ZERO-SHOT VISUAL IMITATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing approaches to imitation learning distill both what to do—goals—and how to do it—skills—from expert demonstrations. This expertise is effective but expensive supervision: it is not always practical to collect many detailed demonstrations. We argue that if an agent has access to its environment along with the expert, it can learn skills from its own experience and rely on expertise for the goals alone. We weaken the expert supervision required to a single visual demonstration of the task, that is, observation of the expert without knowledge of the actions. Our method is "zero-shot" in that we never see expert actions and never see demonstrations during learning. Through self-supervised exploration our agent learns to act and to recognize its actions so that it can infer expert actions once given a demonstration in deployment. During training the agent learns a skill policy for reaching a target observation from the current observation. During inference, the expert demonstration communicates the goals to imitate while the skill policy determines how to imitate. Our novel skill policy architecture and dynamics consistency loss extend visual imitation to more complex environments while improving robustness. Our zero-shot imitator, having no prior knowledge of the environment and making no use of the expert during training, learns from experience to follow experts for navigating an office with a TurtleBot and manipulating rope with a Baxter robot. Videos and detailed result analysis available at `https://sites.google.com/view/zero-shot-visual-imitation/home`

1 INTRODUCTION

Carrying out a task by imitation should be an attractive approach for both the expert and the agent. The expert should be able to simply demonstrate tasks capably without lots of effort, instrumentation, or engineering. Collecting many demonstrations is time-consuming, exact state-action knowledge is impractical, and reward design is involved and takes more than task expertise. The agent should be able to achieve goals given demonstrations without having to devote learning to each and every task. To address these issues we recast learning from demonstration into *doing from demonstration* by (1) only giving demonstrations during inference and (2) restricting demonstrations to visual observations alone rather than full state-actions. Instead of imitation learning, our agent must learn to imitate.

Imitating after observing just one expert demonstration is challenging and out-of-reach for existing methods. Behavioral cloning, or imitation learning as supervised learning from states to actions, cannot learn without full state-actions. Inverse reinforcement learning, or imitation learning by learning a reward and optimizing for it by reinforcement learning, cannot fit a general reward to so little data. One-shot imitation learning, though it only takes one demonstration of a task during inference, still requires full state-actions and a wealth of demonstrations during learning.

In general, a single demonstration is not enough for end-to-end optimization. Furthermore there is tacit knowledge that is difficult to fully capture by demonstration, such as the dynamics of the environment, inverse dynamics of actions, and visual representation. To compensate for the limited expert supervision in our setting, our agent learns these shared aspects of the environment by exploration and self-supervision. Autonomously learning what can be done equips the agent with task priors for imitation.

To illustrate these difficulties consider the tasks involved in navigating an office. Tasks could have spatial goals, finding locations, or semantic goals, finding certain kinds of places or things. Whatever

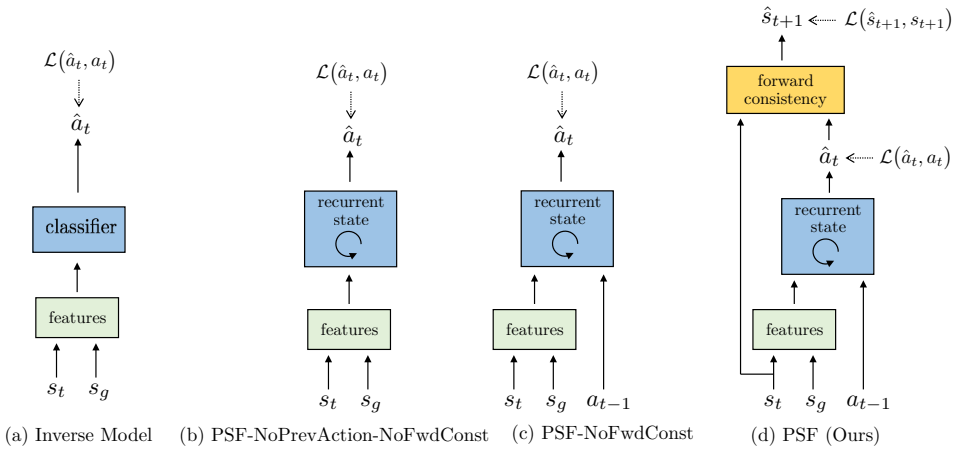


Figure 1: Different goal-conditioned skill architectures, or parametric skill functions (PSFs). A PSF takes an image of a goal state and learns to output a sequence of actions that would lead to goal starting from current state. (a) Vanilla inverse model [Nair etal 2017] (b) PSF without previous action history and no forward consistency (c) PSF with previous action history but no forward consistency (d) Our forward-consistent PSF.

the particulars, navigation in this setting requires coping with obstacles and a dynamic environment of changing furniture and unpredictable people. More difficult yet would be to navigate through the office next-door, having never explored it, after only one demonstration of how to find the meeting room. At the most extreme, the task could be given as an end goal without a demonstration for reaching it. The navigation and search skills needed to do this task are not covered by a shortest path demonstration by an expert, much less only observing the goal. However, an agent that learns its own skills from experience could succeed: our method navigates a TurtleBot through a new, unseen environment to reach a goal.

