boldsymbol{z}$ is the intermediate representation from end-to-end training. This closely resembles the visual training and models in [1, 2], but we do not also add pose-based losses, in order to investigate end-to-end learning without these auxiliary losses.
5. *Ours, Dense descriptor 2D image coordinates (DD):* $\boldsymbol{z}$ is the expected image-space locations of the descriptor set $\{\boldsymbol{d}\}_i$, where the visual model was trained on dense correspondence.

Note that the two vision-based baselines (3) and (4) share an identical model architecture for producing $\boldsymbol{z}$, and differ only in the method used to train the parameters. The model is close to those in [1–3] with the key architectural traits of having a few convolutional layers followed by a channel-wise spatial expectation operation, which has been widely used [5, 6, 16, 17, 28–30]. See the Appendix for additional model and task details.

The binary success metrics of Figure 2(b) do not fully convey the methods' performances. We also experiment with varying the number of demonstrations, and characterize the performance distributions. By plotting the performance for the *"Reach, T + R"* task over a projection of the sampled object configurations (Figure 4), we learn that the few failures of our method occur when the box position lies outside the convex hull of the training data. Interestingly the GT-2D baseline also struggles with similar failure modes, while the GT-3D method succeeds in more cases outside the convex hull. This suggests that policies that consume 3D information are better able to extrapolate outside the training distribution. The baseline vision-based methods do not generalize as well; for example, the E2E performance distribution is shown in Figure 4. On this task we find that with just 30 demonstrations our method outperforms both AE and E2E with 200 demonstrations.

The pushing tasks are of particular interest since they demand closed-loop visual feedback. Disturbances are applied to the object both while collecting demonstrations and deploying the learned policies. Since the *"Push box"* task used a dynamic state feedback controller to provide demonstrations, we find that we need the sequence model (LSTM) for the policy network to achieve the task, even when the policy has access to ground truth object state. On the other hand, the *"Push plate"* task employed a static feedback controller to provide demonstrations, and so MLP models that consume only the current observation, $\pi_{\theta_p}(\boldsymbol{o}_t)$, are sufficient.

Interestingly all methods performed well on the *"Push box"* task while large differences were evident in the *"Push plate"* task. We speculate that this is because higher precision is required to accurately

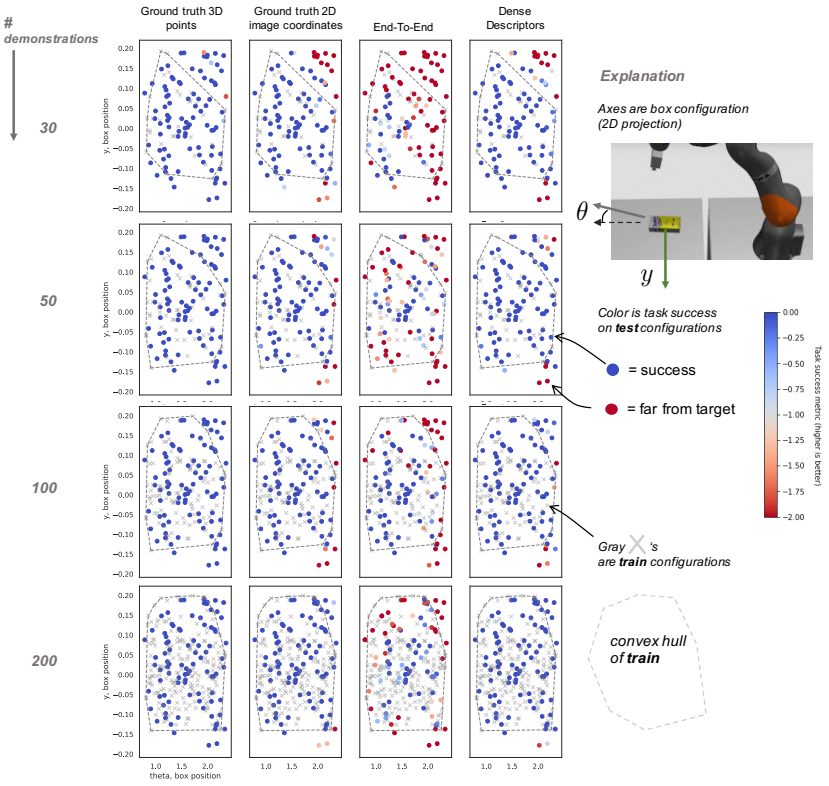


Figure 4: Task success distribution plotted over the 2D projection of the varied box configurations for the "Reach, T + R" task. The color of each point represents the result of deploying the learned policy with the object at that θ, y. Specifically the color encodes the distance to target threshold: $\min(0, -(\Delta\text{translation} + \Delta\text{rotation}) + \epsilon)$, where ϵ is the success threshold. The x coordinate is not shown in order to plot in 2D. Dark blue corresponds to perfect performance on the task with the object in that configuration, red is poor performance. Note that the color scale cuts off at -2 in order to highlight differences in the range [-2,0]. Each gray "x" in each subplot represents the configurations of the box in the training set, for either (from top to bottom) 30, 50, 100, or 200 demonstrations. The dashed gray line shows the convex hull of the respective training sets.

