No IRL

Feature Extraction

Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation

YuXuan Liu[†*], Abhishek Gupta[†*], Pieter Abbeel[†‡], Sergey Levine[†]
[†] UC Berkeley, Department of Electrical Engineering and Computer Science
[‡] OpenAI
{yuxuanliu,abhigupta,pabbeel,svlevine}@berkeley.edu

***Abstract*— Imitation learning is an effective approach for autonomous systems to acquire control policies when an explicit reward function is unavailable, using supervision provided as demonstrations from an expert, typically a human operator. However, standard imitation learning methods assume that the agent receives examples of observation-action tuples that could be provided, for instance, to a supervised learning algorithm. This stands in contrast to how humans and animals imitate: we observe another person performing some behavior and then figure out which actions will realize that behavior, compensating for changes in viewpoint, surroundings, object positions and types, and other factors. We term this kind of imitation learning "imitation-from-observation," and propose an imitation learning method based on video prediction with context translation and deep reinforcement learning. This lifts the assumption in imitation learning that the demonstration should consist of observations in the same environment configuration, and enables a variety of interesting applications, including learning robotic skills that involve tool use simply by observing videos of human tool use. Our experimental results show the effectiveness of our approach in learning a wide range of real-world robotic tasks modeled after common household chores from videos of a human demonstrator, including sweeping, ladling almonds, pushing objects as well as a number of tasks in simulation.**

I. Introduction

Learning can enable autonomous agents, such as robots, to acquire complex behavioral skills that are suitable for a variety of unstructured environments. In order for autonomous agents to learn such skills, they must be supplied with a supervision signal that indicates the goal of the desired behavior. This supervision typically comes from one of two sources: a reward function in reinforcement learning that specifies which states and actions are desirable, or expert demonstrations in imitation learning that provide examples of successful behaviors. Both modalities have been combined with high-capacity models such as deep neural networks to enable learning of complex skills with raw sensory observations [1], [2], [3], [4]. One major advantage of reinforcement learning is that the agent can acquire a skill through trial and error with only a high-level description of the goal provided through the reward function. However, reward functions can be difficult to specify by hand, particularly when the success of the task can only be determined from complex observations such as camera images [5].

* These authors contributed equally to this work.

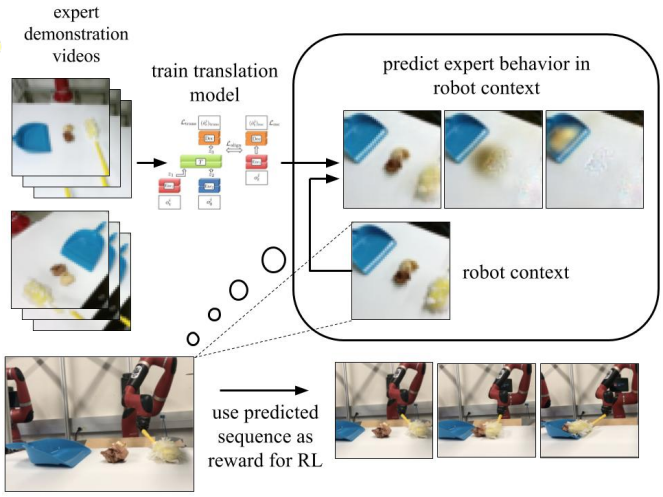

Fig. 1: Imitation from Observation using Context-Aware Translation. We collect a number of videos of expert demonstrations from a human demonstrator, and use them to train a context translation model. At learning time, the robot sees the context of the task it needs to perform. Then, the model predicts what an expert would do in the robot context. This predicted sequence is used to define a cost function for reinforcement learning thus enabling imitation from observation. The task shown here is illustrative of a wide range of tasks that we evaluate.

Imitation learning bypasses this issue by using examples of successful behavior. Popular approaches to imitation learning include direct imitation learning via behavioral cloning [4] and reward function learning through inverse reinforcement learning [6]. Both settings typically assume that an agent receives examples that consist of sequences of observation-action tuples, and try to learn either a function that maps observations to actions from these example sequences or a reward function to explain this behavior while generalizing to new scenarios. However, this notion of imitation is quite different from the kind of imitation carried out by humans and animals: when we learn new skills from observing other people, we do not receive egocentric observations and ground truth actions. The observations are obtained from an alternate viewpoint and the actions are not known. Furthermore, humans are not only capable of learning from live observations of demonstrated behavior, but also from

video recordings of behavior provided in settings considerably different than their own. Can we design imitation learning methods that can succeed in such situations? A solution to this problem would be of considerable practical value in robotics, since the resulting imitation learning algorithm could directly make use of natural videos of people performing the desired behaviors obtained, for instance, from the Internet.

We term this problem imitation-from-observation. The goal in imitation-from-observation is to learn policies only from a sequence of observations (which can be extremely high dimensional such as camera images) of the desired behavior, with each sequence obtained under differences in context. Differences in context might include changes in the environment, changes in the objects being manipulated, and changes in viewpoint, while observations might consist of sequences of images. We define this problem formally in Section III.

