Human-to-Robot Imitation in the Wild

Shikhar Bahl Abhinav Gupta* Deepak Pathak*
Carnegie Mellon University

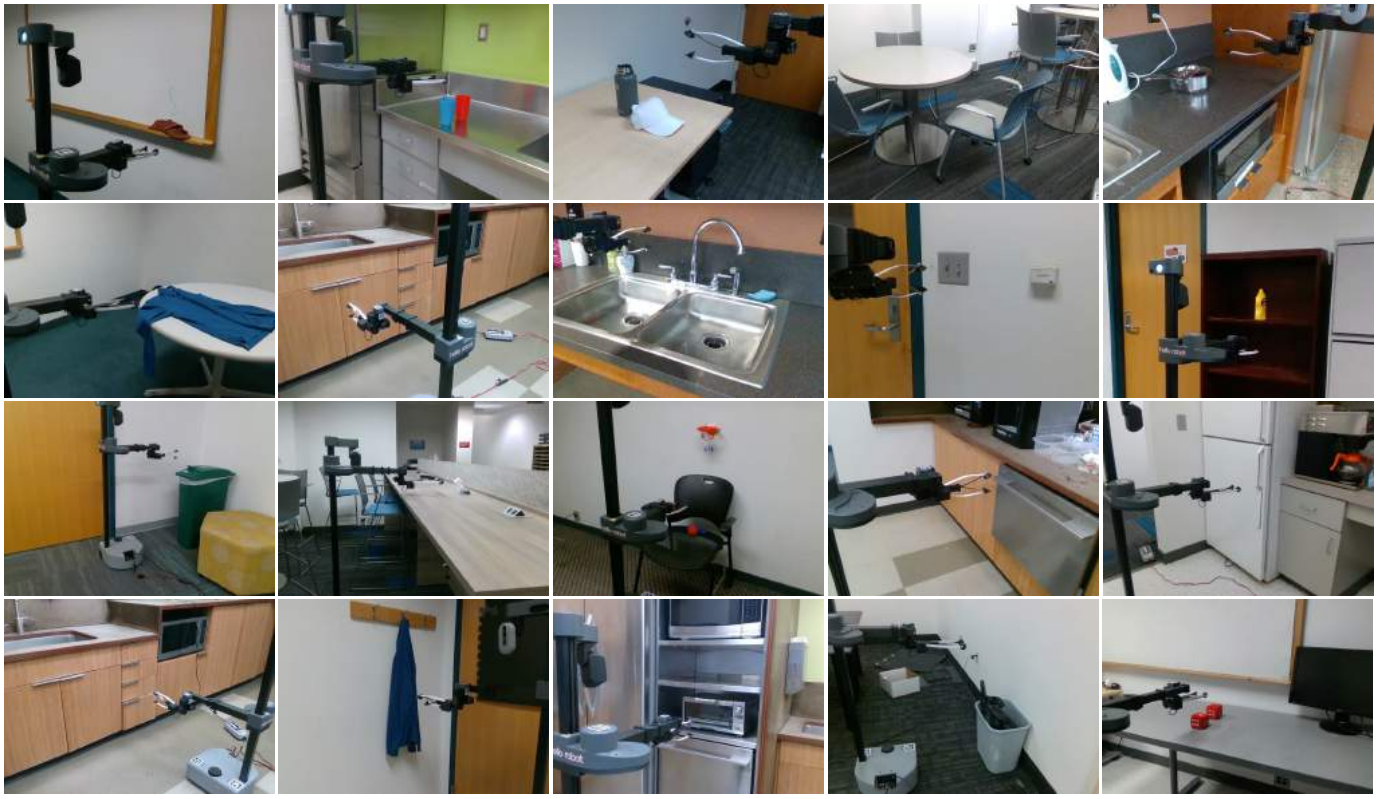


Fig. 1: We present WHIRL, an efficient real-world algorithm for one-shot visual imitation in the wild. WHIRL is able to directly learn from unstructured human videos and generalize to new tasks as well. Videos and webpage at: https://human2robot.github.io

***Abstract*—We approach the problem of learning by watching humans in the wild. While traditional approaches in Imitation and Reinforcement Learning are promising for learning in the real world, they are either sample inefficient or are constrained to lab settings. Meanwhile, there has been a lot of success in processing passive, unstructured human data. We propose tackling this problem via an efficient one-shot robot learning algorithm, centered around learning from a third person perspective. We call our method WHIRL: In-the-Wild Human Imitating Robot Learning. WHIRL extracts a *prior* over the intent of the human demonstrator, using it to initialize our agent's policy. We introduce an efficient real-world policy learning scheme that improves using interactions. Our key contributions are a simple sampling-based policy optimization approach, a novel objective function for aligning human and robot videos as well as an exploration method to boost sample efficiency. We show one-shot generalization and success in real world settings, including 20 different manipulation tasks in the wild. Videos at https://human2robot.github.io.**

I. Introduction

In recent years, there has been significant advances in robot manipulation: from grasping to pushing and pick/place tasks [2, 36, 23]; from manipulating a rubik's cube [1] to opening cabinet doors or makeshift doors [60, 49]. While there has been substantial progress, most experiments in this area have still been restricted to simulation [37, 4, 71] or table-top experiments in the lab [31, 11]. We ask a basic question as to why hasn't this progress transferred to manipulation in the real-world setup and why do we still see most experiments in lab setups or simulations? Although there have been efforts to perform grasping in home setups [19, 69], general manipulation is still studied in either simulation or lab-like settings.This work delves the question of how we could move from from lab experiments to more in-the-wild setups.

We believe the biggest bottleneck for learning manipulation *in the wild* is the lack of scalable and safe frameworks. Traditionally, designing a controller or policy for manipulation tasks requires learning via reinforcement (RL), which can be data-hungry and unsafe especially in the real world. While RL has had success in simulated tasks, real world tasks do not have structured rewards, thus making the problem that of sparse search. A popular alternative is to use imitation learning (IL) based approaches, but common IL approaches rely on lots of kinesthetic or teleoperated demonstrations per task. However, this data can be expensive to obtain in the real world and may

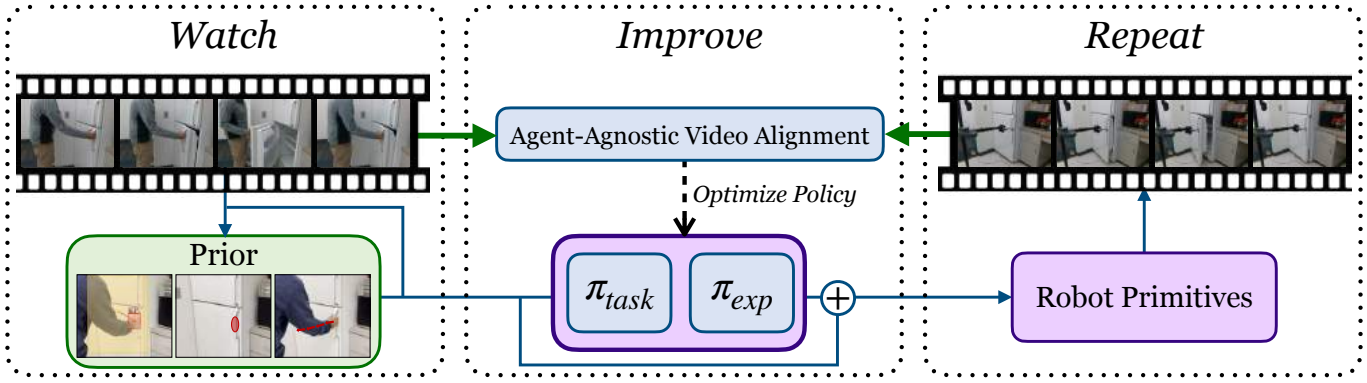


Fig. 2: Our method (WHIRL) provides an efficient way to learn from human videos. We have three core components: we first "watch" and obtain human priors such as hand movement and object interactions. We "repeat" these priors by interacting in the real world, by both trying to achieve task success and explore around the prior. We "improve" our task policy by leveraging our agent-agnostic objective function which aligns human and robot videos.

not be generalizable to new settings or different robots. There have been attempts at one shot imitation at inference, but these methods requires thousands of demonstrations or interactions during training [12, 16, 46].

