VIEW: Visual Imitation Learning with Waypoints

Ananth Jonnavittula · Sagar Parekh · Dylan P. Losey

Abstract Robots can use Visual Imitation Learning (VIL) to learn everyday tasks from video demonstrations. However, translating visual observations into actionable robot policies is challenging due to the high-dimensional nature of video data. This challenge is further exacerbated by the morphological differences between humans and robots, especially when the video demonstrations feature humans performing tasks. To address these problems we introduce **V**isual **I**mitation l**E**arning with **W**aypoints (VIEW), an algorithm that significantly enhances the sample efficiency of human-to-robot VIL. VIEW achieves this efficiency using a multi-pronged approach: extracting a condensed prior trajectory that captures the demonstrator's intent, employing an agent-agnostic reward function for feedback on the robot's actions, and utilizing an exploration algorithm that efficiently samples around waypoints in the extracted trajectory. VIEW also segments the human trajectory into grasp and task phases to further accelerate learning efficiency. Through comprehensive simulations and real-world experiments, VIEW demonstrates improved performance compared to current state-of-the-art VIL methods. VIEW enables robots to learn a diverse range of manipulation tasks involving multiple objects from arbitrarily long video demonstrations. Additionally, it can learn standard manipulation tasks such as pushing or moving objects from a single video demonstration in under 30 minutes, with fewer than 20 real-world rollouts. Code and videos here: https://collab.me.vt.edu/view/

Keywords Visual Imitation Learning, Deep Learning, Few-shot Learning

A. Jonnavittula
Mechanical Engineering Department, Virginia Tech
E-mail: ananth@vt.edu

S. Parekh
Mechanical Engineering Department, Virginia Tech
E-mail: sagarp@vt.edu

D. Losey
Mechanical Engineering Department, Virginia Tech
E-mail: losey@vt.edu

1 Introduction

Imagine teaching a person to pick up a cup placed on a table. The quickest method is often to physically demonstrate this task. Through physical demonstration, the observer can discern which object to pick up and how to manipulate that object. Humans efficiently learn everyday tasks in this way, including moving items, pouring tea, or stirring the contents of a pan.

Teaching robots these same tasks, however, proves to be more cumbersome. Typically, robots employ either Imitation Learning (IL) or Reinforcement Learning (RL) methods. IL generally requires many demonstrations from humans to obtain effective policies [15,21]. During this process, the human teacher often needs to kinesthetically guide or teleoperate the robot to show it exactly what actions it should take. On the other hand, RL methods require a substantial number of rollouts for robots to perform even simple tasks [30,45]. Additionally, defining appropriate reward functions for reinforcement learning poses a challenge, particularly in unstructured everyday settings [13].

In this paper we therefore study how robots can learn tasks by *watching* humans. The human provides a demonstration directly in the environment (e.g., physically picking up a cup), and the robot collects an RGB-D video of the human's demonstration. Our objective is for the robot to leverage this single video to learn the task and correctly manipulate the same object. The primary issue here lies in the overload of information conveyed by video demonstrations. Each video is comprised

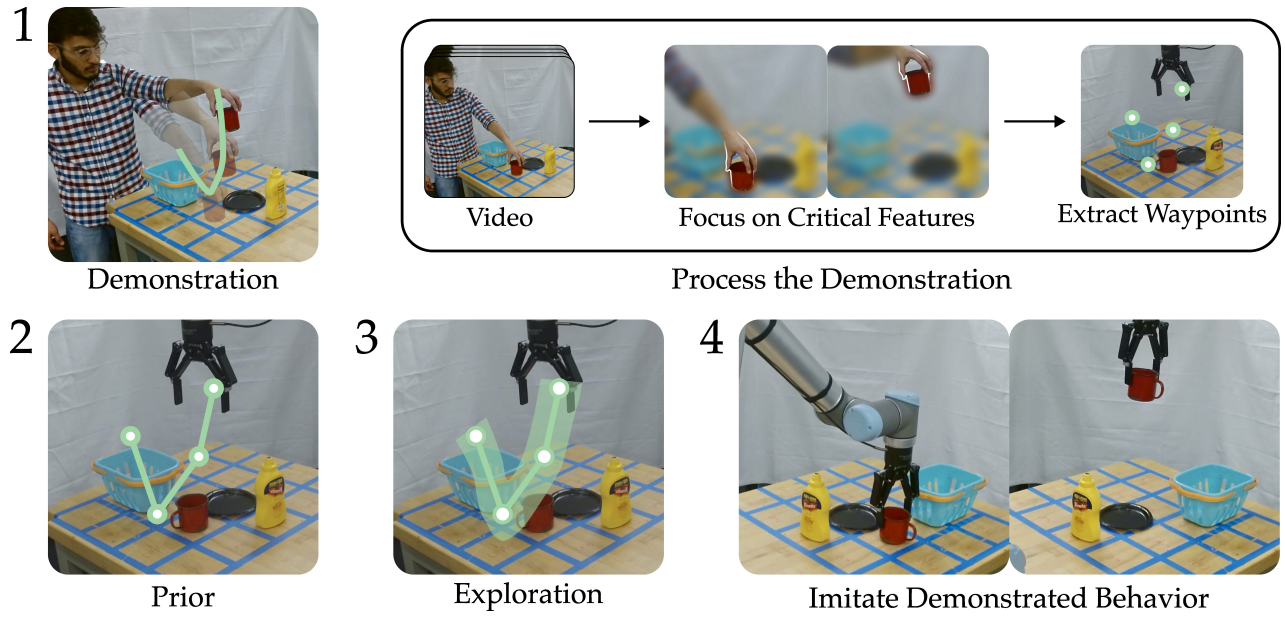

Fig. 1 Robot learning from visual demonstration. 1) A human demonstrates the task directly in the environment: here we use the example of picking up a cup. Under our proposed approach, the robot processes a single video of that demonstration to selectively focus on important features such as the human hand and the manipulated object. From these trajectories the robot obtains *waypoints* that capture the critical parts of the task (e.g., grasping the cup). 2) These extracted waypoints serve as a prior for the correct robot trajectory. 3) In practice, simply executing this prior rarely leads to task success due in part to the morphological differences between human and robot (in this case, the robot misses the cup entirely). Therefore, the robot must explore in a region around the initial waypoints to iteratively improve its trajectory. 4) After repetitively interacting with the environment, the robot learns to successfully imitate the behavior demonstrated in the human video.

of thousands of image frames, and each frame contains numerous pixels. These raw pixel values — when seen in isolation — are not sufficient for the robot to determine what actions it should take (i.e., how to move the robot arm). Consequently, robots that learn from video demonstrations must extract pertinent information from a large amount of data.