Could these issues be solved by inverse dynamics alone? While inverse dynamics has a key role in our method, it is insufficient on its own. Learning inverse dynamics by self-supervision of state-actions from exploration and then directly inferring expert actions is appealing in its simplicity. Nevertheless it is less general and less robust than our full approach. One-step inverse dynamics can only follow dense demonstrations with an expert transition at every timestep and cannot connect sparse goals. When actions are subject to noise, or the expert action is difficult to infer, following the demonstration step-by-step is prone to drift and compounding error. Should conditions have changed since the demonstration, it may no longer be feasible to follow the expert action-for-action to achieve the goal. Recovering the expert's actions is not enough.

Our approach is to first explore the environment, learn forward and inverse dynamics models, and then condense experience into an goal-parameterized skills policy for reaching a given observation. Learning both forward and inverse dynamics yields an actionable representation that captures what the agent can do and might be called on to do for imitation. Having learned its skills policy once, our agent can accept the visual observations of a demonstration as its goals and take actions to achieve them. Because a goal can take multiple and variable numbers of steps to reach, our agent must be capable of taking extended action and recognizing once it is done. To these ends we jointly optimize the skills policy with a stopping criterion for determining when a goal is satisfied in order to go to the next goal in turn. This combination makes imitation more robust, since the imitator can make an attempt but keep trying if the goal is not yet reached. By focusing on the consistency of imitator and expert dynamics, we can mitigate the multi-modality of inverse dynamics in which different trajectories can reach the same successors. Our skills policy architecture, dynamics consistency loss, and twin forward and inverse dynamics all contribute to successful imitation.

Experimental results show that our imitator follows visual demonstrations on real-world robots for office navigation and rope manipulation tasks. For navigation we steer a simple wheeled robot around known and novel offices according to dense, dense, sparse, and final step goal demonstrations. These tasks are carried out in partially-observable, dynamic environments: size, field of view,

and occlusion limit observation. Our approach improves on classical landmark and structure from motion methods as well as learned one-step inverse dynamics modelling. Furthermore, the navigation skills learned generalize to demonstrations in an unseen environment with new goals. For manipulation we select a series of grasp positions, orientations, and displacements for arranging a rope in a fixed, fully-observed workspace following dense demonstrations. Our approach nearly doubles the success rate for rope manipulation over the prior best visual imitation method.

2 RELATED WORK

Our work is closely related to imitation learning, but we address a different problem statement that gives less supervision and requires generalization across tasks during inference.

Imitation Learning The two main threads of imitation learning are behavioral cloning (Pomerleau, 1989), which directly supervises the mapping of states to actions, and inverse reinforcement learning (Abbeel & Ng, 2004; Ho & Ermon, 2016; Levine et al., 2016; Ng & Russell, 2000; Ziebart et al., 2008), which recovers a reward function that makes the demonstration optimal (or nearly optimal). Behavioral cloning of course requires actions, and inverse reinforcement learning is most commonly demonstrated with state-actions, though in principle state occupancy could be sufficient for fitting the reward. All of these methods require many demonstrations to do a new task. One-shot imitation learning (Duan et al., 2017) only takes a single demonstration of a new task for inference, but requires many demonstrations of full state-actions for learning.

Visual Demonstration The common definition of demonstration is to have full knowledge of expert states and actions, but this degree of supervision can be impractical, and instead relaxed to visual observations of the expert. Nair et al. (2017) observe a video of the expert manipulating rope and then reproduce the manipulation. Sermanet et al. (2016) perform reinforcement learning on a visual reward based on pre-trained recognition features. Third person imitation learning (Stadie et al., 2017) and imitation-from-observation (Liu et al., 2017) learn to translate expert observations into agent observations such that they can do policy optimization to minimize the distance between the agent trajectory and the translated demonstration. These methods and ours all make use of visual demonstrations, but all except ours and Nair et al. (2017) require demonstrations for learning.

Forward and Inverse Dynamics Forward and inverse dynamics are supervisory signals for representation learning. Modeling these dynamics in nonlinear features can help cope with visual observations to guide policy optimization. Watter et al. (2015) implement model-based control over visual inputs by mapping observations to achieve simpler, latent dynamics. Agrawal et al. (2016) jointly optimize forward and inverse dynamics models in a shared deep architecture. Inverse dynamics selects actions according to current and goal observations while forward modeling regularizes learning. Nair et al. (2017) extends this approach to following visual demonstrations of rope manipulation and serves as a baseline to our work. Pathak et al. (2017) forms a self-supervised intrinsic reward from these forward and inverse models to explore rich environments and even address tasks without any extrinsic reward. Our approach learns not only dynamics models but a skills policy that is conditioned on that can take multiple steps of actions, and we further supervises this policy with dynamics consistency over time.