Our imitation-from-observation algorithm is based on learning a context translation model that can convert a demonstration from one context (e.g., a third person viewpoint and a human demonstrator) to another context (e.g., a first person viewpoint and a robot). By training a model to perform this conversion, we acquire a feature representation that is suitable for tracking demonstrated behavior. We then use deep reinforcement learning to optimize for the actions that optimally track the translated demonstration in the target context. As we illustrate in our experiments, this method is significantly more robust than prior approaches that learn invariant feature spaces [7], perform adversarial imitation learning [8], or directly track pre-trained visual features [9]. Our translation method is able to provide useful perceptual reward functions, and performs well on a number of simulated and real manipulation tasks, including tasks that require a robot to emulate human tool use. Videos can be found on `https://sites.google.com/site/imitationfromobservation/`

II. Related Work

Imitation learning is usually thought of as the problem of learning an expert policy that generalizes to unseen states, given a number of expert state-action demonstration trajectories [10], [11]. Imitation learning has enabled the successful performance of tasks in a number of complex domains such as helicopter flight through apprenticeship learning [12], learning how to put a ball in a cup and playing table tennis [13], performing human-like reaching motions [14] among others. These methods have been very effective however typically require demonstrations provided through teleoperation or kinesthetic teaching, unlike our work which aims to learn from observed videos of other agents performing the task. Looking at the imitation learning literature from a more methodological standpoint, imitation learning algorithms can largely be divided into two classes of approaches: behavioral cloning and inverse reinforcement learning.

Behavioral cloning casts the problem of imitation learning as supervised learning, where the policy is learned from state(or observation)-action tuples provided by the expert. In imitation-from-observation, the expert does not provide actions, and only provides observations of the state in a different context, so direct behavioral cloning cannot be used. Inverse reinforcement learning (IRL) methods instead learn a reward function from the expert demonstrations [6], [15], [16], [17]. This reward function can then be used to recover a policy by running standard reinforcement learning [18], [19], though some more recent IRL methods alternate between steps of forward and inverse RL [20], [21], [22], [8]. While IRL methods can in principle learn from observations, in practice using them directly on high-dimensional observations such as images has proven difficult.

Aside from handling high-dimensional observations such as raw images, our method is also designed to handle differences in context. Context refers to changes in the observation function between different demonstrations and between the demonstrations and the learner. These may include changes in viewpoint, object positions, surroundings, etc. Along similar lines, [7] directly address learning under domain shift. However, this method has a number of restrictive requirements, including access to expert and non-expert policies, directly optimizing for invariance between only two contexts (whereas in practice demonstrations may come from several different contexts), and performs poorly on the more complex manipulation tasks that we consider, as illustrated in Section VI. [9] proposes to address differences in context by using pretrained visual features, but does not provide for any mechanism for context translation, as we do in our work, relying instead on the inherent invariance of visual features for learning. Follow-up work proposes to further increase the invariance of the visual features through multi-viewpoint training [23]. [24] propose to learn robotic skills from first person videos of humans by using explicit hand detection and a carefully engineered vision pipeline. In contrast, our approach is trained end-to-end, and does not require any prior visual features, detectors, or vision systems. [25] proposed to use demonstrations as input to policies by training on paired examples of state sequences, however our method operates on raw observations and does not require any actions in the demonstrations, while this prior method operates only on low-dimensional state variables and does not deal with context shift like our method.

Our technical approach is related to work in visual domain adaptation and image translation. Several works have proposed pixel level domain adaptation [26], [27], [28], as well as translation of visual style between domains [29], by using generative adversarial networks (GANs). The applications of these methods have been in computer vision, rather than robotic control. Our focus is instead on translating demonstrations from one context to another, conditioned on the first observation in the target context, so as to enable an agent to physically perform the task. Although we do not use GANs, these prior methods are complementary to ours, and incorporating a GAN loss could improve the performance of our method further.

In our work we consider tasks like sweeping, pushing,

ladling(similar to pouring) and striking. Several prior methods have looked at performing tasks like these although typically with significantly different methods. Tasks involving cleaning with a brush, similar to our sweeping tasks was studied in [30] but is done using a low cost tool attachment and kinesthetic programming by demonstration. Besides [9], tasks involving pouring were also studied in [31] using a simple PID controller with a specified objective volume rather than inferring the objective from demonstrations. Similar flavors of tasks were also considered in [32], [33], but we leave those specific tasks to future work. Other work [34] also considers tasks of pushing objects on a table-top but uses predictive models on point-cloud data and uses a significantly different intuitive physics model with depth data.