To address this fundamental problem, our hypothesis is that robots do not need to reason over *all* the video data. Consider our motivating example of picking up a cup: when humans learn by watching other humans, we do not focus on environmental clutter or extraneous details. Instead, we just need to see where the human grabs the cup and how they carry it. At a high level, we can think about these critical parts of the task as *waypoints*: the cup's initial position, the human's hand configuration when grasping, key frames along the cup's motion, and where the human finally places the cup. Robots that can learn these waypoints from the human's video demonstration will be able to perform the overall task without having to reason over every single aspect of every video frame.

We leverage this hypothesis to develop VIEW: Visual Imitation lEarning with Waypoints (see Figure 1). Our approach starts with a video of the human performing their desired manipulation task. We then process that video to get an initial trajectory (e.g., a best guess) of how the robot should complete the same task. To obtain this initial guess we extract the human's hand trajectory, and then autonomously identify the critical waypoints along that trajectory in visual and Cartesian spaces. In an idealized scenario, the robot could directly execute this initial trajectory and complete the task. However, the initial trajectory almost always fails because of (a) the morphological differences between human demonstrator and robot learner and (b) the sensor noise in the initial RGB-D video. Returning to our cup example, we often find that — even though the video shows the human picking up the cup — the robot's extracted trajectory misses that cup entirely.

Accordingly, the second part of VIEW focuses on iteratively improving the robot's prior and correctly completing the task. We develop sampling strategies so that the robot can intelligently explore around its waypoints. This includes waypoints where the robot needs to grasp an object (e.g., pick up the cup) and waypoints where the robot is manipulating that object (e.g., carrying the cup to a goal location). Again, we rely on our hypothesis: instead of reasoning about every aspect of the video, we focus on the object's location in the waypoint frames. This leads to an iterative learning process where the robot corrects its initial trajectory and eventually

completes the original task shown in the video. Overall, VIEW enables robot arms to efficiently learn everyday tasks such as picking, pushing, or moving objects, requiring fewer than 20 real-world trials and less than 30 minutes from demonstration collection to successful task execution. Additionally, our method enables robots to learn from long horizon videos that involve a combination of tasks — such as moving a cup and pouring tea into it, or placing multiple objects in a pan — using no additional information besides the initial human demonstration video.

Overall, we make the following contributions:

Condensed prior extraction. We present a new approach for distilling a prior from video demonstrations that accurately reflects the human demonstrator's intent. We achieve this by extracting a concise set of waypoints that capture the human hand trajectory and its interaction with objects.

Agent agnostic rewards. For sample efficient exploration, the robot requires effective feedback when exploring around the waypoints in the extracted prior. To provide this feedback, we propose an agent-agnostic reward function. Our reward model only focuses on the critical components of the task — i.e., movement of the object — regardless of which agent performs the task.

Sample efficient exploration. Given morphological disparities and noise introduced during prior extraction, directly replicating the human trajectory is impractical. We introduce an algorithm that segments the prior into *grasping* and *task* phases, sequentially focusing on locating the object and replicating its movement.

Few shot adaptation. While our approach can bridge the embodiment gap between human and robot through efficient exploration around the human prior, each new task requires starting from scratch. However, the robot gains valuable insights into the morphological differences and camera noise with each solved task. By integrating a residual learner that leverages this insight with our prior extraction, we demonstrate that the robot can achieve few-shot learning on new tasks.

Comparing VIEW to baselines. We conduct a comparative analysis of our method against existing state-of-the-art approaches in visual imitation learning. Additionally, an ablation study is performed in a simulated environment to underscore the significance of each component within our framework. These comparative analyses and ablation studies collectively demonstrate our method's efficacy in enabling robots to quickly imitate a wide range of tasks based on video demonstrations.

2 Related Work

We study how robots can efficiently learn to replicate a task based on a single video demonstration. Our approach builds upon existing learning from demonstration methods, particularly those that use videos, waypoints, and human activity recognition.

Learning from demonstrations (LfD). LfD is a general learning framework [51, 58] that is used across domains such as autonomous driving [48, 10], robotics [25, 26, 27, 42, 53], and video games [2, 59, 60]. In robotics, LfD has been employed to learn from teleoperated expert demonstrations [27, 57, 28, 43], extended to include imperfect demonstrations [25, 5, 4], and combined with other modalities such as preferences [42, 72] or language [66, 39, 36]. A significant aspect of LfD in robotics involves the sourcing of demonstrations, predominantly obtained from human actions within the agent's environment. For example, in autonomous driving, demonstrations encompass steering controls similar to those the agent uses [48], while in robotic manipulation, demonstrations are acquired via direct teleoperation [26] or kinesthetic teaching [42]. This reliance on human provided demonstrations presents certain challenges, especially in robotics. Humans primarily use their hands for manipulation, whereas robots utilize end-effectors with distinct morphologies. This fundamental discrepancy limits the feasibility and diversity of the training data collected for robot learning.

To address the morphological disparities between humans and robots, some researchers have advocated for the use of tools such as reacher-grabbers that resemble grippers commonly employed in robotics [49, 62, 71, 79], or utilizing camera angles that reduce the effects of hand to gripper morphology [29, 12]. These tools facilitate the recording of demonstrations that can be more easily translated into actionable robot policies, without the need for teleoperation. While these approaches have proven effective, they do not ameliorate the underlying limitation: the demonstrations are inherently restricted in scope due to the specialized interface. In response to this challenge, there has been a shift towards compiling extensive robot demonstration datasets, like Open-X [47], aiming to establish a foundational resource akin to ImageNet for robot learning. But these datasets overlook the vast reservoir of already existing human video demonstrations, which could significantly expedite the learning process for robots. VIEW builds upon prior LfD works by learning from human demonstrations. However, VIEW focuses on learning directly from human videos, and does not rely on kinesthetic demonstrations, teleoperated inputs, or intermediary tools.