To move towards general robot manipulation out of the lab and tabletop settings, we believe visually imitating humans provide a safe and scalable alternative. Rather than asking humans to teleoperate, robot should observe humans to learn as they interact in the world. Humans provide a rich source of data as they often act in interesting and near optimal ways given a task or an environment. In the visual imitation framework, the agent observes other agents perform action without access to actions (just the pixels). This data is then used to guide their own exploration and learning. However, there are several challenges in making visual imitation learning work: first, there is the issue of embodiment mismatch (robots and humans are different agents and have different bodies). Second, there is no access to actions from humans, which have to be inferred. Third, there is no access to task information including rewards beyond pixels. Current approaches use end-to-end learning [65, 78, 67] which requires a lot of samples during training and are hence restricted to lab/simulation settings.

In this paper, we propose to revive this visual human-imitation framework to move robot manipulation out of the lab and into the wild. Instead of learning end-to-end from scratch, we propose leveraging advances in computer vision and computational photography to (a) infer a trajectory and interaction information from the human, thus obtaining a prior; (b) learning an improvement policy via interactions in the real world; and (c) bridging the embodiment gap between human demonstrations and robot videos. But above all, we only use imitation data as **priors** for our policy. Naively using priors will not result in success due to a host of issues, e.g. varying morphologies or inaccuracies in detections. It is crucial to *interact* with the real world to learn generalizable manipulation. We introduce a sampling based optimization framework, similar to the Cross Entropy Method (CEM), in order to iteratively improve the interaction policy. To make WHIRL operate without supervision, we introduce an *agent-agnostic* alignment objective function for the described optimization approach. In order to not be too restricted by the prior, we employ a novel task-agnostic exploration policy which allows the agent to sample new and interesting actions. This all leads to an efficient framework for manipulation tasks in real world.

We demonstrate our framework on 20 different tasks in 3 different environments. We show one-shot, in the wild generalization and success in various real world settings, including manipulation tasks such as opening and closing doors or fridges, putting objects in shelves, folding shirts, cleaning white boards, opening taps and a variety of other tasks. We analyze our approach thoroughly in terms of task success, generalization, and performance compared to state-of-the-art baselines. To the best of our knowledge, this is the first effort that takes robot manipulation out of the lab and into the real world at this scale.

II. Related Work

A. Detecting Humans

The field of computer vision has studied the problem of detecting humans in a wide variety of approaches. Most such applications are contained in the domain of graphics, but many have applications in real world robotics as well. There are many possible uses of such, for example modeling human bodies, detecting poses, inferring dynamics or understanding interactions between humans and the world. From a modeling perspective, works such as MANO [53] and SMPL [35] have proposed analytical models of human hands and bodies respectively. Hand and body pose estimation [72, 24, 54] can be useful in the context of robotics, as it can allow for spatially grounding a demonstration, which is something that we leverage in our approach. Estimation and detection of humans, while useful, does not help in understanding *what* the human is doing. For this, large annotated video datasets can help detect and infer human actions, such as the Something-Something [18], YouCook [10], ActivityNet datasets [14] or the 100 Days of Hands [62] (100DOH) dataset. 100DOH [62] is particularly useful as it contains object level interaction annotations. WHIRL aims to remain as general as possible in

terms of the human prior used, using only object interaction data. We employ both the models from Rong et al. [54] and the hand-object detector from Shan et al. [62] for estimating hand position and interaction information. It is possible to combine WHIRL with stronger priors for human hands, for example, building a knowledge graph of objects and functional grasps [39] or using heat sensing [3] to understand interactions.

B. Imitation and Reinforcement Learning from Videos

Learning From Human Videos A large field of robot learning (Learning from Demonstrations: LfD) is focused on learning from expert demonstrators [44, 50, 48, 55]. However, most of the work in this area tackles the problem of learning from demonstrations that humans provide directly to the robot via kinesthetic teaching or teleoperation. This is an expensive way to gather data for teaching robots. On the other hands, videos of humans perfoming daily activities are widely available on the internet and can provide good semantic supervision for robotics tasks. However, extracting the right knowledge, for example aligning human videos with robot videos, is challenging. One solution is to learn a direct correspondence. The use of paired human and robot data [65, 64, 33] is a common approach in this line of work. For example, Sharma et al. [65] aim to learn to produce subgoals in the robot's perspective, conditioned on a human video. Liu et al. [33] seeks to learn a translation model based on the paired demonstrations directly. Collecting paired demonstrations is challenging, and only a limited amount of data can be collected. Thus, previous work [67, 79] has employed cycle-consistency [82, 13] to learn an unsupervised pairing. Similarly, Sermanet et al. [59] uses a contrastive loss between frames close to and far away from the anchor point in the video, in order to obtain a representation. Sermanet et al. [60] trains a classifier using human demonstrations, which is then used to build a reward function. Unsupervised methods can learn a translation model for single tasks, however they have to be trained in every new setting, which is time consuming. Most such approaches require many random interactions to learn representations, and this process often yields unstable models [67]. WHIRL, on the other hand, does not need any random data to learn representations, and can work with even a single demonstration, in a variety of in-the-wild settings.

Offline Videos and Datasets Instead of using human videos, recent approaches have attempted to employ a reacher-grabber tool as the demonstration collection device [69, 77, 43]. These approaches have the advantage of having a smaller domain gap between robot and human actions, since the videos are in first person view. However, such a setup limits the number of tasks that are achievable, and adds considerable effort in collecting the data, since the approaches are not able to use large-scale human datasets, for example Youtube videos. On the other hand, advances in many computer vision tasks such as action recognition [80, 18, 20, 5, 73, 15], video understanding [15, 32, 34] or self-supervised representation learning [21, 42, 8, 41] have leveraged videos collected offline. These video datasets include the Something-Something [18], Epic Kitchens [9] or ActivityNet datasets [14]. These can provide important semantic information as well a high amount of visual and task diversity, which can aid in generalization. Similarly, works such as Chen et al. [6] and Shao et al. [63] find that using a large-scale human datasets, augmented with a few demonstrations from the robot as well as task labels, can help learn a semantic action classifier which generalizes to new tasks. Unlike these approaches, we do not use any task labels or robot specific fine-tuning for the feedback module. Embedding task specific knowledge into reward classifiers does not scale to in-the-wild settings, contrary to our approach.