Goal Conditioning By parameterizing the value or policy function with a goal, an agent can learn and do multiple tasks. In standard imitation learning, every task must be learned from its own demonstrations one-at-time. Universal value functions (Schaul et al., 2015) learn a family of value functions conditioned on goals. We adopt the clear analogy for policies, by giving our skills policy current and goal observations to decide how to act. Hindsight experience replay (Andrychowicz et al., 2017) learns goal-parameterized value functions for sparse reward tasks by augmenting the true goal and reward with "hindsight" goals drawn from experience. During learning we likewise draw goals from experience, but we optimize our skills policy by supervised learning and dynamics modeling instead of reinforcement learning. In our setting, all of the expert goals are followed zero-shot since they are only revealed after learning.

Classical methods for making robots that can learn by watching experts decompose the problem into visual parsing and planning (Breazeal & Scassellati, 2002; Dillmann, 2004; Ikeuchi & Suehiro, 1994; Kuniyoshi et al., 1989; 1994; Yang et al., 2015). Visual parsing of real-world scenes is complicated, and errors are likely to propagate into planning. Motion planning is computationally

intensive and can be sensitive to noise in state estimation. Designing and engineering the structured approaches required for both steps is demanding and difficult to improve through experience, unlike end-to-end optimized approaches.

3 LEARNING TO IMITATE WITHOUT EXPERT SUPERVISION

An agent without any prior knowledge will be unable to imitate a given task by watching a single visual demonstration by the expert in all but simplest of cases. The natural question therefore is: in order to imitate, what form of prior knowledge must the agent possess? Consider the example of rope manipulation where a visual demonstration of knot-tying is given. To be able to imitate such a complex task, the agent would need to have a knowledge of how the rope would react to its own motor controls, and then this knowledge could be used to explicitly plan the trajectory. However, this knowledge is difficult to design, particularly for non-rigid objects like rope. In another instance, if the demonstration illustrated a route to navigate, prior knowledge about 3D geometry could be used to construct a map (i.e. visual structure from motion) and then a motion planning algorithm could be used to output a sequence of actions. However, for any such 3D geometry or SLAM approaches to work, the images in the demonstration would have to be dense enough to reliably maintain correspondence along the trajectory, and hence would not work if the demonstration only contained a sparse sequence of landmarks.

Instead of relying on any hand-coded prior knowledge, in this work, we present a general method that allows an agent to learn this knowledge in the form of skills while exploring its environment in a self-supervised manner. Our method is "zero-shot" in that we never see expert actions and never see demonstrations during learning. We show that our approach allows the robot to manipulate rope by imitating the human demonstration, and navigate to a desired location by following sparse visual landmarks provided by the demonstration in real world scenarios where classical methods fail to construct a map.

3.1 LEARNING THE PARAMETRIC SKILL FUNCTION

Our agent must learn a skill representation that will aid it to imitate. In our formulation, skills take the form of a parametric skill function (PSF) h, that takes as input a pair of visual observations of states (s_i, s_g) and outputs actions $(\vec{a}_\tau)$ required to reach the goal state (s_g) from the initial state (s_i):

$$\vec{a}_\tau = h(s_i, s_g; \theta_h) \tag{1}$$

where $(a_\tau : a_1, a_2 ... a_T)$ represents a sequence of actions of variable length (that is inferred by our model). We represent h by a deep network with parameters θ_h to be capable of learning complex mappings necessary to select the necessary actions from visual observations. In this form, h can be thought of as a variable-step inverse dynamics model or universal policy (Schaul et al., 2015) with the difference that s_g need not be the task end goal but an intermediate checkpoint.

Our agent learns this PSF h in a completely self-supervised manner using state-action trajectories $(s_1, a_1, s_2, a_2, ..., s_T)$ collected by itself as it explores its environment without any extrinsic supervision. At inference, our agent can use the learned PSF to output the actions required to follow the observation sequence from a demonstration. The inverse dynamics supervision of the PSF eliminates the need for knowledge of the expert actions while the multi-step action policy from the PSF affords following sparse demonstrations of key states.

We first explain our one-step model and then extend it to a multi-step model of arbitrary, model-inferred length. One-step trajectories take the form (s_t, a_t, s_{t+1}) and the goal is to learn the function, $\hat{a}_t = h(s_t, s_{t+1}; \theta_h)$, where a_t is the action executed by the agent, $\hat{a}_t$ is the predicted action. The states s_t, s_{t+1} are first individually passed through a stack of convolution layers to obtain feature representation of the states $\phi(s_t), \phi(s_{t+1})$ respectively. These states are then concatenated and passed through a series of fully connected layers, the last of which has the output $\hat{a}_t$. The parameters θ_h can be estimated by minimizing loss $\mathcal{L}(a_t, \hat{a}_t)$ using stochastic gradient descent. The choice of the loss is critical because multiple actions can lead to the same result. If the action space is discrete, a standard loss function to use is cross-entropy which minimizes the following:

$$\mathcal{L}(a_t, \hat{a}_t) = \sum_i p(a_t^i | s_t, s_{t+1}) \log h(\hat{a}_t^i | s_t, s_{t+1}; \theta_h) \tag{2}$$

where p is the ground-truth distribution and h outputs the estimated distributions respectively. We don't have access to the true inverse dynamics distribution p, and only have access to the true action sample a_t executed by the robot. If p is inherently multi-modal and high-dimensional, it requires many samples to obtain a good estimate of p. This is in general the case because there are multiple actions that could lead to same final state given the initial one. For instance, consider rope manipulation, there could be multiple combinations of control actions (i.e. location, length, angle) that could deform the rope to match the final configuration. Similarly, in navigation, if robot is stuck against a corner, turning or moving forward all collapse to same effect, thereby making it extremely difficult to learn inverse dynamics. Moreover, this multi-modality issue becomes more critical as the length of trajectories grow, for instance, multiple paths may take the agent from initial state to final goal depending on environment dynamics.

3.2 Modeling action distribution via forward consistency

Our key insight is that, in many scenarios, we only care about whether the agent reached the final state or not and the exact trajectory is of lesser interest. Instead of penalizing the actions predicted by the inverse model, we propose to learn the PSF by minimizing the distance of state $\hat{s}_{t+1}$ which the predicted action $\hat{a}_t$ would lead to, given the initial state s_t. In addition to directly optimizing for reaching the visual goal location, a forward consistency objective is also helpful in dealing with multi-modality. Given deterministic environment and full observability, all modes of correct action trajectories will lead to the same state starting from initial state, unlike optimizing directly for action prediction. In expectation, action prediction loss should also ideally learn to model modes, but the forward consistency is more helpful in reducing the variance of gradients, thereby helping to learn better by naturally dealing with modes in action trajectories. Although, it is not immediately obvious as to how to operationalize such a formulation for the following two reasons: (a) we need the access to a good dynamics model which could reliably predict the next state given the action and the current state (b) such a dynamics model should be differentiable in order to train the PSF for state prediction error. Both of these issues could be resolved if an analytic formulation of forward dynamics is known,

However, in many scenarios of interest, especially if states are represented as images, an analytic forward model is not available. In this work, we propose to learn the forward dynamics model from the data. The forward dynamics is defined as $\hat{s}_{t+1} = f(s_t, \hat{a}_t; \theta_F)$ and learned jointly with the PSF h to optimize the following objective:

$$\begin{aligned} \min_{\theta_h, \theta_f} \quad & \lambda \| s_{t+1} - \hat{s}_{t+1} \|_2^2 \quad + \quad \mathcal{L}(a_t, \hat{a}_t) \\ \text{s.t.} \quad & \hat{s}_{t+1} = f(s_t, \hat{a}_t; \theta_f) \\ & \hat{a}_t = h(s_t, s_{t+1}; \theta_h) \end{aligned} \tag{3}$$

where $\hat{s}_t$ is the state predicted by the forward dynamics model f. Note that, in this formulation, the output of the inverse model is evaluated by the forward model f and the error in reaching the desired state is used to update the skill function h. Note that learning θ_h, θ_f jointly from scratch is not ideal because the forward model f is not good in the beginning, and hence makes the gradient updates noisier for h. To alleviate this issue, we first pre-train the forward model and PSF separately by blocking the gradient flow, and then fine-tune jointly.

We extend our one-step optimization to variable length sequence of actions in a straightforward manner by having a multi-step PSF h model with a step-wise forward consistency loss. The PSF h maintains an internal memory of the system and outputs actions conditioned on current state s_t, starting from s_i to reach goal state s_g. The forward consistency loss is computed at each time step, and jointly optimized with the action prediction loss over the whole trajectory. The final objective can be written as follows:

$$\min_{\theta_h, \theta_f} \sum_{t=i}^{t=T} \Big(\lambda \| s_{t+1} - f(s_t, \hat{a}_t; \theta_f) \|_2^2 \quad + \quad \mathcal{L}(a_t, h(s_t, s_{t+1}; \theta_h)) \Big) \tag{4}$$

We implement this multi-step *forward-consistent PSF* using a recurrent neural network which at every step takes as input the feature representation of the current ($\phi(s_t)$) state, goal ($\phi(s_g)$) states, action at the previous time step (a_{t-1}) and the internal hidden representation h_{t-1} of the recurrent units and predicts $\hat{a}_t$. Note that inputting the previous action to PSF at each time step is redundant

given that hidden representation is already maintaining a history of the trajectory. Nonetheless, it is helpful in explicitly modeling the history of trajectory. This formulation amounts to building an auto-regressive model of the joint action that estimates probability $P(a_t|s_1, a_1, ...a_{t-1}, s_t, s_g)$ at every time step. Further potential generalization of our *forward-consistent PSF* is to extend forward dynamics to incorporate multiple steps of trajectory, but we leave that direction of future work.