Learning from video demonstrations. There has been a parallel research focus on teaching robots with videos of robots performing the desired task [8, 52, 77, 76]. In these methods a human teleoperates the robot in the video demonstration (i.e., the video demonstration is of the *robot* completing the task), and the robot learns to imitate the resulting RGB-D video. These methods tackle the challenges that arise from a lack of explicit action information [8, 77, 76]. A significant aspect of this research is the reduction of data complexity through an emphasis on keyframes [77, 76], aiming to simplify robot learning by concentrating on achieving these specific frames. Nevertheless, these methods necessitate a large number of trials with the robot [8], and do not solve the key problem of obtaining generalized video demonstrations from humans or other agents.

Alternatively, some researchers have explored learning from human video demonstrations directly, with considerable effort dedicated to transforming human videos into a format applicable to the robot's domain [69, 35, 78, 64]. To facilitate this transformation, these methods utilize cycle consistency networks [22] to translate human videos into equivalent robot videos [78, 64]. Once videos are translated, they extract key points from the videos, which serve as the basis for learning [78, 37]. However, a significant drawback of this approach is the necessity for a vast collection of videos, showcasing both humans and robots performing tasks. This requirement poses a substantial limitation, adversely affecting the scalability of such methods.

Several approaches closely align with our method, focusing on human-to-robot imitation learning [33, 23, 24, 68, 1, 61, 9, 50, 65, 3]. These methods extract meaningful representations of a task from videos [9] or use neural networks to learn reward functions from the videos to facilitate reinforcement learning [61, 1]. Despite the promise shown by many of these methods, they share a common challenge: the necessity for a substantial number of robot rollouts in real-world scenarios to learn tasks effectively. One particularly similar approach here is WHIRL [3], which mirrors our method but employs a variational autoencoder-based exploration exploitation strategy. As we will show, WHIRL requires a large number of rollouts to converge, and struggles to scale for long horizon tasks. Additionally, it relies on video inpainting [34] that can be time-consuming. Our proposed method VIEW aims to solve these challenges by segmenting the task and sequentially solving it: in our experiments, we will directly compare VIEW and WHIRL in terms of task success and training time.

Waypoint-based learning. While the methods discussed so far primarily focus on learning action policies directly from demonstrations, a growing trend in robotics is a shift towards teaching robots to reach designated waypoints. This approach is gaining traction, particularly because it aligns well with the use of separate planning and control algorithms, allowing low-level controllers to reach goals set by high-level planners. This concept has seen application in reinforcement learning for tasks such as object pick-and-place and door opening [41]. However, the success of these algorithms hinges on the creation of meticulous reward functions to guide learning. In the domain of imitation learning, it has facilitated the learning of intricate tasks, such as operating a coffee machine [67]. Nevertheless, similar to methods discussed on learning from video demonstrations, Shi *et al.* [67] learn from a video demonstration of the robot performing the task. This distinction is crucial, as robot demonstrations bypass the morphological differences encountered when learning from human videos. Our approach to prior compression bears similarities to these waypoint-focused methods. However, it differs in that VIEW learns directly from a single human video, where the human physically performs the task without a robot.

Human activity recognition. Many of the methods discussed above rely on human intent and activity recognition for enabling robots to understand object affordance. Within the realm of robot manipulation, numerous studies have proposed methods for discerning human actions and activities from video footage [14, 38, 31, 32]. Some works extend human posture tracking to identify fine-grained activities within limited spatial contexts [40]. However, merely detecting human presence is insufficient to learn from human videos. To bridge this gap, using annotated video datasets such as SomethingSomething [19], YouCook [11], ActivityNet [6], or the 100 Days of Hands (100DOH) [63] becomes instrumental. The 100DOH dataset is particularly valuable due to its detailed object interaction annotations. Building upon prior works, our approach utilizes the 100DOH framework to extract crucial data on hand positioning and interactions with objects.

In contrast to the many of the methods discussed here, VIEW distinguishes itself by focusing on sample-efficient learning directly from human videos. Our approach aims to teach robots manipulation tasks — such as picking or moving objects — with minimal human supervision. The only interaction required from the human is providing a single video demonstration.

3 Problem Statement

We consider manipulation tasks in unstructured environments. First a human teacher physically demon-

strates their desired task within the robot's workspace. During this demonstration the robot is moved out of the way (i.e., the human does not interact with the robot) and the robot records the human's behavior with a stationary RGB-D camera. After the demonstration is complete the video is provided to the robot, and the robot must learn how to replicate the same task based on this single video. We highlight that the human and robot have morphological differences — e.g., the human's hand is different from the robot's gripper — and so the way the human performed the task may not transfer directly to the robot arm.

Environment. We formulate the robot's environment as a Markov Decision Process without rewards: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T} \rangle$. The robot's end-effector position in Cartesian space is its state s, and the robot's workspace becomes its state space $\mathcal{S}$. In every state the robot can take an action $a \in \mathcal{A}$ which is the end-effector velocity. This action moves the robot to a new state s' based on the environment transition probability $\mathcal{T}(s' \mid s, a)$. We assume the environment remains unchanged between the human demonstration and the robot's learning activity; that is, all objects maintain their positions. Additionally, we only consider scenarios where the environment is captured from a fixed camera perspective.

Video Demonstration. The robot learns from a single video demonstration (V_i) of a human performing the task (τ_i) captured using a stationary RGB-D camera. The human does not interact with the robot beyond providing this video. Although the human only provides one video for task τ_i, they may provide demonstrations for multiple tasks: e.g., the robot could receive a set of n videos $V_1, \ldots, V_n$ for n different tasks $\tau_1, \ldots, \tau_n$. The robot's objective is to map each video into a trajectory that completes the demonstrated task.

4 VIEW

Our approach to imitation learning from video demonstrations relies on our intuition that efficient learning requires focusing on critical waypoints. Coming back to our motivating example of teaching a robot how to pick up a cup, the robot only needs to focus on the cup, how the human grasps it, and its movement throughout the video. The robot can complete the task by leveraging this information to guide its interaction with the environment. In this section we discuss our approach that consists of three main parts: first, extracting which object to pay attention to, how this object moves throughout the task, and the human's hand trajectory during the task. Second, designing a robust reward signal that compares the robot's behavior to the human's behavior. Third, exploring around the extracted waypoints in a sample-efficient manner. We refer to our method as **VIEW: V**isual **I**mitation l**E**arning with **W**aypoints. Refer to Figure 2 and Algorithm 1 for a summary.