(a) Drawer (b) Dishwasher (c) Door

Fig. 3: We perform various experiments in the wild. We select a subsample of tasks, as shown above, to perform a thorough study of our WHIRL as well as baselines and ablations. These tasks are: drawer, door and dishwasher opening and closing.

Learning Action Policies from Priors While learning reward functions and representations from offline videos can be useful in robotics, videos of humans contain stronger priors. Learning keypoints [7, 27, 75] or object-level [47, 58] from videos, and using these as input to a control policy has been shown to be useful for certain tasks, but requires knowledge of the task and careful design, for example knowing the number of objects or keypoints. This can be a limiting factor when trying to scale to a general robot setup. Previous approaches have also used hand [29, 40] and object tracking [76] to learn action policies, however, these have been limited to simple settings and require very structured planning algorithms that are task specific. Our approach on the other hand is flexible and works for almost any manipulation task. Previous approaches do not perform any iterative improvement, contrary to WHIRL.

III. Human-to-Robot Visual Imitation In the Wild

We address the challenge of learning from humans by extracting priors from observing their actions, leveraging the priors to learn an interaction policy in the real world, and exploring around the prior in an efficient manner. We build a general robot learning algorithm that can work in many in-the-wild settings. We call this approach WHIRL: In-the-Wild Human Imitating Robot Learning. In this section, we describe how WHIRL works.

A. Human Priors

1) Extracting Human Priors

Most trajectories (τ) of interest for manipulation tasks can be broken down into smaller sub-trajectories: $\tau_{\text{pre-interaction}}$, $\tau_{\text{interaction}}$ and $\tau_{\text{post-interaction}}$. Throughout the paper, we refer to these as primitives. A more complex task can be thought

of as a composition of such primitives. Once we are able to use human videos to estimate these primitives, we can try to deploy these on a robot, despite any differences in morphology. Videos of the desired task (V), such as door opening, are used to obtain this trajectory parameterization. The key components of a video of a human performing a task include how the target is moving as well as where and when the interactions happen. We describe how we infer this information from third person videos below.

Extracting Hand Information We process each individual frame V_t of the video (V) at timestep t to obtain an estimate of the position of the hand: x_t, y_t, z_t. We obtain this pose using the 100DOH detection model [62], built on top of Faster-RCNN [51] and trained to output hand bounding box (b_t). This is a continuous vector of coordinates in image space. The hand position (in the camera frame) is referred to as h_t. In order for the robot to grasp and interact with an object, the orientation of the wrist and the force applied on the gripper are important as well. We use the MANO [53] parameterization of hands in order to obtain these. Specifically, we use the part of the parameterization that describes the rotation of the wrist, $\theta_{\text{hand}}^{(t)}$.

Extracting Interaction Information Inferring the position of the hand can give useful information, but we also need to understand when the hand interacts with an object. Detecting contact is important in determining $\tau_{pre-interaction}$: it determines where the interaction occurs. Thus, we employ the 100DOH [62] model to detect when this interaction occurs. We use this information and previously computed hand poses to extract waypoints for the robot. Specifically, we use the 100DOH model to obtain a discrete valued contact variable: c_t. This represents a possible contact that might be occurring at frame t of the video. The possible options are: no contact, contact with portable or fixed object, and self contact. However, since out-of-the-box detections in unstructured settings can be noisy, we employ the Savitzky–Golay [57] filter for smoothing c_t across timesteps. Using smoothed detection $\hat{c}_t$ we determine the time-step where the interaction started in the video: $t_{\text{interaction}}$ and when it ended: t_{end}. We denote the hand position at these timesteps as $h_{\text{interaction}}$ and h_{end}. In order to not overfit to the detections, we in fact sample from a distribution centered around the start and end points. We also sample intermediate trajectory waypoints, h_{mid}. We additionally use a simple binary representation of a grasp, determined from the contact variable $\hat{c}_t$.

Overall, our extracted prior from a video demonstration from a human can be described as a set of interaction waypoints: $h_{\text{interaction}}$, h_{mid}, and h_{end}, a grasp or interaction orientation measure θ_{hand}, and commands to close or open the hand: $o_{1:T}$ (where T is the length of the video). Figure 4 shows the different parts of the human prior we use. Note that some tasks may require a more densely sampled set of waypoints. For simplicity we think of h_{mid} as a single point, but it can be also a set of midpoints in the hand trajectory.

(a) Detection (b) Interaction (c) Direction

Fig. 4: We show the different components of the human prior. First we extract the position of the hand and possible object interactions (a). This indicates a possible area of interaction (b) and direction for moving the robot hand (c). We project these to the robot's action space and execute the trajectory.

2) Converting Human Priors to Robot Priors

Once we obtain the desired trajectories from human videos, we can convert them into the robots frame and obtain desired poses, using depth image (d_t) from the external camera. Our setup uses a depth image, but is compatible with any 3D pose estimation approach. Given a video V_k of length T, we then project the obtained priors, $h_{\text{interaction}}$, h_{mid}, h_{end}, θ_{hand}, $o_{1:T}$ to the robot's frame via 3D pose estimation from depth data. For both the gripper open and wrist orientation parameters, we use a robot-specific heuristic function, as every robot's coordinate axis is different. Let the detected waypoints be $\boldsymbol{h} = h_{\text{interaction}}, h_{\text{mid}}, h_{\text{end}}$. This process can be described as:

$$f_{\text{map}}(\boldsymbol{h}, \theta_{\text{hand}}, o_{1:T}) = w_{\text{interaction}}, w_{\text{mid}}, w_{\text{end}}, \theta_{\text{YPR}}, g_{1:T} \triangleq \Psi_k$$

where $\boldsymbol{w}$ are waypoints in the robot's frame, θ_{YPR} is a wrist rotation (yaw-pitch-roll format) in the robots frame, and $g_{1:T}$ are robot gripper open/close continuous parameters. We refer to this vector by Ψ_k.

B. Policy Learning via Interaction

Human priors from videos can give a rough guideline on how to perform the task. They are useful because they can be distilled into a neural network policy, which can possibly generalize beyond the training data. However, directly executing the prior on the task will not generally lead to success, due to differences in morphologies between human and robot hands, inaccuracies in detections, or errors in the calibration process. Thus, we need to learn a policy via real world *interaction* in order to succeed at this task. Such a learning procedure must have 3 important properties:

- The real world interactions must be safe.
- While safe, the interactions must not be too restrictive.
- This process must be sample efficient.

The safety of the interactions can be ensured by the human prior. Following the prior, even one that has errors, will lead to somewhat reasonable behavior, and is very likely to be safe. However, being too close to the prior will restrict the reach of the policy, and thus it will be unable able to solve the task. In order to address this challenge, we employ a *task policy* which aims to solve the task and a task agnostic *exploration policy* that explores around the human prior so that we do not fall into a local minimum. We describe the objective functions of these policies in the following sections. Finally, in order to

ensure the learning process is sample efficient, we introduce a simple and easy to use zeroth order real world optimization procedure (similar to CEM). Since our goal is to efficiently perform many manipulation tasks in the wild, traditional RL methods are infeasible. A summary is in Algorithm 1.