3.3 Goal Satisfaction

Since the skill policy has to be able to take variable number of actions in pursuit of a goal in case of error or noise, it is indispensable to have the ability to identify when the current goal has been realized. Such a *stopping criterion* becomes more critical while having to transit through a sequence of intermediate goal states, otherwise the error accumulation would quickly lead to divergence from the demonstration. However, it is not immediately obvious as to how to categorize a good stopping criterion for two major reasons. Firstly, the environment might have changed since the demonstration was taken or the agent might have taken a different trajectory such that it is impossible to exactly match the goal state. Secondly, knowing when to output stop is a function of agent's dynamics and its interplay the environment, for instance, visual changes differently in moving forward or turning in navigation.

In this work, we learn a robust goal recognition network by classifying transitions collected during exploration. We sample states at random, and for every sampled state make positives of its temporal neighbors, and make negatives of the remaining states more distant than a certain margin. We optimize our goal classifier by cross-entropy loss. The decision to advance to the next goal could be modeled as yet another action, and training auto-regressively, but we found that making goal recognition an independent output worked consistently better.

3.4 Ablations and Baselines

In the previous section, we progressively built up to our full PSF. Our formulation contains following modular concepts: (a) recurrent variable-length skill policy, (b) explicitly encoding previous action in the recurrence, (c) forward-consistent loss function optimization, (d) stopping criterion to decide goal satisfaction. We systematically ablate our proposed *forward-consistent parametric skill function*, quantitatively review the importance of each component, and perform comparisons to the existing methods which could be deployed for the task of visual imitation.

We evaluate our approach on visual navigation and rope manipulation. Particularly, we compare against the existing classical work and against our own ablations. Following methods will be evaluated and compared to in the subsequent experiments section: (1) Classical methods: In visual navigation, we attempted to compare against the state-of-the-art open source classical methods, namely, ORB-SLAM2 (Mur-Artal & Tardós, 2017) and Open-SFM (mapillary, 2016). (2) Inverse Model: Nair et al. (2017) leverage inverse dynamics to follow demonstration in rope manipulation setup. We compare to their method in both visual navigation and manipulation. (3) PSF-NoPrevAction-NoFwdConst is the ablation of our recurrent PSF without previous action history and without forward consistency. (4) PSF-NoFwdConst refers to our recurrent PSF with previous action history, but without forward consistency objective. (5) PSF refers to our complete method with forward consistent parametric skill function optimization. We now discuss the experiments and evaluate these baselines.

4 Results

The two key requirements of a good visuomotor skill function are that it should generalize to unseen environments and new goals while staying robust to irrelevant distractors in the observations. We evaluate our model by testing its performance on two real world robotic platforms: navigation of a wheeled robot in a cluttered office environments and manipulating rope with a humanoid robot. For navigation we check generalization by testing on a novel building/floor. For rope manipulation, we check generalization across difficulty by testing our PSF across complex manipulation tasks like making a given configuration or tying knots which were not seen during random exploration.

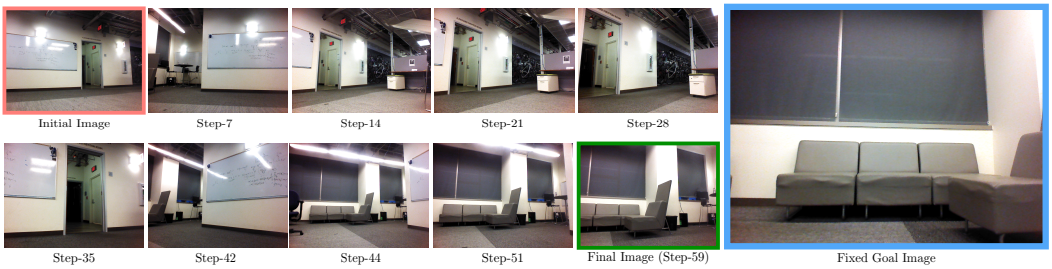


Figure 2: We tested the performance of our model at navigation by tasking the TurtleBot to reach the target location depicted by the goal image shown in the right of the figure. The initial view of the TurtleBot shown in top-left corner (orange border) has no overlap with the goal image. The sequence of images from left to right and top to bottom shows from the perspective of the TurtleBot the visual observations it encounters as it searches for the goal. Note that initially the robot turns left and right searching for the goal and after some exploration it successfully reaches the target location.

4.1 Navigation in Indoor Office Environment

A natural way to command a robot to move in an indoor office environment is to ask it to go near a certain location such as the refrigerator or a certain person's office. Instead of using language to command the robot, in this work, we communicate with the robot by either showing it a single target image if the goal is close or a sequence of way points if the goal is faraway. In each scenario, the robot needs to automatically determine the motor commands it needs to apply to move from the current observation to the goal.

Robotics Setup: We used the TurtleBot2 platform consisting of a wheeled Kobuki base and an ORBBEC Astra 3D camera for all our experiments. Note that we only used the raw RGB images. The robot's action space was moving forward, turning left, turning right, and standing still (i.e. no-op). The forward motion covers approximately 10cm and the turning motions approximately 14-18 degrees. These numbers vary due to the use of velocity control. A powerful on-board laptop was used to process the images and infer the motor commands.