Algorithm 1 VIEW

Input: Video of human interaction V
Residual network Φ
Dataset of previous corrections $\mathcal{D}$
Output: Successful robot trajectory ξ^r
Updated Residual network Φ

$\xi^h = \textsc{ExtractHandTraj}(V)$
ξ^o, $tag = \textsc{ExtractObjectTraj}(V, \xi^h)$
$\xi^h_c = \xi^h + \Phi(\xi^h)$
ξ^h_{grasp}, $\xi^h_{task} = \textsc{DivideTraj}(\xi^h_c)$
$\xi^*_{grasp} = \textsc{GraspExplore}(\xi^h_{task}, \xi^o, tag)$
if ξ^*_{grasp} is success **then**
  $\xi^*_{task} = \textsc{TaskExplore}(\xi^h_{task}, \xi^*_{grasp}, \xi^o, tag)$
  if ξ^*_{task} is success **then**
    $\xi^* = \textsc{CombineTraj}(\xi^*_{grasp}, \xi^*_{task})$
    Add (ξ^h , ξ^*) to $\mathcal{D}$
    Retrain Φ on $\mathcal{D}$
  end if
end if

4.1 Prior Extraction

VIEW relies on three crucial pieces of information extracted from the human's video demonstration: identification of the manipulated object, understanding of the object's movement within the video, and the human's hand movements during the task. Consequently, the prior extraction process can be divided into two main subparts: one concerning the object and its motion in the video, and the other focusing on the human's interactions with that object. Therefore, we separate our prior extraction approach into two distinct components: Hand Trajectory Extraction and Object Trajectory Extraction. A summary of our overall prior extraction method can be found in Figure 3.

Hand Trajectory Extraction. Prior methods have extensively addressed the extraction of hand trajectories from video demonstrations, often leveraging open-source neural networks for this purpose [3, 75, 80]. In our approach (see Figure 3), we analyze each frame (v_t) in a video (V) using the 100 Days of Hands (100DOH) model [63]. This model helps us identify the hand's location and whether it is interacting with any objects via

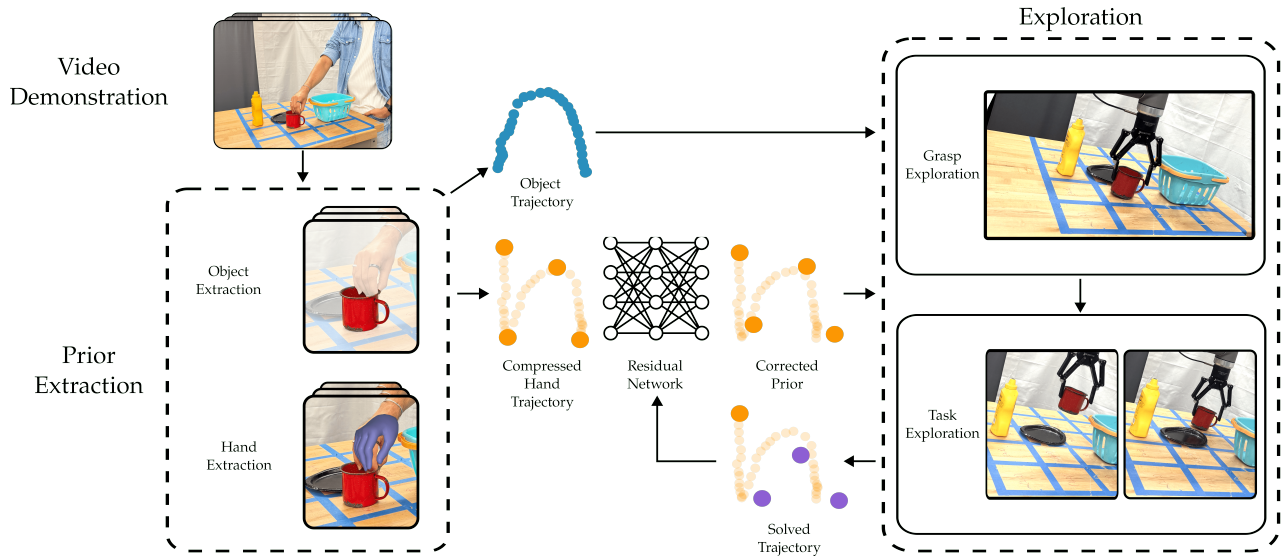

Fig. 2 Outline of VIEW, our proposed method for human-to-robot visual imitation learning. (Top Left) VIEW begins with a single video demonstration of a task. (Bottom Left) From this video we extract the object of interest, its trajectory, and the human's human trajectory. (Middle) We then perform compression to obtain a trajectory prior — a sequence of waypoints the robot arm should interpolate between to complete the task. Unfortunately, this initial trajectory is often imprecise due to the differences between human hands and robot grippers, as well as noise in the extraction process. We therefore refine the prior using a residual network, which is trained on previous tasks to de-noises the current data. (Right) The de-noised trajectory is then segmented into two phases: grasp exploration and task exploration. (Top Right) During grasp exploration, the robot determines how to pick up the object by modifying the pick point in its trajectory. (Bottom Right) Following a successful grasp, the robot proceeds to task exploration, where is simultaneously corrects the remaining waypoints of the trajectory. After completing exploration, the robot synthesizes a complete trajectory. (Middle) This solved trajectory, alongside the prior trajectory, is used to further train the residual network, thus enhancing the performance of our method in future tasks.

bounding box coordinates $(b^h_{x_t}, b^h_{y_t})$ and contact information (c_t). A bounding box alone can be ambiguous with respect to hand orientation and direction. To resolve this ambiguity, we further refine our coordinates with the MANO hand model [55] to pinpoint the human's wrist position $(p^h_{x_t}, p^h_{y_t})$. We convert the 2D image coordinates into 3D world coordinates (x^h_t, y^h_t, z^h_t) using depth information (δ_t) from the camera.

So far our methodology bears a strong resemblance to that used in WHIRL [3]. However, WHIRL overlooks a significant issue: the abundance of points along the extracted trajectory. For instance, a mere ten-second demonstration recorded at 60 frames per second yields a total of 600 frames. While this amount of visual and trajectory data appears substantial, upon closer inspection, much of is redundant. Returning to our example of picking up a cup, the video contains crucial waypoints such as the initial hand position and the cup's grasp position, but it also includes several redundant frames that interpolate between these key waypoints. This principle extends to various manipulation tasks; for instance, pouring tea into a cup necessitates waypoints depicting the start location, teapot grasp, pour location, and final orientation. All other intermediate points can be discarded.