III. Problem Formulation and Overview

In the imitation-from-observation setting that we consider in this work, an agent observes demonstrations of a task in a variety of *contexts*, and must then execute the demonstrated behavior in its own context. We use the term context to refer to properties of the environment and agent that can vary across demonstrations, which may include the viewpoint, the background, the positions and identities of objects in the environment, and so forth. The demonstrations $\{D_1, D_2, ...D_n\} = \{[o_0^1, o_1^1, ...o_T^1], [o_0^2, o_1^2, ...o_T^2], ..., [o_0^n, o_1^n, ...o_T^n]\}$ consist of observations o_t that are produced by a partially observed Markov process governed by an observation distribution $p(o_t|s_t, \omega)$, dynamics $p(s_{t+1}|s_t, a_t, \omega)$, and the expert's policy $p(a_t|s_t, \omega)$, with each demonstration being produced in a different context ω. Here, s_t represents the unknown Markovian state, a_t represents the action (which is not observed in the demonstrations), and ω represents the context. We assume that ω is sampled independently from $p(\omega)$ for each demonstration, and that the imitation learner has some fixed ω_l from the same distribution. Throughout the technical section, we use o_t^i to refer to the observation at time t from a context ω_i.

While a practical real-world imitation-from-observation application might also have to contend with systematic *domain shift* where, e.g., the learner's embodiment differs systematically from that of the demonstrator, and therefore the learner's context ω cannot be treated as a sample from $p(\omega)$, we leave this challenge to prior work, and instead focus on the basic problem of imitation-from-observation. This means that the context can vary between the demonstrations and the learner, but the learner's context still comes from the same distribution. We elaborate on the practical implications of this assumption in Section VI, and discuss how it might be lifted in future work.

Any algorithm for imitation-from-observation must contend with two challenges: first, it must be able to determine what information from the observations to track in its own context ω_l, which may differ from those of the demonstrations, and second, it must be able to determine which actions will allow it to track the demonstrated observations. Reinforcement learning (RL) offers a tool for addressing the latter problem: we can use some measure of distance to the demonstration as a reward function, and learn a policy that takes actions to minimize this distance. But which distance to use? If the observations correspond, for example, to raw image pixels, a Euclidean distance measure may not give a well-shaped objective: roughly matching pixel intensities does not necessarily correspond to a semantically meaningful execution of the task, unless the match is almost perfect. Fortunately, the solution to the first problem – context mismatch – naturally lends us a solution to the problem of choosing a distance metric. In order to address context mismatch, we can train a model that explicitly translates demonstrations from one context into another, by using the different demonstrations as training data. The internal representation learned by such a model provides a much more well-structured space for evaluating distances between observations, since proper context translation requires understanding the underlying factors of variation in the scene. As we empirically illustrate in our experiments, we can use squared Euclidean distances between features of the context translation model as a reward function to learn the demonstrated task, while using the model itself to translate these features from the demonstration context to the learner's context. We first describe the translation model, and then show how it can be used to create a reward function for RL.

IV. Learning to Translate Between Contexts

Since each demonstration D_k is generated from an unknown context ω_k, the learner cannot directly track these demonstrations in its own context ω_l. However, since we have demonstrations from multiple unknown but different contexts, we can learn a context translation model on these demonstrations without any explicit knowledge of the context variables themselves. We only assume that the first frame o_0^k of a demonstration in a particular context ω_k can be used to implicitly extract information about the context ω_k.

Our translation model is trained on pairs of demonstrations $D_i = [o_0^i, o_1^i ..., o_T^i]$ and $D_j = [o_0^j, o_1^j ..., o_T^j]$, where D_i comes from a context ω_i (the source context) and D_j comes from a context ω_j (the target context). The model must learn to output the observations in D_j conditioned on D_i and the first observation o_0^j in the target context ω_j. Thus, the model looks at a single observation from a target context, and predicts what future observations in that context will look like by translating a demonstration from a source context. Once trained, this model can be provided with any demonstration D_k to translate it into the learner's context ω_l for tracking, as discussed in the next section.

The model (Fig 2), assumes that the demonstrations D_i and D_j are aligned in time, though this assumption could be relaxed in future work by using iterative time alignment [35]. The goal is to learn the overall translation function $M(o_t^i, o_0^j)$ such that its output $M(o_t^i, o_0^j) = (\hat{o}_t^j)_{\text{trans}}$ closely matches o_t^j for all t and each pair of training demonstrations D_i and D_j. That is, the model translates observations from D_i into the context ω_j, conditioned on the first observation o_0^j in D_j.

The model consists of four components: a source observation encoder $\text{Enc}_1(o_t^i)$ and a target initial observation encoder $\text{Enc}_2(o_0^j)$ that encode the observations into source and target

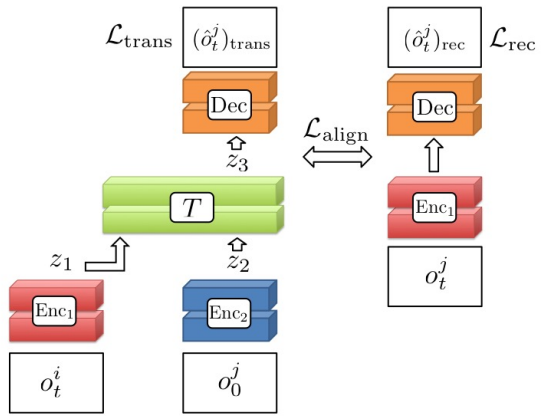

Fig. 2: Context translation model: The source observation o_t^i is translated to give the prediction of the observation in the target context $(\hat{o}_t^j)_{\text{trans}}$, given the context image o_0^j from the target context. The convolutional encoders are Enc_1 and Enc_2, while the deconvolutional decoder Dec decodes features back into observations. Colors indicate tied weights.