Algorithm 1 Training Procedure for WHIRL

Require: Task videos: $V_{1:K}$, f_{map}: video to robot actions function, prior task and exploration policies: π, π_{exp}. Video-level (Φ) and frame-level (Φ_f) agent agnostic representation. M real world interactions per task.

```
while not converged do
  for k = 1...K do
    Ψ_k = f_map(V_k)
    for m = 1...M do
      Sample ΔΨ_{k,m} = π_exp(V_k, Ψ_k) (prob: p)
      Sample ΔΨ_{k,m} = π(V_k, Ψ_k) (prob: 1 − p)
      a_{j,m} = Ψ_k + ΔΨ_{k,m}
      Execute a_{k,m}, collect video: R_{k,m}
    end for
  end for
  for j = 1...K do
    rank Cost(Φ(R_{k,m}), Φ(V_k)) for every m
    pick E = {elite examples}
    fit π(.) as a VAE to Ψ_{k,m} ∈ E
    pick E_exp = {Φ_f(R_{k,m}) with highest "change"}
    fit π_exp(.) as a VAE to Ψ_{k,m} ∈ E_exp
  end for
end while
```

1) Policy Structure

When trying to achieve the desired task via interaction, it is easy to simply get stuck in the local minimum around the prior. Thus we not only need to train a task policy, but an exploration policy as well.

Task policy We would like to learn an interaction policy which will allow the agent to achieve the task. Given prior Ψ_k extracted from video V_k, we propose learning a (task and exploration) policy $\pi(\Psi_k, V_k) = \Delta\Psi_k$, outputting the residual to the prior. Residual learning is common in robot learning [22] as it allow for the policy to search around the prior in order to avoid unsafe behavior. Using this residual structure also allows the policy to initialize close to the human prior, and then explore from the prior as a starting point. Both of our policies are neural network based. The task policy π will try to maximize the robot's performance with respect to the demonstration video.

To be able to sample around the prior, the policy needs to learn a distribution and not just a mean prediction. For a single human video, there are multiple different ways a robot can perform the task. A naive stochastic neural network policy would not be able capture this multi-modal distribution and hence has difficulty in generalizing to new videos. We leverage Variational Auto-Encoders (VAEs) [26, 52] which are popularly used to capture multi-modal distributions. In particular, we fit a Conditional VAE [68] to learn a mapping from a set of samples $\Delta\Psi_{k,m}$ and an embedding of the input demonstration video $\phi(V_k)$. We condition the distribution on the video, V_k and learn to encode the prior residuals. The encoder, $q(z|c, x)$ takes input $x = \Delta\Psi_{k,m}$, and c is an embedding of the human video $\phi(V_k)$. The decoder $p(x|z, c)$ takes a latent sample from $p(z)$ as well as the human video embedding. At inference time, we can use the policy, π to output $\Delta\hat{\Psi}_k$ (the residual), conditioned on video $\phi(V_k)$ and latent $z \sim \mathcal{N}(0, 1)$, where $\pi = p(x|z, c)$. Since this policy is conditioned on an input video, with enough collected data it is able to generalize to new human demonstrations as well.

Exploration Policy On the other hand, the exploration policy will try to explore around the prior. Many methods of exploration have been studied in literature, for example intrinsic motivation [45], or maximizing state coverage [17]. Instead, our exploration policy, π_{exp}, aims to maximize the change that the agent causes in the environment. Since our actions are close to the prior, it is likely that any changes caused by the agent in the environment will be meaningful and not destructive. Mathematically, for a given video R_k this can be described as:

$$c_k = \max_{i,j} ||\Phi_f(R_{k,i}) - \Phi_f(R_{k,j})||_2 \quad (1)$$

where i and j are different frames of the video and Φ_f is a frame-by-frame embedding of the video. This exploration policy uses the same exact inputs and setup as the task policy, as well as the CVAE architecture.

As the robot interacts with the environment, we want the task policy to improve to achieve more success. Thus, we need to have a notion of how good the robot is doing compared to the target human video. Given that human and robots have different morphologies, and perform tasks differently, how do we create a representation space for our objective function?

2) Representations for Human-to-Robot Video Alignment

Trying to learn correspondences between human and robot videos can be a challenging task. Prior works [65, 61, 67, 60] have attempted to achieve this via learning paired or unpaired videos from a single scene to learn a joint embedding. This would not scale to large set of in-the-wild manipulation tasks described in Figure 1. Instead of trying to learn a tight coupling between human and robots, we aim to get a comparison between human and robot video at a *high level*. We postulate that the effect that the agent had on the environment is more important than how the agent moved, since that can vary with different morphologies. We embed both robot and human videos into a space that is agnostic to the agent. We find that inpainting both human and robot from the video allows us to do so. We employ Copy-Paste Networks [30], which, given a mask of an agent, trains a network to copy information from other video frames and inpaint the area masked out. An example of this procedure can be seen in Figure 5.

Only inpainting the human out of the video is not enough to compare the two semantically, since the videos might be

(a) Robot Image (b) Robot Inpainted (c) Human Image (d) Human Inpainted image

Fig. 5: We show a sample of our agent agnostic representation, using the video inpainting method by Lee et al. [30]. An image of a human performing the task is shown in (c) and (d) shows the human inpainted from the image. Similarly for the robot, (a) and (b) show the original and inpainted images. We train segmentation models [74, 51] to obtain human and robot masks.

of different lengths, different speeds or may have other minor differences. Advances in action recognition [80, 18, 20, 5, 73, 15] have allowed models to determine if two videos which may look different are performing the same task. We use such an action recognition model from Monfort et al. [38], trained on large-scale passive data. This model takes in an entire video, and outputs an embedding. We call the composition of the action recognition model and the inpainting model our *agent-agnostic representation*. We denote this with Φ. Using this agent-agnostic representation, human video V and robot video R, our objective function is the distance:

$$|\Phi(V) - \Phi(R_k)||_2 \tag{2}$$

We use this function to train the task policy. The exploration policy objective (Equation 1) which is to maximize "change" leverages frame-wise version of Φ, denoted by Φ_f. In order to increase robustness, we sample costs over multiple embeddings with different video-level augmentations. Given a good way to align robot and human videos, how do we optimize our policies in a sample efficient way?

3) Sampling-based Optimization Procedure

RL methods have shown promise in learning by interaction. However, they remain too sample inefficient for large scale robot learning in unstructured settings. Instead, we propose a simple zeroth order sampling based alternative, in a similar fashion to CEM [56]. In our sampling procedure, we initially extract the prior from a third person human video V_k, and samples residuals:

$$\Delta\Psi_k \sim \mathcal{N}(0, \sigma^2)$$

We execute these M samples $\Psi_{k,m} + \Delta\Psi_{k,m}$ in the real world, and capture resulting videos $R_{k,m}$. We aim to fit π to the best performing samples. We repeat this process till convergence. In the following iterations, instead of sampling only from $\mathcal{N}(0, \sigma^2)$, we sample from π as well. Using the objective (agent-agnostic) functions described above, we rank trajectories based on these costs and fit the policy to the 10 highest ranking $\Delta\Psi_{k,m}$. This set, E, is the set of "elites" in CEM. The result of this procedure are trained exploration and task policies. We provide an overview in Algorithm 1.