To take advantage of this redundancy, we augment our extraction procedure with a trajectory compression algorithm. Numerous methods have been proposed for this purpose, including iterative [46] and dynamic programming approaches [67]. In our implementation we use Spatial Quality Simplification Heuristic - Extended (SQUISHE), a method that provides provable guarantees on trajectory error [46]. SQUISHE operates by minimizing the synchronized Euclidean distance of the trajectory. This metric calculates the distance between a waypoint in the trajectory and its interpolated counterpart. The interpolated counterpart is estimated using the position and velocity information from neighboring waypoints. If a waypoint's removal and interpolation result in an accurate estimation (i.e., we can ignore that waypoint while still maintaining the same trajectory shape), that point is pruned from the trajectory. Revisiting our cup example, if a linear movement exists between the hand's initial position and the cup grasp position, SQUISHE automatically removes all frames in between these key waypoints. By integrating SQUISHE with the extracted trajectory, we obtain a highly condensed and concise prior. This prior selectively retains waypoints corresponding to significant trajectory changes, including shifts in hand direction or alterations in hand-object contact patterns. With

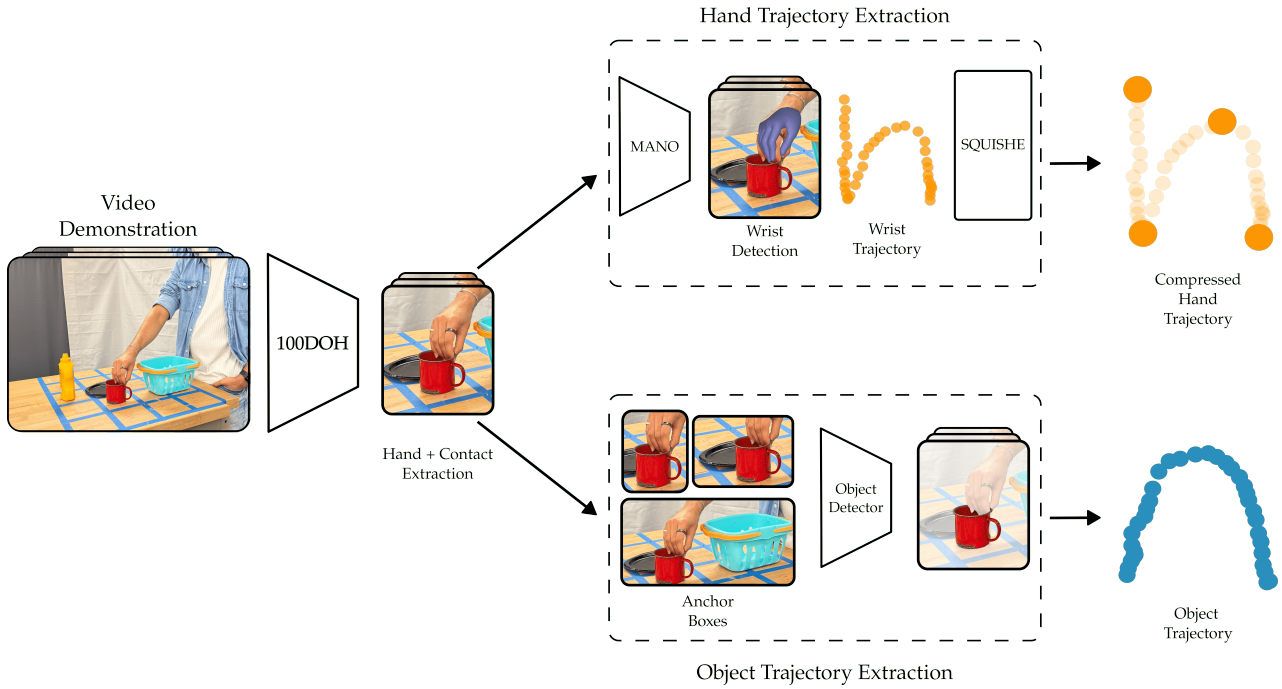

Fig. 3 An overview of our prior extraction method (Bottom Left in Figure 2). Utilizing the 100 Days of Hands (100DOH) detector [63], we first identify the location of the hand and if it is in contact with any objects present in the frame. We then refine the human's hand trajectory using the MANO model [55] to capture wrist movements. Subsequently, to eliminate redundancy, we apply the SQUISHE algorithm [46]. This produces an initial trajectory with key waypoints that the robot should interpolate between. To pinpoint the object of interest amidst potential clutter, we analyze frames where hand-object contact occurs, creating anchor boxes that — in conjunction with an object detector — reveal the object the human interacts with most frequently. This identification enables us to construct an accurate object trajectory from the human's video.

SQUISHE we often observe a substantial reduction in trajectory length — e.g., from over 300 points to just three or four waypoints.

Object Trajectory Extraction. Thus far, we have focused on extracting a concise prior trajectory from the human's hand movements. However, merely mimicking human actions is not sufficient for the robot to solve the task. In reality, the critical aspect of these demonstrations lies in understanding how the human is interacting with and moving objects. Revisiting our cup example, the focus should not be on repeating the human's hand movements; instead, the robot must learn to move the cup to the correct location.

To extract the object trajectory we start by identifying which object the human is manipulating. Here we capitalize on the human's hand interactions in the video demonstration. From the hand trajectory extraction, we know that the 100DOH model can indicate when the human's hand interacts with objects in a frame (c_t). Building on this insight, we initiate a process akin to anchor boxes for region proposal networks [54] in image detection algorithms. By generating anchor boxes of varying sizes around the hand and detecting objects within these boxes, we extract objects that are in close proximity to the human's hand at points of interaction. While there may be frames where the human hand is in proximity to multiple objects, we hypothesize that the majority of the frames will solely contain the human's intended object. Therefore, by finding the object that most commonly appears in contact frames, we can accurately identify the intended object (tag).

Once the object of interest is identified, we proceed to generate its trajectory (ζ_h^o) in the video demonstration. Utilizing our object detector, we generate bounding boxes around the object's location for all frames in the video. With the object's location obtained in pixel coordinates ($p^o_{x_t}, p^o_{y_t}$), we apply the same de-projection technique used in hand trajectory extraction and use the depth information (δ_t) to translate two-dimensional image frame coordinates into three-dimensional world coordinates (x_t^o, y_t^o, z_t^o). This process not only identifies the manipulated object but also delineates its trajectory throughout the video. Refer to Figure 3 and Algorithm 2 for a summary.