features, referred to as z_1 and z_2, a translator $z_3 = T(z_1, z_2)$ that translates the features z_1 into features for the context of z_2, which are denoted z_3, and finally a target context decoder $\text{Dec}(z_3)$, which decodes these features into $\hat{o}_t^j$. We will use $F(o_t^i, o_0^j) = z_3$ to denote the feature extractor that generates the features z_3 from an input observation and a context image. The encoders Enc_1 and Enc_2 can have either different weights or tied weights depending on the diversity of the demonstration scenes. To deal with the complexities of pixel-level reconstruction, we include skip connections from Enc_2 to Dec. The model is supervised with a squared error loss $\mathcal{L}_{\text{trans}} = \|(\hat{o}_t^j)_{\text{trans}} - o_t^j\|_2^2$ on the output o_t^j and trained end-to-end.

However, we need the features z_3 to carry useful information, in order to provide an informative distance metric between demonstrations for feature tracking. To ensure that the translated features z_3 form a representation that is internally consistent with the encoded image features z_1, we jointly train the translation model encoder Enc_1 and decoder Dec as an autoencoder, with a reconstruction loss $\mathcal{L}_{\text{rec}} = \|\text{Dec}(\text{Enc}_1(o_t^j)) - o_t^j\|_2^2$. We simultaneously regularize the feature representation of this autoencoder to align it with the features z_3, using the loss $\mathcal{L}_{\text{align}} = \|z_3 - \text{Enc}_1(o_t^j)\|_2^2$. This forces the encoder Enc_1 and decoder Dec to adopt a consistent feature representation, so that the observation from the target context o_t^j is encoded into features that are similar to the translated features z_3. The training objective for the entire model is then given by the combined loss function $\mathcal{L} = \sum_{(i,j)} (\mathcal{L}_{\text{trans}} + \lambda_1 \mathcal{L}_{\text{rec}} + \lambda_2 \mathcal{L}_{\text{align}})$, with D_i and D_j being a pair of expert demonstrations chosen randomly from the training set, and λ_1 and λ_2 being hyperparameters. If we don't regularize the encoded features of learning trajectories and translated features of experts to lie in the same feature space, the reward function described in Section V-A is not effective since we are tracking features which have no reason to be in the same space. Examples of translated demonstrations are shown in Section VI and the project website.

V. Learning Policies via Context Translation

The model described in the previous section can translate observations and features from the demonstration context into the learner's context ω_l. However, in order for the learning agent to actually perform the demonstrated behavior, it must be able to acquire the actions that track the translated features. We can choose between a number of deep reinforcement learning algorithms to learn to output actions that track the translated demonstrations given the reward function we describe below.

A. Reward Functions for Feature Tracking

The first component of the feature tracking reward function is a penalty for deviations from the translated features. At each time step, the translation function F (which gives us z_3) can be used to translate each of the demonstration observations o_t^i into the learner's context ω_l. The reward function then corresponds to minimizing the squared Euclidean distance between the encoding of the current observation to all of these translated demonstration features, which is approximately tracking their average, resulting in

$$\hat{R}_{\text{feat}}(o_t^l) = -\|\text{Enc}_1(o_t^l) - \frac{1}{n}\sum_i^n F(o_t^i, o_0^l)\|_2^2,$$

where $\text{Enc}_1(o_t^l)$ computes the features of the learner's observation at time step t, given by o_t^l, and $F(o_t^i, o_0^j)$ computes translated features of experts.

Unfortunately, feature tracking by itself may be insufficient to successfully imitate complex behaviors. The reason for this is that the distribution of observations fed into Enc_1 during policy learning may not match the distribution from the demonstrations that are seen during training: although a successful policy will closely track the translated policy, a poor initial policy might produce observations that are very different. In these cases, the encoder Enc_1 must contend with out-of-distribution samples, which may not be encoded properly. In fact, the model will be biased toward predicting features that are closer to the expert, since it only saw expert data during training. To address this, we also introduce a weak image tracking reward. This reward directly penalizes the policy for experiencing observations that differ from the translated observations, using the full observation translation model M:

$$\hat{R}_{\text{img}}(o_t^l) = -\|o_t^l - \frac{1}{n}\sum_i^n M(o_t^i, o_0^l)\|_2^2$$

The final reward is then the weighted combination $\hat{R}(o_t^l) = \hat{R}_{\text{feat}}(o_t^l) + w_{\text{rec}}\hat{R}_{\text{img}}(o_t^l)$, where w_{rec} is a small constant (tuned as a hyperparameter in our implementation).