IV. Experimental Setup

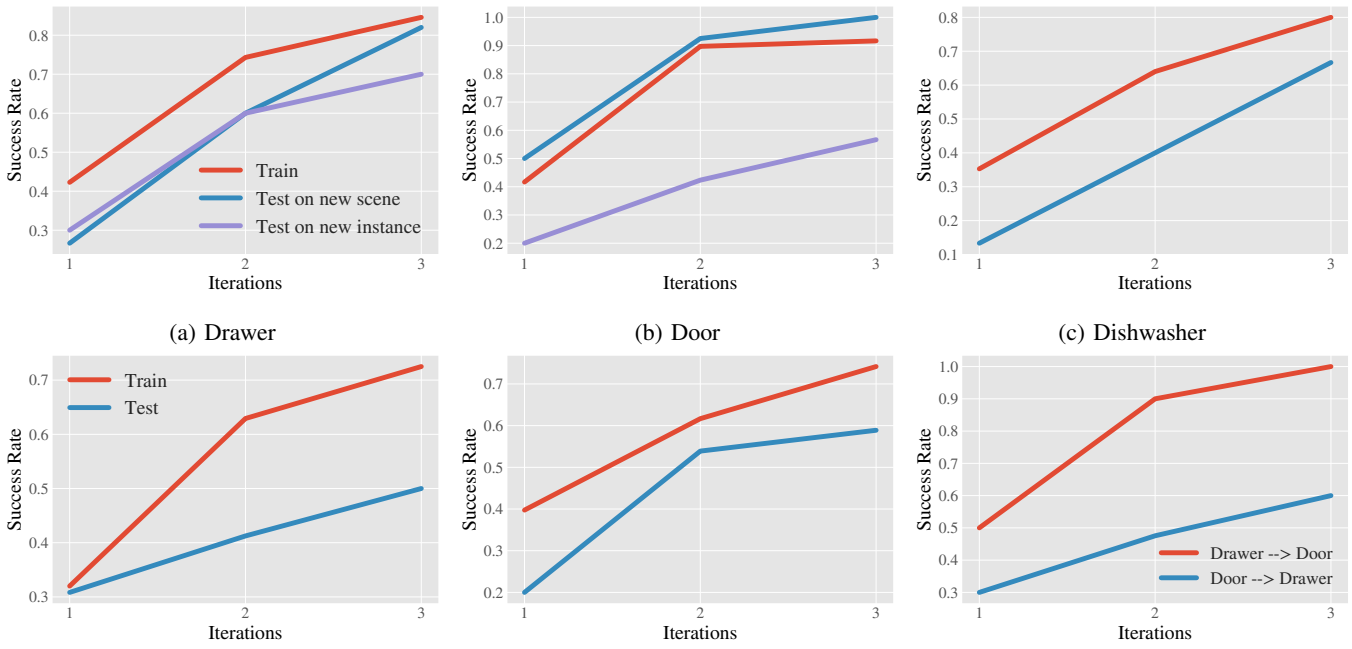
A. Experimental Details

Hardware In order to perform manipulation in the wild, a mobile robot is needed. Thus, we use the Stretch Robot [25]. This is a mobile base with a 6 dof arm and gripper. Pictures of the hardware setup can be seen in Figure 3. We use a Cartesian position control to command the translation of the wrist, and an in-built orientation controller for the rotation. The default gripper comes with suction cups as fingertips. Our setup uses an Intel Realsense D415 depth camera.

Environment and Data Collection We perform experiments on every-day objects and settings, for example drawers, dishwashers, fridges in different kitchens, doors to various cabinets. Data collection is performed in various in the wild settings. Each of the tasks presented in Figure 6 was trained over three demonstrations. Our setup involves 20 tasks seen in Figure 1 and in the Appendix.

B. Baselines and Ablations

Our method relies on several key ideas, such as policy learning from interactions, the high-level video alignment to compare robot trajectories and human demonstrations, and the task-agnostic policy we use. We test both the performance and sensitivity of WHIRL to design choices. We compare against several state-of-the-art baselines. Performing RL in the real world is infeasible [81] for our large set of tasks and in diverse settings. Thus we compare against offline RL, which learn interaction policies from data, and do not interact with the environment at training time. We modify a SOTA method in offline RL, Conservative Q-learning (CQL) [28] to work with our extracted human priors. The policy predicts the residual to the prior, just like ours, even using the same inputs as well. The reward function used is the negative L2 error between the 3D ResNet [15] embedding of target and robot videos. This baseline is trained with same number of samples as WHIRL (30 samples x 2 training iterations). We call this approach `CQL`. We also train `CQL` with the same objective function as



(a) Drawer (b) Door (c) Dishwasher

(d) Object in Shelf (e) Multi-task (f) Task-level Generalization

Fig. 6: We present results of our thorough investigation of WHIRL on various kitchen tasks such as drawer (a), door (b) and dishwasher (c) opening and closing, a task involving picking and placing different objects into shelves (d). We test multi-task policies (e) trained on a subset of the tasks and generalization between tasks (f). We report training and testing success rates (out of 1) for 3 iterations of training.

WHIRL (agent-agnostic). We call this baseline `CQL-ours`. We compare against competing SOTA approaches for learning joint human and robot embedding spaces. We train a Time Contrastive Network (TCN) [61] to extract a representation from human and robot videos. We call this baseline `CQL-TCN`. The reward used for CQL is the the distance (between human and robot videos) in TCN embedding space. Similarly, Cycle-GAN [82] has been shown to be useful translating robot and human demonstrations [67]. In the same fashion as Smith et al. [67], we employ Cycle-GAN representations (trained on both human and robot videos) as an embedding space for the reward function. We refer to this baseline as `CQL-CycleGAN`. Finally, we compare to an off-the-shelf implementation of Behavior Cloning (`BC`). This approach is similar to WHIRL, however is without iterative refinement. A key difference is that the policy is trained with a standard L2 loss on the output actions, unlike our VAE-based policy.

V. Results

We evaluate WHIRL in various real world, in-the-wild settings in order to answer the following questions:

- Can WHIRL work for a large set of in-the-wild robot manipulation tasks?
- How can we use WHIRL to generalize to new scenes, objects and settings?
- How much do the individual components (i.e. policy learning and agent agnostic cost function) of WHIRL help?
- How does WHIRL compare agents SOTA approaches?

We attempt to answer these via various experiments on in-the-wild tasks, analyzing generalization capabilities of WHIRL on a few tasks, comparing to competing methods, and addressing the question of the importance of the individual components of our method, such as iterative improvement, our agent-agnostic objective and the exploration policy. In Figure 6 we present the results of our experiments on four tasks: opening and closing a drawer, door and dishwasher, as well as placing different objects in shelves. The setup is as described in Section IV.

A. Robot Learning in the Wild

We provide results on a large set of 20 tasks, ranging from turning on a water tap to folding a shirt. Images of these tasks can be seen in Figure 1 or in the Appendix. We show that WHIRL is able to to scale to a wide variety of tasks that involve large fixed objects such as fridges, or smaller rigid objects such as our ball-in-hoop task, or handling soft objects such as our shirt folding and whiteboard cleaning tasks. We are able to train these in only a few hours, in diverse locations and settings. We provide videos of these tasks in the supplementary material and at `https://human2robot.github.io`.