Overall our extracted prior provides us with three key pieces of information: a condensed trajectory representing the human's visited waypoints (ξ^h), a label identifying the object of interest (tag), and a trajectory indicating the object's movement (ξ_h^o).

Algorithm 2 Object Trajectory Extraction

Input: Video of human interaction V
Depth information D^h
Human hand trajectory ξ^h
Object detection model OD
Set of Anchor boxes A
Camera intrinsic and extrinsic parameters $\mathcal{C}_r^c$
Output: Object tag
Object trajectory in pixel space ζ_h^o
Object trajectory in 3D space ξ_h^o

Initialize OD
Initialize object count
for contact information c_t in ξ^h if c_t = True **do**
  v_t = ExtractVideoFrame(V, c_t)
  for Anchor box α in A **do**
    objects = OD(α)
    Update object count
  end for
end for
tag = Max(object count)
ξ_h^o, ζ_h^o = ExtractObjectTraj(V ,tag)
return ξ_h^o, ζ_h^o, tag

function ExtractObjectTraj(V, tag)
  Initialize an empty lists ξ_h, ζ_h
  for Video frame v_t in V **do**
    $p_{x_t}^o$, $p_{y_t}^o$ = $OD(v_t, tag)$
    Append $(p_{x_t}^o, p_{y_t}^o)$ to ζ_h
    $\delta_t^o = D^h(p_{x_t}^o, p_{y_t}^o)$
    $x_t^o, y_t^o, z_t^o = \mathcal{C}_r^c(p_{x_t}^o, p_{y_t}^o, \delta_t^o)$
    Append (x_t^o, y_t^o, z_t^o) to ξ_i^o
  end for
  return ξ_h, ζ_h
end function

4.2 Agent-Agnostic Rewards

After we get the human's hand trajectory (ξ^h) from the video demonstration, the robot executes this trajectory in the environment to try and solve the task (i.e., the human's hand trajectory becomes the robot's initial trajectory). However, this trajectory almost always fails because of morphological differences and sensor noise. In order to improve the initial trajectory over repeated interactions, the robot explores around ξ^h to find the correct waypoints that solve the task (see Figure 2). We will describe this exploration in detail in Section 4.3. But before we get to the exploration, we first need a feedback mechanism that allows the robot to differentiate between "good" and "bad" waypoints. More specifically, we design a reward model that compares how the robot is manipulating the target object to how the human manipulated the same object during their video demonstration.

Our prior from Section 4.1 contains the *tag* identifying the target object and its trajectory throughout the demonstration video. Similarly, we can take videos of the robot's interactions in the environment and extract the actual trajectory of the target object using the same procedure. To compare the movement of the object for the two agents, we take the mean square error (MSE) between the corresponding waypoints in their respective trajectories. The negative of this distance serves as our reward. For clarity, let the object trajectory from the prior ξ_h^o consist of waypoints $(p_{x_t}^h, p_{y_t}^h)$ and the object trajectory from the robot interaction ξ_r^o consist of waypoints $(p_{x_t}^r, p_{y_t}^r)$. Then, the reward corresponding to each waypoint is given as:

$$r_t = - \mid\mid \omega_t^r - \omega_t^h \mid\mid \tag{1}$$

$$\omega_t = (p_{x_t}, p_{y_t}) \tag{2}$$

We note that we measure the distance in pixels to mitigate any inaccuracies caused by transforming from the camera coordinate frame to the robot coordinate frame. Intuitively, our reward procedure is *agent-agnostic* because it does not matter who is manipulating the objects — either human or robot. We extract the object trajectories across videos from both agents, and then contrast those trajectories to quantify how similar the robot's behavior is to the human's behavior.

4.3 Exploration for Iterative Improvement

Now that we have a metric for quantifying the robot's performance, we are ready to iteratively improve the robot's trajectory. Referring back to Figure 2, the robot starts by executing its initial trajectory extracted from the human's hand movement, and then gradually improves this trajectory by exploring around the trajectory waypoints. A typical task entails grasping an object and then manipulating that object in the same way as the human. However, the robot cannot learn about this manipulation until it has learned how to grasp the object: the successful completion of the task is contingent on the robot moving the correct object. In our running example of teaching the robot to pick up a cup, the robot cannot succeed if it grabs the wrong object (e.g., picks up a plate), or if the robot does not grasp the object securely (e.g., drops the cup). We therefore divide the overall exploration into two parts: *grasp*, where the robot finds the grasp location for the object, and

task, where the robot learns to imitate how the human manipulates that object.

More formally, the initial trajectory extracted from the human's video consists of a set of n waypoints $\xi^h = \{\omega^h_t \mid t \in [t_1, t_2, \ldots, t_n]\}$ where each waypoint is a position and contact tuple (x, y, z, c). Here x, y, z indicate the 3D position of the hand and c indicates if the hand is in contact with any objects. From this contact information we can determine when the human grasps and releases objects. For instance, let the waypoint where the contact begins be ω^h_{grasp}. We use this point to divide the prior into two trajectories: $\xi^h_{grasp} = \{\omega^h_{t_1}, \ldots, \omega^h_{grasp+1}\}$ and $\xi^h_{task} = \{\omega^h_{grasp+1}, \ldots, \omega^h_{t_n}\}$. Under VIEW, the robot separately explores around these two trajectories to pick up the object and then perform the task. Below we discuss the robot's exploration strategies for iteratively improving *grasp* and *task*.

4.3.1 Correcting the Grasp Waypoint

In our first phase the robot explores around the prior $\xi^h_{grasp} = \{\omega^h_{t_1}, \ldots, \omega^h_{grasp}, \omega^h_{grasp+1}\}$. Although this prior contains multiple waypoints, the primary one the robot needs to focus on is ω^h_{grasp}, the waypoint where it should grasp the target object. How the robot arm reaches for that object is irrelevant, so long as it is able to successfully pick up and hold the item. Accordingly, in VIEW the robot uses the position where the human grasped the object as a prior (i.e., ω^h_{grasp}), and then the robot intelligently explores around this prior to find a grasp location that is effective for the robot arm and gripper.