Data Collection: In order to learn the PSF, a randomized and automated scheme for data collection was devised that required no human supervision. The robot sampled a no-op with probability 0.05 and other three actions with a randomly uniform probability. If the sampled action was no-op, an action repeat value between 1-2 steps was sampled, whereas if other actions were chosen, an action repeat value between 1-5 steps was randomly chosen. The robot autonomously repeated this process and collected 200K interactions from three floors of a building. If the robot crashed into an object, it performed a reset maneuver on which it first moves backwards and then turns right/left by a random angle between 90-270 degrees.A separate floor of the building with substantially different furniture layout and visual textures was used for testing the learned model.

Self-Supervised Learning: We collected 200K samples from 2 floors of an academic building. These samples were then used to train our *recurrent forward-consistent PSF*. The trunk of the model is a ImageNet pre-trained ResNet-50 (He et al., 2016) network. Input are the images and output are the actions of robot. Forward consistency model was pre-trained and then fine-tuned together with the PSF. The loss weight of forward model was 0.1, and the objective was minimized using Adam (Kingma & Ba, 2014) with learning rate of $5e-4$.

Experiment-1: Goal Finding In our first experiment, we test if the PSF learned by the TurtleBot can enable it to find its way to a goal. To stress test this, we keep the TurtleBot approximately 20-30 steps away from the target location and orient it in a manner so that the initial and goal image depicting the target location have no overlap as shown in Figure 2. Since the current and goal images have no overlap, classical techniques such as structure from motion that rely on feature matching cannot be used to infer the robot's action. In order to reach the target, the robot must explore its surroundings. The robot's search is deemed to succeed if it reaches the goal and predicts to stop and it is deemed as failure if it crashes into furniture, does not reach the goal within 200 steps or

Model Name	Run Id-1	Run Id-2	Run Id-3	Run Id-4	Run Id-5	Run Id-6	Run Id-7	Run Id-8	Num Success
Random Search + Stop	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0
Inverse Model [Nair et. al. 2017] + Stop	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0
PSF-NoPrevAction-NoFwdConst + Stop	39 steps	34 steps	Fail	Fail	Fail	Fail	Fail	Fail	2
PSF-NoFwdConst + Stop	22 steps	22 steps	39	48	Fail	Fail	Fail	Fail	4
PSF + Stop (Ours)	119 steps	66 steps	144 steps	67 steps	51 steps	Fail	100 steps	Fail	6

Table 1: Quantitative evaluation of our PSF model and other baselines in the task of Goal finding where the agent was shown only the goal image and tasked to go there. Our PSF model outperforms all other baselines and ablations. For a fair comparison, we added our best learned stop criterion to all the baselines.

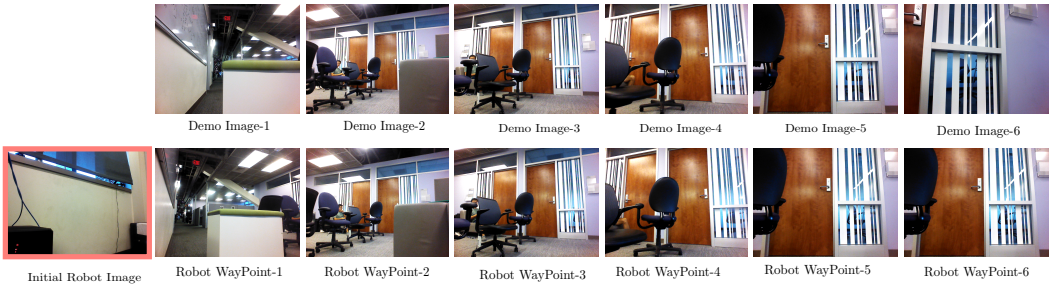


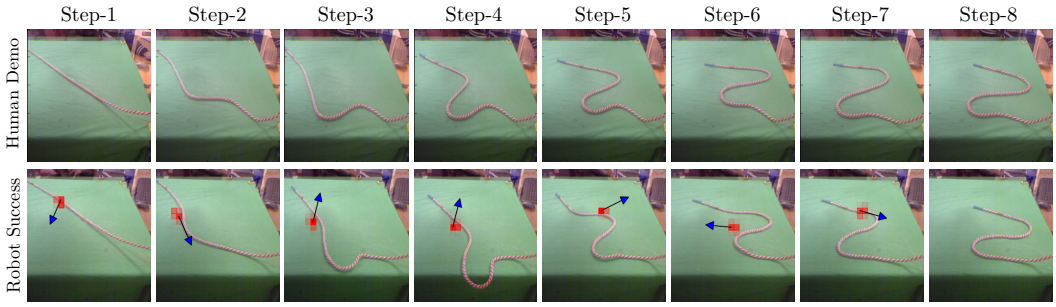
Figure 3: The first five checkpoints of the loop demonstration are shown here (top row), along with the perturbed initial state (approximately half a meter away and a full 180 degrees rotated) and the images the TurtleBot sees when it determines the checkpoint has been reached (bottom row). This case also exemplifies why classical methods fail; there are no possible keypoint matches between checkpoint 1 and checkpoint 2, and the difference between the initial state and checkpoint 1 is even more stark.