B. Reinforcement Learning Algorithms for Feature Tracking

With the reward described in Section V-A, we perform reinforcement learning in order to learn control policies in our learning environment. Our method can be used with any

reinforcement learning algorithm. We use trust region policy optimization (TRPO) [36] for our simulated experiments but not for real world experiments because of it's high sample complexity. For the real-world robotic experiments, we use the trajectory-centric RL method used for local policy optimization in guided policy search (GPS) [3], which is based on fitting locally linear dynamics and performing LQR-based updates. We compute image features z_3, and include these as part of the state. The cost function for GPS is then a squared Euclidean distance in state space, and we omit the image tracking cost described in Section V-A. For simulated striking and real robot pushing, this cost function is also weighted by a quadratic ramp function weighting squared Euclidean distances at later time steps higher than initial ones.

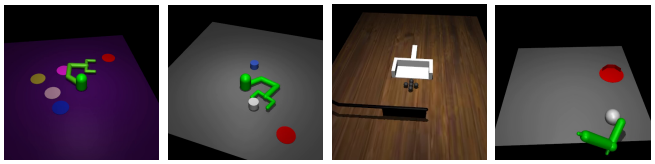

Fig. 3: Four simulated tasks, from left to right: reaching (goal is to reach the red circle), pushing (goal is to push the white can to the red goal), sweeping (goal is to sweep grey balls into the pan), and striking (goal is to strike the white ball to the red goal).

VI. Experiments

Our experiments aim to evaluate whether our context translation model can enable imitation-from-observation, and how well representative prior methods perform on this type of imitation learning task. The specific questions that we aim to answer are: (1) Can our context translation model handle raw image observations, changes in viewpoint, and changes in the appearance and positions of objects between contexts? (2) How well do prior imitation learning methods perform in the presence of such variation, in comparison to our approach? (3) How well does our method perform on real-world images, and can it enable a real-world robotic system to learn manipulation skills? All results, including illustrative videos, video translations and further experiment details can be found on: `https://sites.google.com/site/imitationfromobservation/`

In order to provide detailed comparisons with alternative prior methods for imitation learning, we set up four simulated manipulation tasks using the MuJoCo simulator [37]. To provide expert demonstrations, we hand-specified a reward function for each task and used a prior policy optimization algorithm [36] to train an expert policy. We collected video demonstrations of rollouts from the final expert policy acting in a number of randomly generated contexts.

The tasks are illustrated in Fig. 3. The first task requires a robotic arm to reach varying goal positions indicated by a red disk, in the presence of variation in color and appearance. The second task requires pushing a white cylinder onto a red coaster, both with varying position, in the presence of varied distractor objects. The third task requires the simulated robot to sweep five grey balls into a dustpan, under variation in viewpoint. The fourth task involves using a 4 DoF arm to strike a white ball toward a red target which varies in position. The project website illustrates the variability in appearance and object positioning in the tasks, and also presents example translations of individual demonstration sequences.

A. Network Architecture and Training

For encoders Enc_1 and Enc_2 in simulation we perform four 5×5 stride-2 convolutions with filter sizes 64, 128, 256, and 512 followed by two fully-connected layers of size 1024. We use LeakyReLU activations with leak 0.2 for all layers. The translation module $T(z_1, z_2)$ consists of one hidden layer of size 1024 with input as the concatenation of z_1 and z_2. For the decoder Dec in simulation we use a fully connected layer from the input to four fractionally-strided convolutions with filter sizes 256, 128, 64, 3 and stride $\frac{1}{2}$. We have skip connections from every layer in the context encoder Enc_2 to its corresponding layer in the decoder Dec by concatenation along the filter dimension. For real world images, the encoders perform 4 convolutions with filter sizes 32, 16, 16, 8 and strides 1, 2, 1, 2 respectively. All fully connected layers and feature layers are size 100 instead of 1024. The decoder uses fractionally-strided convolutions with filter sizes 16, 16, 32, 3 with strides $\frac{1}{2}$, 1, $\frac{1}{2}$, 1 respectively. For the real world model only, we apply dropout for every fully connected layer with probability 0.5, and we tie the weights of Enc_1 and Enc_2.

We train using the ADAM optimizer with learning rate 10^{-4}. We train using 3000 videos for reach, 4500 videos for simulated push, 894 videos for sweep, 3500 videos for strike, 135 videos for real push, 85 videos for real sweep with paper, 100 videos for real sweep with almonds, and 60 videos for ladling almonds. We downsample videos to 36×64 pixels for simulated sweeping and 48×48 for all other videos.

B. Comparative Evaluation of Context Translation

Results for the comparative evaluation of our approach are presented in Fig 4. Performance is evaluated in terms of the final distance of the target object to the goal during testing. In the reaching task, this is the distance of the robot's hand from the goal, in the pushing task, this is the distance of the cylinder from the goal, in the sweeping task, this corresponds to the mean distance of the balls from the inside of the dustpan, and in the striking task this is the final distance of the ball from the goal position. All distances are normalized by dividing by the initial distance at the start of the task, and success is measured as a thresholding of the normalized distance. We evaluate each task on 10 randomly generated environment conditions, each time performing 100 iterations of reinforcement learning with 12,500 samples per iteration.