Restricting the Exploration Space. We start by defining a region around the waypoint of interest ω^h_{grasp} in terms of a bounding box $\mathcal{B}$. This region will serve as the area where the robot should explore. A naive approach to creating a bounding box would be to use a limit Δ and define the bounding box as $\omega^h_{grasp} - \Delta$ to $\omega^h_{grasp} + \Delta$. More explicitly, we could define a range in the robot's coordinates with limits Δ: from $(x - \Delta, y - \Delta, z - \Delta)$ to $(x + \Delta, y + \Delta, z + \Delta)$. This would create a bounding box with the waypoint ω^h_{grasp} at the center. However, such a bounding box may be unnecessarily large and span irrelevant parts of the robot workspace. In our running example of learning to pick up a cup, this bounding box could include part of the workspace which is farther away from the cup, as shown Figure 4 (Top). Instead, we propose to leverage the object position we extract from the video to bias our search space towards that object. This leads to a more compact bounding box and reduces the area where the robot needs to explore. More specifically, we create a bounding box $\mathcal{B}$ that circumscribes the waypoint ω^h_{grasp} and the object location ω^o_{grasp}: the line segment between these two points becomes the primary diagonal of $\mathcal{B}$. To account for any sensor or model inaccuracy, we extend the line segment on both ends by a limit Δ. Now $\mathcal{B}$ circumscribes the points $(\omega^o_{grasp} - \Delta\hat{j})$ and $(\omega^h_{grasp} + \Delta\hat{j})$, where $\hat{j}$ is a unit vector from ω^o_{grasp} to ω^h_{grasp}:

$$\hat{j} = \frac{\omega^h_{grasp} - \omega^o_{grasp}}{|| \, \omega^h_{grasp} - \omega^o_{grasp} \, ||} \tag{3}$$

To see an example of this bounding box refer to Figure 4 (Bottom). Intuitively, this bounding box is a more efficient search area because it is based on both the human's hand position and the estimated object position.

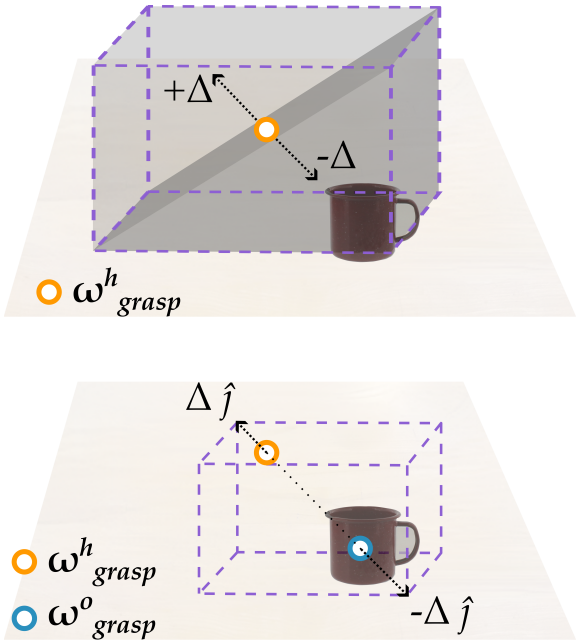

Fig. 4 Generating a bounding box for exploring grasp locations. We define a region around the waypoint $\omega^h_{grasp} = (x, y, z)$ where the human first interacted with the object in the video demonstration. (Top) A naive approach: the bounding box is centered around ω^h_{grasp} with limits Δ. The principal diagonal of the bounding box is defined by $(x - \Delta, y - \Delta, z - \Delta)$ and $(x + \Delta, y + \Delta, z + \Delta)$. (Bottom) Our approach that leverages the estimated object location ω^o_{grasp} at the time of grasping to bias the search space. The principal diagonal of the bounding box are $\omega^h_{grasp} + \Delta\hat{j}$ and $\omega^o_{grasp} - \Delta\hat{j}$, here $\hat{j}$ is the unit vector parallel to the principal diagonal and is calculated using Equation (3). This bounding box is typically smaller and is more likely to include an effective grasp location for the robot.

Rewards for Grasp Exploration. Now that the robot has an exploration region $\mathcal{B}$, the robot can move to different waypoints inside that region to try and grasp the object. However, when performing this exploration a key question emerges: how can we determine if the robot's grasp was successful? Merely observing the object's location at the moment of grasping is not suffi-

cient, as the object does not move from its initial location until it has been both grasped *and* moved.

Therefore, to assess whether the robot has grasped the desired item, we incorporate the subsequent waypoint ($\omega^h_{grasp+1}$) into our grasp exploration trajectory. This allows the robot to execute a grasp at the chosen location and then proceed to the next waypoint along its initial trajectory. Throughout each round of exploration in the environment the robot stores its end-effector location, and we use the video camera to track the *tagged* object using our detection model. Intuitively, we can confirm that the robot has successfully grasped the *tagged* object if the object is positioned close to the robot's end-effector at timestep $grasp+1$. This ensures that the grasp's effectiveness is measured not just by proximity, but also by whether the robot can hold and move the object.

Grasp Exploration. We have established where the robot should search and how to determine if a proposed waypoint has grasped the item. Our final step is developing a method for *exploring* the bounding box to identify an optimal grasp location. This search problem is complicated by two main challenges. First, for most waypoints the robot does not move the object and the rewards are constant. Put another way, we have *sparse rewards*. Second, if the robot reaches a waypoint that is close to the object it may accidentally knock the object over or otherwise *lower its measured rewards*. Hence, waypoints that are actually close to a successful grasp could be penalized by the reward model.

Returning to our cup example, consider a scenario where the robot receives a baseline reward of +10 at waypoints that don't involve moving the cup. If the cup is supposed to be moved left as per the human's video demonstration, and the robot picks a point that hits the cup and moves it to the right, the reward might drop to +5. Conversely, moving the cup correctly to the left might increase the reward to +15. In both cases, the robot has gained valuable information: the chosen waypoint interacted with the object and may be near to a successful grasp location. This variation in rewards — regardless of whether the reward is increasing or decreasing — helps to pinpoint the object's location within the search space $\mathcal{B}$.

Put together, the sparse rewards at grasp locations and locally varying rewards around those locations make it difficult for the robot to efficiently optimize for successful grasps. We therefore propose a quality-diversity (QD) approach for intelligently searching the space $\mathcal{B}$. Our proposed QD algorithm is broken into a *high-level* search — which divides $\mathcal{B}$ into regions of interest — and a *low-level* search — which explores within those regions to pinpoint a successful grasp location. We summarize this overall method in Algorithm 3.