outputs ten consecutive no-ops when it is not the proximity of the target location. We perform this test on the floor of a building which the robot has never seen before. We find that our PSF model outperforms the baseline models in reaching the target location. Our model learns the exploratory behavior of turning at its location unless it sees in a part of its visual field some features of the goal image. While baseline models, also end up learning a similar exploration strategy, their exploration strategy ends up having a left or right turning bias which results in significantly worse performance. Results are shown in Table- 1. Videos can be found in the supplementary materials [1].

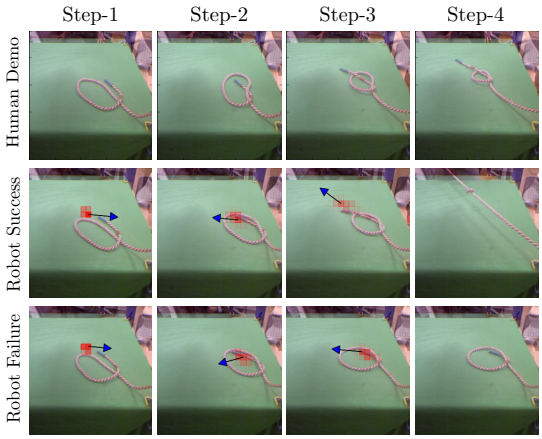
Experiment-2: Zero-shot Visual Imitation The results above show that the robot equipped with our model is able to search for goal locations even if they are out of field of view but are close to the robot. Since the robot approximately reaches the goal location, it was possible that when the robot was provided with a sequence of images it might drift away and be unable to follow the expert demonstration. Therefore, next we tested the performance of our robot on following a sequence of checkpoint images provided from a expert demonstration. The checkpoint images were generated by an expert that moved the robot from a starting to a goal location. The robot was then re-initialized to its starting position and every fifth image from the demonstration was provided to the robot. The rationale behind providing only the fifth image is that as the model of the robot improves, it would require lesser and lesser supervision from the expert. This means that sub-sampling (as shown in Figure 3) the sequence provides an easy way to tune the complexity of the task. We found that when we sampled every frame, our method and classical approaches such as structure from motion could both be used to follow the demonstration. However, at subsampling of five, SIFT based feature matching approaches did not work and ORBSLAM2 (Mur-Artal & Tardós, 2017) failed to generate a map, whereas our method outperformed both SIFT based and other baseline methods for learning a skill function. We will revisit these methods in the final version of paper.

Model Name	Maze Run 1	Maze Run 2	Maze Run 3	Perturbed Loop 1	Perturbed Loop 2	Perturbed Loop 3	Total Loop Successes
PSF-NoPrevAction-NoFwdConst + Stop	12/20 (109)	14/20 (184)	20/20 (263)	—	—	—	—
PSF-NoFwdConst + Stop	13/20 (147)	18/20 (325)	20/20 (166)	0/17 (0)	0/17 (0)	0/17 (0)	0%
PSF + Stop (Ours)	20/20 (353)	12/20 (194)	20/20 (168)	0/17 (0)	17/17 (243)	17/17 (165)	33%

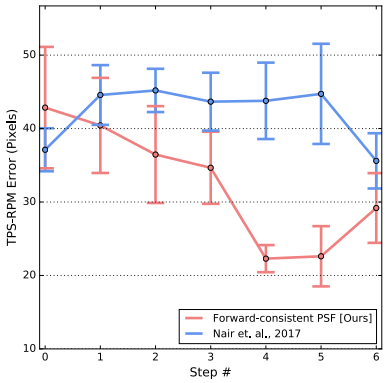
Table 2: Evaluation of our model against the baselines. The number of checkpoints reached is shown in the fraction, and the number of steps taken to reach that checkpoint is shown in brackets. The perturbed loop run is a scenario in which we give it a demonstration around a loop of furniture, but start it with a differing initial starting position and orientation. 6 trajectories were given to each tested model for the loop.



(a) Manipulating rope into an 'S' shape



(b) Manipulating rope into tying a knot



(c) TPS-RPM error for 'S' configuration

Figure 4: The two rows of the top image show sequence of images of rope manipulation provided by the human the sequences of states reached by the robot while imitating the demonstration. The robot is able to successfully imitate the demonstration and the performance of our method (measured as the TPS-RPM error metric) is significantly better than the baseline method as shown in the bottom right subplot. The bottom subplot figure shows success and failure examples of the robot attempting to tie a knot.