Our comparisons include our method with TRPO for policy learning, an oracle that trains a policy with TRPO on the ground truth reward function in simulation, which represents an upper bound on performance, and three prior imitation learning methods. The first prior method learns a reward using pre-trained visual features, similar to the work of [9]. In this method, features from an Inception-v3

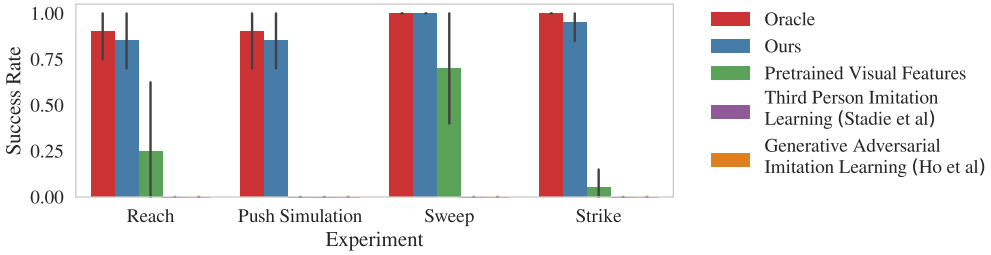

Fig. 4: Comparisons with prior methods on the reaching, pushing, sweeping, and striking tasks. The results show that our method successfully learned each task, while the prior methods struggled to perform the reaching, pushing and striking tasks, and only the pretrained visual features approach was able to make a reasonable improvement on the sweeping task. Third person imitation learning [7] and generative adversarial imitation [8] learning are both at 0% success rate on the graph.

network trained on ImageNet classification [38] are used to encode the goal image from the demonstration, and the reward function corresponds to the distance to these features averaged over all the training demonstrations. We experimented with several different feature layers from the Inception-v3 network and chose the one that performed best. The second prior method, third person imitation learning (TPIL), is an IRL algorithm that explicitly compensates for domain shift using an adversarial loss [7], and the third is an adversarial IRL-like algorithm called generative adversarial imitation learning (GAIL) [8], using a convolutional model to process images as suggested by [39]. Among these, only TPIL explicitly addresses changes in domain or context.

The results, shown in Fig 4, indicate that our method was able to successfully learn each of the tasks when the demonstrations were provided from random contexts. Notably, none of the prior methods were actually successful on the reaching, pushing or striking tasks, and struggled to perform the sweeping task. This indicates that imitation-from-observation in the presence of context differences is an exceedingly challenging problem.

C. Ablation Study

To evaluate the importance of different loss functions while training our translation model, and the different components for the reward function while performing imitation, we performed ablations by removing these components one by one during model training or policy learning. To understand the importance of the translation cost, we remove cost $\mathcal{L}_{\text{trans}}$, to understand whether features z_3 need to be properly aligned we remove model losses $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{align}}$. In Fig 6 we see that

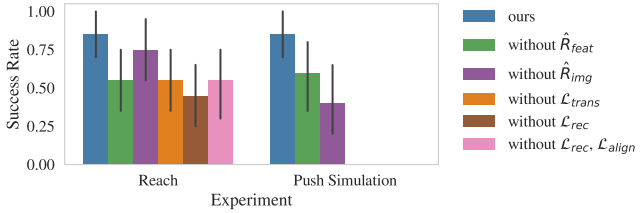

Fig. 6: Ablations on model losses and reward functions for the simulated reaching and pushing tasks.Our method with all components does consistently the best across tasks. Note: for the Push Simulation task, we did not perform ablations "without $\mathcal{L}_{trans}$", "without $\mathcal{L}_{rec}$", and "without $\mathcal{L}_{rec}, \mathcal{L}_{align}$"

the removal of each of these losses significantly hurts the performance of subsequent imitation. On removing the feature tracking loss $\hat{R}_{\text{feat}}$ or the image tracking loss $\hat{R}_{\text{image}}$ we see that overall performance across tasks is worse.

D. Natural Images and Real-World Robotic Manipulation

To evaluate whether our method is able to scale to real-world images and robots, we focus on manipulation tasks involving tool use, where object positions and camera viewpoints differ between contexts. All demonstrations were provided by a human, while the learned skills were performed by a robot. Since our method assumes that the contexts of the demonstrations and the learner are sampled from the same distribution, the human and the robot both used the same tool to perform the tasks, avoiding any systematic domain shift that might result from differences in the appearance of the human or robot arm. To this end, we apply a cropping of each video around task-relevant areas of each demonstration.