B. Evaluation and Comparison to Baselines

Tasks We perform a drawer opening and closing task in a kitchen. We report results averaged across three human demonstrations on two drawers (Figure 3a), and 3 iterations of WHIRL. Similarly to the drawer task, we perform a door opening and closing task. Figure 3b shows the task setup. We train on two doors and test on the third one. We perform 3 iterations of WHIRL. The third task involves opening and

closing a dishwasher, as shown in Figure 3c. Due to a lack of dishwashers in the kitchens, we only train on one dishwasher, and test on a held-out dishwasher in a different scene. For the fourth task we train a policy for picking and placing four objects in three different shelves. These objects vary in size and shape (bottles, cans, etc.). We provide a single demonstration for each object. We test on two held-out objects and a held-out shelf placement (starting point is top shelf and goal is middle shelf). Details about the task can be found in the Appendix.

Training Evaluation We look into the (training) results running WHIRL on the various tasks described above. In Figure 6a, we show a learning curves of success rates over 3 iterations of WHIRL for the drawer task. The training curve (red) shows an increase from about a 43% success rate to 83% success rate after two iterations. The initial success rate is the success rate of the prior. We see a clear improvement during training with WHIRL. For the door task (Figure 6b, red), the training curve shows an increase in performance from about 40% success to 92 % success, indicating that WHIRL is able to learn to improve iteratively. The training curve for the dishwasher task (Figure 6c, red) shows that WHIRL also improves for this task, similarly to the drawer and door tasks. Figure 6d provides learning curves for the task of picking and placing objects from shelves. We can see that the the train (red) curves show improvement. However, we did note that this task required a lot more precision than the kitchen tasks presented above, which is why we see the test success rate to be much lower than the train one. We often found that if the robot predicted the waypoints incorrectly by even a couple of centimetres, it would hit a shelf and get stuck.

Comparison to Baselines We compare WHIRL to both offline RL and Behavior Cloning. We present results of multiple instances of offline RL in Table I (we report and compare training results). All methods had the same amount of data to train on. We report success rates out of 1. Our method strongly outperforms all the baselines. Offline RL, especially with smaller datasets, has difficulty in learning without any online interaction. Learning both actor and critic require data that covers more of the action space than is likely available in the wild. Interestingly, `CQL-ours` tends to outperform other approaches such as `CQL-CycleGan` (similar to that presented by Smith et al. [67]) `CQL-TCN` (similar to Sermanet et al. [61]), and `CQL` [28]. Similarly, Behavior Cloning, which uses our agent-agnostic cost function to filter the top trajectories, mostly outperforms the offline RL approaches. This indicates that our objective function is able to differentiate between good and bad trajectories in many algorithmic settings.

C. Generalization to New Instances

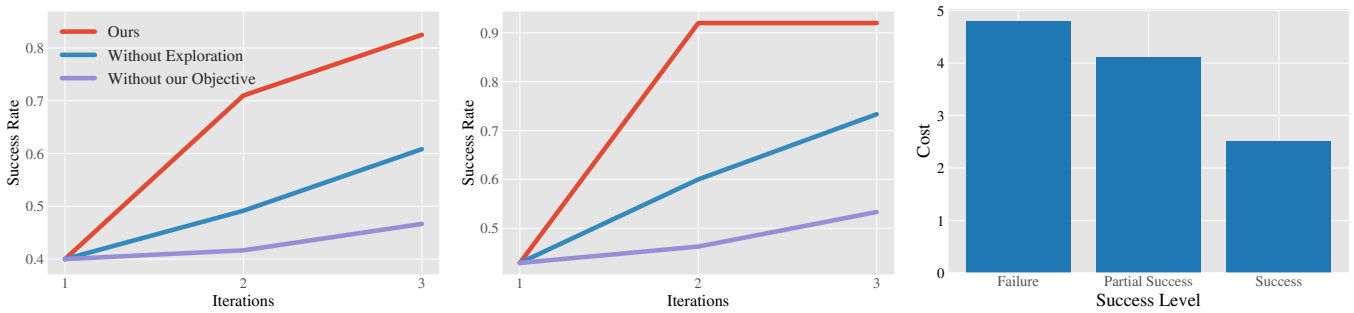
We also evaluate how well policies trained with WHIRL are able to perform on new instances of the same task that they were trained on. In the drawer task, we test the policy on a held-out drawer. The learning curve for the held-out drawer (Figure 6a, blue) shows that the achieved success is lower than that on the training drawers. The detected prior for each demonstration may have different biases and errors. A policy may not necessarily transfer as well to a new demonstration without any training, however, as we see here, we expect improvement as there are commonalities in the structure of the task. Another common aspect is the camera and geometry information for both train and test demonstrations (since the view is the same). For the door task (Figure 6b, blue), interestingly, the success rate is higher here than the train one. As discussed previously, we expect that this is due to the strength of the prior for this specific door. For the object task, The policy was tested on two held objects and held out shelf placements. The test (blue) curves in Figure 6d shows a definite improvement in success rate over multiple iterations. From these experiments, we see that WHIRL does have the ability to generalize to new instances of the same task it was trained on.

	Drawer	Door
No iterative improvement:		
Behavior Cloning	0.53	0.30
Offline RL (`CQL-ours`)	0.47	0.30
Offline RL (`CQL-CycleGAN`) [67]	0.23	0.30
Offline RL (`CQL-TCN`) [61]	0.27	0.20
Offline RL (`CQL`) [28]	0.33	0.13
No agent-agnotic objective:		
WHIRL (ours)	0.47	0.53
No Exploration Policy:		
WHIRL (ours)	0.60	0.73
WHIRL (ours)	**0.83**	**0.92**

TABLE I: We present a set of evaluations on two real world tasks: drawer and door. The task is to imitate the third person demonstration of the human. The results presented our averaged over 30 trials.

D. Generalization to New Scenes:

To analyze how well WHIRL performs in a new scene, where the calibration and geometry are different, we try the trained policy on a drawer in a different part of the kitchen. Note that the camera angle and view also change. The resulting curve is shown in Figure 6a (purple). We see that similar to the test curve on the held-out drawer, there is definitely an increase in the success rate, however, the performance is still worse than the train drawers. In the case of the door task (Figure 6b, purple) the performance of this policy in a new setting is significantly worse, unlike the drawer task,. This is likely due to mismatches in the train and test demonstrations. Although the performance is not as strong, there is still an improvement in success rate from 20% to about 57%. For the dishwasher task (Figure 6c, blue) we see a strong improvement in the success rate, similarly to the other two tasks. As expected, the policy does not perform as well as it does on the train dishwasher. We see that WHIRL allows for generalization to new scenes, however, in most cases the performance is worse than running WHIRL on new instances, most likely due to large visual changes, as well as different geometry and calibrations.



(a) Drawer (b) Door (c) Cost Function

Fig. 7: We ablate different aspects of WHIRL. We analyze the need for our task-agnostic exploration policy, as well as our agent-agnostic representations. (a) is the drawer task, (b) is the door task. In (c) we analyze on the drawer task how our cost functions compare for different levels of success obtained by a trajectory.