push the plate as compared to the box. Since the rectangular robot finger experiences a patch contact with the box, while only a point contact with the plate, there is more open loop stability in pushing the box. On the harder *"Push plate"* task we found that our method performed almost as well as the GT-2D baseline and significantly outperformed both the AE and E2E approaches.

S.6 Real Hardware Experimentation

We used a Kuka IIWA LBR robot with a Schunk WSG 50 parallel jaw gripper to perform imitation learning for the five tasks detailed in Figure 3. RGBD sensing was provided by RealSense D415 cameras rigidly mounted offboard the robot and calibrated to the robot's coordinate frame. Note that for effective correspondence learning between views, it is ideal to have views with *some* overlap such that correspondences exist, but still maintain different-enough views. Our hardware setup included three available cameras, but for each task only two views were used for correspondence training, and policy training used only a single monocular RGB stream as image input to the policies. Human demonstrations were provided by teleoperating the robot with a mouse and keyboard. All real hardware experiments use LSTM policy networks, since we suspect our human demonstrators use dynamic internal state.

As in simulation, we *only use demonstration data* for both visual training and policy learning; no additional data collection is needed. While the simulation results provide a controlled environment for comparisons, there are a number of additional challenges in our real world experiments: (i) visual complexity (textures, lighting, backgrounds, clutter), (ii) use of human demonstrations rather than

Task	Success criterion	Trained with disturbances	*Without disturbances* # attempts	# success	%	*With disturbances* # attempts	# success	%	*Demonstration data* # total	time (min.)
Push sugar box	box is < 3 cm from finish line	yes	6	6	100.0	70	68	97.1	51	13.9
Flip shoe, single instance	shoe is upright	no	43	42	97.7	40	35	87.5	50	6.5
Flip shoe, class-general										
previously seen shoes (14)	shoe is upright	no	43	38	88.4	–	–	–	146	17.5
novel shoes (12)	shoe is upright	no	22	17	77.3	–	–	–	146	17.5
Pick-then-hang hat on rack	hat is on the rack	yes	–	–	–	41	28	68.3	52	11.5
Push-then-grab plate	plate is off the table	yes	22	21	95.5	27	22	81.5	94	27.4
Total			136			181				

Table 1: Summary of task attempts and success rates for hardware validation experiments. Autonomous re-tries are counted as successes.

expert simulation controllers, (iii) real physical contact, and (iv) imperfect correspondence learning due to noisy depth sensors and calibration. Our hardware experiments test all of these aspects.

S.7 Model Architectures and Training Details

Policy Networks: All experiments using an ***"MLP"*** had a two-layer network with 128 hidden units, 20% dropout, in each layer and ReLU nonlinearities. Training was 75,000 steps with RMSProp, $\alpha = 0.9$, with a batch size of 16, and *lr* starting at $1e-4$, and decaying by a factor of 0.5 every 10,000 steps. All experiments using an ***"LSTM"*** had a single LSTM layer with 100 units preprocessed by two MLP layers of 100 units, 10% dropout, and layer-normalized prior to the LSTM layer. Training was 200,000 steps with RMSProp, $\alpha = 0.9$, with *lr* starting at $2e-3$, decaying 0.75 every 40,000 steps, with truncated backpropagation of maximum 50 steps, and gradient clipping of maximum magnitude 1.0. As recommended in [8] we train LSTMs on downsampled trajectories, we use 5 Hz.

Vision Networks: All networks used $Z = 32$. Both ***"AE"*** and ***"E2E"*** methods used an identical architecture, with the only difference being the additional decoder used for the AE method during autoencoding. The network is almost exactly as in [1] and [3], but we provided a full-width image, 320×240. ***"DD"*** architecture is identical to [18]. The ***"E2E (34-layer)"*** network is exactly the DD architecture but with $D = 16$ and channel-wise 2D spatial softmax to obtain $\boldsymbol{z}$.