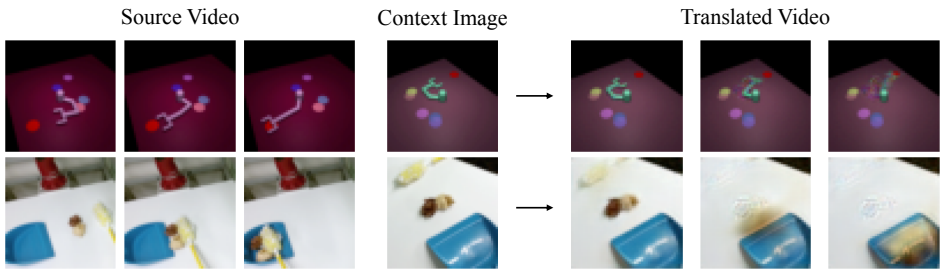

Fig. 5: Translations from a video in the holdout set to a new context for the reaching task (top) and paper sweeping task (bottom).

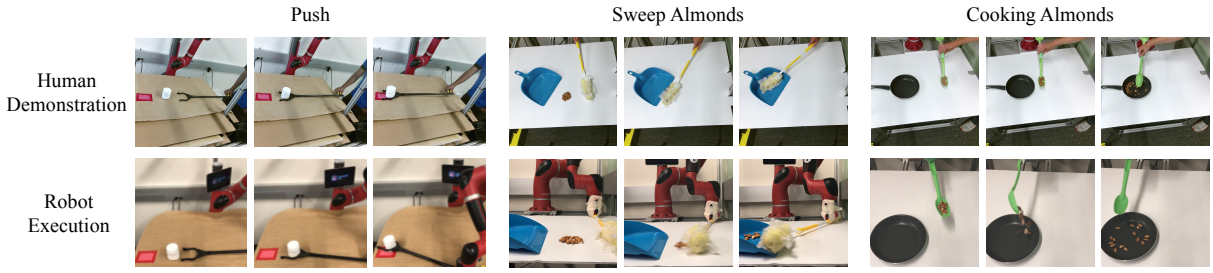

Fig. 7: Top Row: Demonstrations by a human demonstrator showing the robot how to perform the pushing, sweeping and ladling almonds task in the real world. Bottom Row: Execution of the robot successfully performing the pushing, sweeping and ladling almonds tasks.

Investigating domain shift is left for future work, and could be done, for example, using domain adaptation [40]. In the present experiments, we assume that the demonstration and test contexts come from the same distribution, which is a reasonable assumption in settings such as tool use and navigation, or tasks where the focus is on the objects in the scene rather than the arm or end-effector.

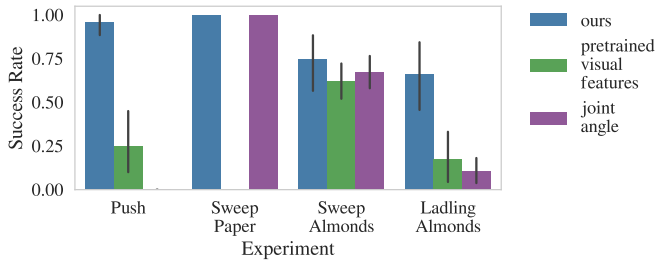

Fig. 8: Plot depicting success rate for our method versus other baselines on the real world tasks with the Sawyer robot. Success metrics differ per task as described in Section VI-D. As is seen clearly, our method consistently performs well on all the real world tasks, and outperforms the baseline methods.

1) Pushing: In the first task, the goal is to push a cylinder to a marked goal position. The success metric is defined as whether the final distance between the cylinder and goal is within a predefined threshold. We evaluate our method in the setting where real-world demonstrations are provided by a human and imitation is done by a robot in the real world.

We evaluated how our method can be used to learn a pushing behavior with a real-world robotic manipulator, using a 7-DoF Sawyer robot. Since the TRPO algorithm is too data intensive to learn on real-world physical systems, we use GPS for policy learning (Section V-B).

For comparison, we also test GPS with a reward that involves tracking pre-trained visual features from the Inception-v3 network (Section VI-B), as well as a baseline reward function that attempts to reach a fixed set of joint angles, specified through kinesthetic demonstration. Note that our method itself does not use any kinesthetic demonstrations, only video demonstrations provided by the human. In order to include the high-dimensional visual features in the state for guided policy search, we apply PCA to reduce their dimensionality from 221952 to 100, while our method uses all 100 dimensions of z_3 as part of the state. We found that our method could successfully learn the skill using demonstrations from different viewpoints, and outperforms the pre-trained features and kinesthetic baseline, as shown in Fig 8.

2) Sweeping: The pushing task illustrates the basic capability of our method to learn skills involving manipulation of rigid objects. However, one major advantage of learning visual reward functions from demonstration is the ability to acquire representations that can be used to manipulate scenes that are harder to represent analytically, such as granular media. In this next experiment, we study how well our method can learn two variants of a sweeping task: in the first, the robot must sweep crumpled paper into a dustpan, and in the second it must sweep a pile of almonds. We used almonds in place of dirt or fluids to avoid damaging the robot. Quantitative results are summarized in Fig 8.