4.2 Vision-based Rope Manipulation

One very challenging task for robots is to manipulate deformable objects such as ropes. Even humans learn complex rope manipulation such as tying knots either by observing experts or receiving instruction. We test our proposed method for following visual demonstration provided by a human expert manipulating a rope into a complex 'S' shape and tying a knot. We closely follow the experimental setup of (Nair et al., 2017) and re-use their data that was collected by a Baxter robot interacting a rope kept on a table in front of it in a completely self-supervised manner. The robot interacted with the rope by using a pick and place primitive that chooses a random point on the rope and displaces it by a randomly chosen length and direction. The robot repeated this process

[1] https://sites.google.com/view/zero-shot-visual-imitation/home

70K times to collect 70K pairs of interaction data. We used this data to train the baseline inverse model that takes as inputs pair of images and outputs the action described in (Nair et al., 2017) and compare this against our PSF model. We used these trained models to follow visual demonstrations.

Model Implementation To keep the comparison fair, we keep the exact same architecture as described in (Nair et al., 2017). This model consists of a pre-trained AlexNet, whose features are fed into three different classification networks to predict the location of the grasp, and the direction and distance of the rope manipulation. A forward consistency loss was then applied to the predicted actions from the classification networks and features output by the AlexNet. Optimization was done using Adam (Kingma & Ba, 2014) with a learning rate of $1e-4$. The forward consistency loss weight was 0.1. For the first 40K iterations, the AlexNet weights were frozen, and then fine-tuned jointly.

Model Name	Mean Accuracy	Bootstrap standard deviation
Inverse Model [Nair et.al. 2017]	35.8%	8.3%
Forward-consistent PSF [Ours]	60.2%	8.6%

Table 3: Knot tying accuracy of rope manipulation. Our forward-consistency optimization doubles accuracy of knot tying, improving significantly over a direct inverse model baseline.

Accuracy Evaluation The performance of the model was evaluated at every step in the demonstration by measuring the non-rigid registration cost between the rope state achieved by the robot and the state demonstrated by the human at every step in the demonstration. The matching cost was measured using the thin plate spline robust point matching technique (TPS-RPM) described in (Chui & Rangarajan, 2003). While TPS-RPM provides a good metric for measuring performance for constructing the 'S' shape, it is not an apt metric for knots because the configuration of the rope in knot is 3D due intertwining of the rope, and fails to find the point correspondences. We therefore report accuracy of knot tying in Table-3. Note that the knot shape is never encountered during the self-supervised data collection phase and therefore performance accuracy on knot tying is a good measure of generalization of different skill functions.

5 Conclusion

Zero-shot imitators learn to follow demonstrations without any expert supervision during learning. Our approach learns task priors of representation, goals, and skills from the environment in order to imitate the goals given by the expert during inference. Equipped with these priors, our imitators can follow dense, sparse, or even final goals alone to accomplish a new task. We have shown real-world results for office navigation and rope manipulation but make no domain assumptions limiting the method to these problems. Our imitators sidestep the expertise bound on learning from demonstration by learning to imitate from their own experience.

Having defined zero-shot imitation and established these first results, there are several avenues of extension. Although we have focused on reducing expert time, since it is more precious than agent time, we could still improve efficiency of environment interaction. This could be done by more sophisticated exploration, as in self-supervised curiosity (Pathak et al., 2017), or by re-incorporating demonstrations into learning when they can be collected. The requirements on demonstrations could be relaxed further still to handle shifts in observations or actions between the agent and experts, as in third person imitation (Liu et al., 2017; Stadie et al., 2017) or other domain adaptation approaches. To refine our agent's skills, learning could be augmented with past experience (Agrawal et al., 2016) or hindsight experience replay (Andrychowicz et al., 2017). The combination of learning to imitate, curiosity, and hindsight together could further improve efficiency for both expert and imitator.

References

P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *NIPS*, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

Cynthia Breazeal and Brian Scassellati. Robots that imitate humans. *Trends in cognitive sciences*, 6 (11):481–487, 2002.

Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141, 2003.

Rüdiger Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2):109–116, 2004.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. 2016.

Katsushi Ikeuchi and Takashi Suehiro. Toward an assembly plan from observation. i. task recognition with polyhedral objects. *IEEE transactions on robotics and automation*, 10(3):368–385, 1994.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yasuo Kuniyoshi, Masayuki Inaba, and Hirockika Inoue. Teaching by showing: Generating robot programs by visual observation of human performance. In *Proc. of the 20th International Symp. on Industrial Robots*, pp. 119–126, 1989.

Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE transactions on robotics and automation*, 10(6):799–822, 1994.

Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv:1603.02199*, 2016.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *arXiv preprint arXiv:1707.03374*, 2017.

mapillary. Open source structure from motion pipeline. `https://github.com/mapillary/OpenSfM`, 2016.

Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 2017.

Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *ICRA*, 2017.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *ICML*, pp. 663–670, 2000.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pp. 305–313, 1989.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1312–1320, 2015.

Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015.

Yezhou Yang, Yi Li, Cornelia Fermüller, and Yiannis Aloimonos. Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web. In *AAAI*, pp. 3686–3693, 2015.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pp. 1433–1438, 2008.