E. Generalization to New Settings:

In order to test the generalization between tasks, we test the trained policy on a held-out door (Figure 3b), in order to test the task level generalization. We see a strong performance on the door, as shown in Figure 6f (red). We see generalization from the drawer policy to the door task. We suspect that the final high success rate is due to the fact that the prior for this specific door was much more accurate than for other instances or tasks. Nevertheless, we see an improvement on the held-out door as the policy is trained more, indicating that WHIRL is able to improve the performance for not only the task at hand, but also allows for some degree of generalization across tasks. For transferring the policy trained on the door task to the drawer task, we see in Figure 6f (blue) an improvement in performance on the drawer task, similarly to the policy trained on the drawer task. This definitely indicates that there is some degree of task-level generalization in WHIRL.

F. Multi-task Generalization

In the previously described experiments, we trained policies for one task only, even if it was tested on another one. With this experiment, we aim to answer how training a joint policy would work. Using a similar approach as described above for each of the 3 tasks (drawer, door and dishwasher) we train the same policy π. We test this policy on the same test scenes as the previous experiments. We present results in Figure 6e. We can see clear improvement over all the iterations, both in the training and testing instances. However, the success rates are lower than policies trained for individual tasks which makes sense. We see a big increase from the first iteration. This is likely because there is more data thus generalization becomes a little easier as compared to individual policies. However, as more training happens this does not remain the case.

G. Sensitivity of WHIRL

We test the sensitivity of our method to both the inpainting process and the exploration policy. Firstly, We train WHIRL without the agent-agnostic representations. Secondly, We also compare WHIRL against a version which does not use the exploration policy. In Figure 7 we perform ablations to test the sensitivity of WHIRL to various components. In previous experiments, we have seen that the iterative improvement provides a lot of benefit, as evidence by the increase in performance over iterations in almost all of the tasks and scenarios shown in Figure 6, as well as the boost in performance over offline RL methods, as shown in Table I.

Agent-Agnostic Objective We train our policy without our agent-agnostic objective function, on both the drawer and door tasks, as shown in Figure 7a and 7b (purple). We see almost no gain in success rate from the initial samples from the prior in either task. This is likely due to the fact that video alignment models focus too much on the agents, thus the true top trajectories might not be selected. We conclude that an agent-agnostic objective is crucial to the success of WHIRL.

Exploration We test a version of WHIRL without our exploration policy. We see this in Figure 7a and7b, in the blue curve. While the performance of this version of WHIRL outperforms the ablation that does not use agent-agnostic objective, we still see a drop in success rate from the WHIRL with the exploration policy. This shows that the method can learn without biasing exploration around maximizing "change" in the environment, but it will be slower.

Analyzing our Objective Function In Figure 7c, we show a bar plot of distances in our agent-agnostic embedding space for the drawer task. We present the cost for three types of trajectory. Note that each category is averaged over multiple trajectories. "Failure" trajectories completely fail, and the robot never touches the drawer. "Partial Success" trajectories open the wrong drawer or do not open/close the drawer fully. "Success" trajectories match the human demonstrations. We see that there the cost decreases between these three and that there is big drop between "Partial Success" and "Success". Since the cost is not close to 0, we expect there to be noise in the measurement of video alignment. However, empirically we found that our embedding space was robust enough to differentiate between successful and unsuccessful trials. In future work, we hope to train such an embedding space.

VI. Conclusion

We propose WHIRL, an efficient real-world robot learning algorithm that can learn manipulation policies in-the-wild from

human videos. Our method leverages advances in computer vision to understand human videos and obtain priors such as hand interactions, movement and direction. WHIRL is able to efficiently improve in the real world by using our sampling-based policy optimization strategy and agent-agnostic representations, as well as our proposed exploration strategy that maximizes the changes seen in the environment. We perform a thorough evaluation in terms of absolute performance, comparison to SOTA baselines, and generalization to new tasks and scenes on multiple tasks on real kitchens and see strong results. We show that our method is able to work on 20 different tasks in the wild (outside lab settings). WHIRL is a first step towards learning robot skills from watching internet videos. In the future, we hope to build upon our method to try to solve tasks without online interactions and human demonstrations, i.e. directly learn policies from passive offline video datasets such as YouTube.

Acknowledgments

We thank Jason Zhang, Yufei Ye, Aravind Sivakumar, Sudeep Dasari and Russell Mendonca for very fruitful discussions and are grateful to Ananye Agarwal, Alex Li, Murtaza Dalal and Homanga Bharadhwaj for comments on early drafts of this paper. AG was supported by ONR YIP. The work was supported by Samsung GRO Research Award, NSF IIS-2024594 and ONR N00014-22-1-2096.

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 1

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NIPS*, 2017. 1

[3] Samarth Brahmbhatt, Cusuh Ham, Charles C. Kemp, and James Hays. ContactDB: Analyzing and predicting grasp contact via thermal imaging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019. 3

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv:1606.01540*, 2016. 1

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 3, 6

[6] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021. 3

[7] Boyuan Chen, Pieter Abbeel, and Deepak Pathak. Unsupervised learning of visual 3d keypoints for control. *arXiv preprint arXiv:2106.07643*, 2021. 3

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 3

[9] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018. 3

[10] Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *CVPR*, 2013. 2

[11] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin S Wang, Abitha Thankaraj, Karanbir Singh Chahal, Berk Calli, Saurabh Gupta, et al. Rb2: Robotic manipulation benchmarking with a twist. In *NeurIPS Datasets and Benchmarks Track (Round 2)*, 2021. 1

[12] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *NIPS*, 2017. 2

[13] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1801–1810, 2019. 3

[14] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015. 2, 3

[15] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 3, 6, 13, 14

[16] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *CoRL*, 2017. 2

[17] Justin Fu, John D Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. *NIPS*, 2017. 5

[18] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, pages 5842–5850, 2017. 2, 3, 6

[19] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *NeurIPS*, 2018. 1

[20] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018. 3, 6

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 3

[22] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *ICRA*, 2019. 5

[23] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021. 1

[24] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017. URL http://arxiv.org/abs/1712.06584. 2

[25] Charles C Kemp, Aaron Edsinger, Henry M Clever, and Blaine Matulevich. The design of stretch: A compact, lightweight mobile manipulator for indoor human environments. *arXiv preprint arXiv:2109.10892*, 2021. 6, 13

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[27] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *arXiv preprint arXiv:1906.11883*, 2019. 3

[28] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. 6, 8

[29] Jangwon Lee and Michael S Ryoo. Learning robot activities from first-person human videos using convolutional future regression. In *CVPR Workshops*, pages 1–2, 2017. 3

[30] Sungho Lee, Seoung Wug Oh, DaeYeun Won, and Seon Joo Kim. Copy-and-paste networks for deep video inpainting. In *ICCV*, 2019. 5, 6, 14, 15

[31] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*, 2016. 1

[32] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *CVPR*, pages 7083–7093, 2019. 3

[33] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *ICRA*, 2018. 3

[34] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 3

[35] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 2

[36] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. 1

[37] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 1

[38] Mathew Monfort, Bowen Pan, Kandan Ramakrishnan, Alex Andonian, Barry A McNamara, Alex Lascelles, Quanfu Fan, Dan Gutfreund, Rogerio Feris, and Aude Oliva. Multi-moments in time: Learning and interpreting models for multi-action video understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 6, 14, 15

[39] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. *arXiv preprint arXiv:2011.06431*, 2020. 3

[40] Anh Nguyen, Dimitrios Kanoulas, Luca Muratore, Darwin G Caldwell, and Nikos G Tsagarakis. Translating videos to commands for robotic manipulation with deep recurrent neural networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3782–3788. IEEE, 2018. 3