On the easier crumpled paper task, both our method and the kinesthetic teaching approach works well, but the reward that uses pre-trained visual features is insufficient to accomplish the task. On the almond sweeping task (Fig 7), our method achieves a higher success rate than the alternative approaches. The success metric is defined as the average percentage of almonds or paper pieces that end up inside the dustpan.

3) Ladling Almonds: Our last task combines granular media (almonds) and a more dynamic behavior. In this task, the robot must ladle almonds into a cooking pan (Fig 7). This requires keeping the ladle upright until over the pan, and then dumping them into the pan by turning the wrist. The success metric is the average fraction of almonds that were ladled into the pan. Learning from only raw videos of the task being performed by a human in different contexts, our method achieved a success rate of 66% , while the alternative approaches generally could not perform this task. An insight into why the joint angles approach wouldn't work on this task is that the spoon has to remain upright until just the right position over the pan after which it should rotate and pour into the pan. The joint angle baseline can simply interpolate between the final turned spoon position and the initial position and pour the almonds in the wrong location. Quantitative results and comparisons are summarized in Fig 8.

VII. Discussion and Future Work

We investigated how imitation-from-observation can be performed by learning to translate demonstration observation sequences between different contexts, such as differences in viewpoint. After translating observations into a target context, we can track these observations with RL, allowing the learner

to reproduce the observed behavior. The translation model is trained by translating between the different contexts observed in the training set, and generalizes to the unseen context of the learner. Our experiments show that our method can be used to perform a variety of manipulation skills, and can be used for real-world robotic control on a diverse range of tasks patterned after common household chores.

Although our method performs well on real-world tasks and several tasks in simulation, it has a number of limitations. First, it requires a substantial number of demonstrations to learn the translation model. Training an end-to-end model from scratch for each task may be inefficient in practice, and combining our method with higher level representations proposed in prior work would likely lead to more efficient training [9], [23]. Second, we require observations of demonstrations from multiple contexts in order to learn to translate between them. In practice, the number of available contexts may be scarce. Future work would explore how multiple tasks can be combined into a single model, where different tasks might come from different contexts. Finally, it would be exciting to explore explicit handling of domain shift in future work, so as to handle large differences in embodiment and learn skills directly from videos of human demonstrators obtained, for example, from the Internet.

REFERENCES

[1] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive UAV control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation*, 2013.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, 2016.

[4] D. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *NIPS*, 1988.

[5] A. Edwards, C. Isbell, and A. Takanishi, "Perceptual reward functions," *arXiv preprint arXiv: 1608.03824*, 2016.

[6] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00, 2000.

[7] B. C. Stadie, P. Abbeel, and I. Sutskever, "Third-person imitation learning," in *Proceedings of the International Conference on Learning Representations, ICLR 2017*.

[8] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems 29*, 2016.

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

[10] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[11] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[12] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *I. J. Robotics Res.*, vol. 29, no. 13, pp. 1608–1639, 2010.

[13] J. Kober and J. Peters, "Learning motor primitives for robotics," in *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*. IEEE, 2009, pp. 2112–2118. [Online]. Available: https://doi.org/10.1109/ROBOT.2009.5152577

[14] A. Billard and M. J. Mataric, "Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture," *Robotics and Autonomous Systems*, vol. 37, no. 2-3, pp. 145–160, 2001.

[15] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. ACM, 2004.

[16] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems 24*, 2011.

[17] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*. AAAI Press, 2008.

[18] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich, "Maximum margin planning," in *Machine Learning, Proceedings of the Twenty-Third International Conference ICML*, 2006.

[19] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[20] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proceedings of the 33nd International Conference on Machine Learning, ICML*, 2016.

[21] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning objective functions for manipulation," in *2013 IEEE International Conference on Robotics and Automation*, 2013.

[22] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS*, 2011.

[23] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," *arXiv preprint arXiv:1704.06888*, 2017.

[24] J. Lee and M. S. Ryoo, "Learning robot activities from first-person human videos using convolutional future regression," *arXiv preprint arXiv:1703.01040*, 2017.

[25] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *NIPS*, 2017.

[26] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[27] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *CVPR*, 2017.

[28] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[29] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[30] Z. Xu and M. Cakmak, "Enhanced robotic cleaning with a low-cost tool attachment," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014.

[31] C. Schenck and D. Fox, "Visual closed-loop control for pouring liquids," in *2017 IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2017.

[32] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control," in *Robotics: Science and Systems XI*, 2015.

[33] J. Sung, S. H. Jin, I. Lenz, and A. Saxena, "Robobarista: Learning to manipulate novel objects via deep multimodal embedding," *In International Symposium on Robotics Research (ISRR), 2015*.

[34] A. Byravan and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks," in *ICRA*, 2017.

[35] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2017.

[36] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

[37] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

[39] Y. Li, J. Song, and S. Ermon, "Inferring the latent structure of human decision-making from raw visual inputs," *arXiv preprint arXiv:1703.08840*, 2017.

[40] E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell, "Adapting deep visuomotor representations with weak pairwise constraints," in *Workshop on Algorithmic Robotics*, 2016.