[41] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 3

[42] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *CVPR*, 2021. 3

[43] Jyothish Pari, Nur Muhammad, Sridhar Pandian Arunachalam, Lerrel Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 3

[44] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009. 3

[45] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 5

[46] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *ICLR*, 2018. 2

[47] Sören Pirk, Mohi Khansari, Yunfei Bai, Corey Lynch, and Pierre Sermanet. Online object representations with contrastive learning. *arXiv preprint arXiv:1906.04312*, 2019. 3

[48] Dean A Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *NIPS*, 1989. 3

[49] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering

self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019. 1

[50] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *International Conference on Humanoid Robots*, 2007. 3

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 4, 6

[52] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014. 5

[53] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017. 2, 4

[54] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *CVPR (ICCV) Workshops*, pages 1749–1759, October 2021. 2, 3, 15

[55] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. 3

[56] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013. 6

[57] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1964. 4

[58] Rosario Scalise, Jesse Thomason, Yonatan Bisk, and Siddhartha Srinivasa. Improving robot success detection using static object data. In *IROS*. IEEE, 2019. 3

[59] Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 486–487. IEEE, 2017. 3

[60] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. In *RSS*, 2017. 1, 3, 5

[61] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018. 5, 7, 8, 15

[62] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9869–9878, 2020. 2, 3, 4, 14, 15

[63] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14), 2021. 3

[64] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018. 3

[65] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019. 2, 3, 5

[66] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 14

[67] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *RSS*, 2020. 2, 3, 5, 7, 8

[68] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *NeurIPS*, 2015. 5

[69] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020. 1, 3

[70] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021. 15

[71] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012. 1

[72] Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A Otaduy, Dan Casas, and Christian Theobalt. Rgb2hands: real-time tracking of 3d hand interactions from monocular rgb video. *ACM Transactions on Graphics (TOG)*, 39(6): 1–16, 2020. 2

[73] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 3, 6

[74] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6, 15

[75] Haoyu Xiong, Quanzhou Li, Yun-Chun Chen, Homanga Bharadhwaj, Samarth Sinha, and Animesh Garg. Learning by watching: Physical imitation of manipulation skills from human videos. *arXiv preprint arXiv:2101.07241*, 2021. 3

[76] 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*, 2015. 3

[77] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020. 3

[78] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tian-

hao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *RSS*, 2018. 2

[79] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *CoRL*, 2022. 3

[80] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *ICCV*, pages 2248–2255, 2013. 3, 6

[81] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020. 6

[82] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017. 3, 7

APPENDIX

A. Videos

Videos of our results can be found at:

- https://human2robot.github.io

B. Implementation Details

1) Real Robot Setup

We use use the Stretch robot from Kemp et al. [25]. This is a robot with a mobile base and a wrist with suction cups as fingertips. All 6 degrees of the freedom of the robot are controllable, and we use code provided in https://github.com/hello-robot to control the robot. To capture videos of humans and the robot we use an Intel Realsense D415. We obtain both depth and RGB images from this camera setup. Each human demonstration takes about 30 seconds. Similarly, each robot episode also takes 30-45 seconds. Overall training takes anytime between 4 and 6 hours, including time to compute inpainted videos (which is the bottleneck in terms of software).

Our real robot tasks are all in the wild (i.e. outside labs). We perform tasks using everyday objects and in every day locations such as kitchens etc. Due to torque limits on the robots, we had to use non-standard objects for some settings, such as the ball in hoop task, as the robot's gripper cannot grasp a heavier and bigger basketball. In Figure 9 we present a full list of tasks with images. We show details on train and test objects for our shelf pick-and-place task in Figure 8.

2) Data Collection

Human demonstrations are very easy obtain and each takes about 30 seconds to collect. The tasks presented in Figure 1 are trained from one demonstration. During each iteration, 30 samples were taken, and we used the top 10 ranking ones to fit the policy. The Stretch robot [25] took less than 1 minute per episode, thus around 20 minutes per iteration.

(a) Training Objects (b) Test Objects

Fig. 8: Images of the objects we used in our shelf pick-and-place task. Objects a-d are train objects, objects e and f are test objects

3) Hyper-parameters and Design Choices

Our policy is a 4 layer MLP, which takes as input an embedding of a demonstration video as well as the prior. The output of the policy is the residual to the prior. We process the human video using an action-recognition pipeline, using features from state-of-the-art pretrained action recognition models such as SlowFast 3D ResNets [15]. We train the policy as a Variational Auto-Encoder, using a KL-divergence loss with weight 0.0005 to train the model and latent dimension = 4. However, larger latent dimensions work well too, but this

(a) Drawer (b) Door (c) Dishwasher (d) Pulling Garbage Bag

(e) Ball in Hoop (f) Cleaning Whiteboard (g) Garbage Can (h) Arrange Chair

(i) Stacking Cups (j) Stacking Dice (k) Fridge (l) Place Hat

(m) Remove Shirt from Hanger (n) Remove Lid (o) Turn Off Light (p) Shelf Pick-and-Place

(q) Fold Shirt (r) Pull Plug from Socket (s) Open Tap (t) Toaster

Fig. 9: Images of our 20 tasks.

should be dependent on the size of the action space for the robot (which in our case is 13 dimensional). Our exploration policy is structured in a similar manner. For the human prior, we use the hand-object interaction detector from Shan et al. [62]. We employ Copy-Paste Networks [30] for inpainting humans and robots from videos. We use action recognition models such as Multi-Moments [38] and SlowFast 3D ResNets [15] as our representation space (Φ) for aligning humand and robot videos. Furthermore, for measuring "change" in the environment we used features from the VGG16 [66] network. Our exploration policy uses the same exact architecture. We use the following video augmentations for our representations: Salt-Pepper Jittering, Random Crops, Gaussian Blurs, Vertical and Horizontal Flips). For the hand-object and wrist detection modules we used default parameters provided by the respective codebases. To smooth the predictions we used the filter

from https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html. We used about 200 labeled images of the robot to train the instance segmentation module, with the default network sizes from the codebase. All of our image and video sizes were 640 x 480. For the policy training module, our optimization approach fits to the top 10 (out of 30) samples.

4) Codebases

We use the following codebases:

- For hand-object detection we use the codebase from Shan et al. [62] , https://github.com/ddshan/hand_detector.d2
- For wrist detection we use FrankMocap [54], https://github.com/facebookresearch/frankmocap
- For the instance segmentation module we use code from https://github.com/wkentaro/labelme to label robot instances, and use code from Detectron2 [74] (https://github.com/facebookresearch/detectron2) code provided in https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html to train an instance segmentation model.
- For the TCN [61] baseline we use code from https://github.com/kekeblom/tcn
- For the CycleGAN baseline we use code from https://github.com/Lornatang/CycleGAN-PyTorch
- Our Inpainting model [30]: https://github.com/shleecs/Copy-and-Paste-Networks-for-Deep-Video-Inpainting
- We use the model from Monfort et al. [38] (https://github.com/zhoubolei/moments_models) to compute representations for our cost functions
- Our offline RL baselines are from Takuma Seno [70] (https://github.com/takuseno/d3rlpy)
- https://github.com/okankop/vidaug