High-Level Grasp Exploration. We first discretize the bounding box $\mathcal{B}$ into a set of regions for the robot to explore. To obtain these regions we use Centroidal Voronoi Tessellation (CVT) [74]: we numerically sample a large number of points inside $\mathcal{B}$, and then use k-means to group these sampled points into M clusters. In practice, these clusters provide M regions that are equally spread across the bounding box $\mathcal{B}$. Within our high-level exploration process the robot will determine which of these clusters are of interest — i.e., which clusters could contain a successful grasp location — for more targeted low-level optimization.

Let the centroids of the high-level clusters form a discrete set of potential waypoints: $\Omega_{unvisited} = \{\omega \quad \forall i = 1, 2, \dots M\}$. With no additional information to differentiate them, all centroids are considered *equally likely* grasp locations. A straightforward method to select waypoints would be to uniformly sample from this set of univisited centroids ($\Omega_{unvisited}$). However, random sampling from a uniform distribution may lead to waypoints that are close to previously tested centroids (see Figure 5 Top). We therefore propose a sampling strategy for selecting waypoints from $\Omega_{unvisited}$ that encourages the robot to visit previously unexplored parts of the bounding box $\mathcal{B}$.

Under our approach the robot reasons over previously attempted centroids to select a new high-level waypoint that is different from the ones it has already tested. This can be achieved by choosing a waypoint from the unvisited centroids ($\Omega_{univisted}$) that maximizes the distance from already visited waypoints ($\Omega_{visited}$). Mathematically, we optimize the selection of the next waypoint as follows:

$$\omega_{next} = \underset{\omega \in \Omega_{unvisited}}{\arg\max} \quad \mathcal{D}(\omega, \Omega_{visited}) \tag{4}$$

$$\mathcal{D}(\omega, \Omega_{visited}) = \frac{1}{k} \sum_{\omega_j \in N_i^k} || \, \omega - \omega_j \, || \tag{5}$$

Here $\mathcal{D}(\omega, \Omega_{visited})$ represents the mean distance between each unvisited waypoint in $\Omega_{unvisited}$ and all the waypoints in the visited set $\Omega_{visited}$. To tractably handle the potentially large number of centroids, we approximate this distance using the k-nearest neighbors (N_i^k) of each unvisited waypoint ω.

Unfortunately, only maximizing the mean distance to all visited waypoints introduces a potential issue: a waypoint might be very close to one visited waypoint but far from others, resulting in a high mean distance that does not truly reflect diversity (see Figure 5 Middle). To address this, we introduce a constraint to

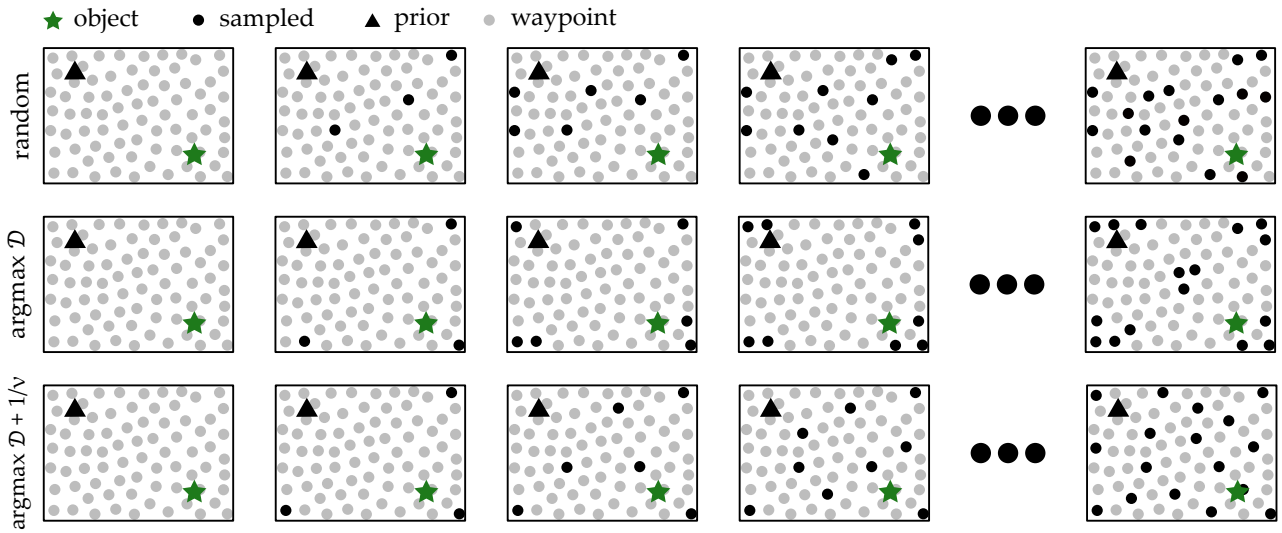

Fig. 5 Comparison of different sampling methods in our *high-level grasp exploration*. We show an example task in a two-dimensional space which is bounded around the prior (black triangle) and the object (green star). (Top) Each new high-level waypoint point is uniformly randomly sampled from our set of unvisited waypoints. This method can eventually reach the object with sufficient exploration. However, new samples my be close to previously tested points. (Middle) To quickly reduce the uncertainty about the unknown object location, we can sample high-level waypoints that maximize the distance to all previously visited waypoints. We expect that these waypoints will explore new regions of the search space. In practice, however, the distance-based estimation from Equation (5) results in points that are clustered at the corners and center. (Bottom) Our proposed solution is to add a regularizing term in Equation (6) to ensure that the next high-level waypoint is truly from an unexplored region of workspace. Our experiments show that this approach finds the grasp location more rapidly.

our optimization criteria, ensuring that the new sampled waypoint is equidistant from all the waypoints in $\Omega_{visited}$. This constraint acts as a regularizer in our objective. We now calculate the distances between each unvisited waypoint and all visited waypoints:

$$\Theta_i = || \ \omega_i - \omega_j \ || \quad \forall \omega_j \in \Omega_{visited}$$
$$\Theta = \{\Theta_i \quad \forall \omega_i \in \Omega_{unvisited}\}$$

We then compute the variance of these distances to enforce equidistance:

$$\nu(\Theta_i) = \frac{|| \ d_i^j - \bar{d}_i \ ||}{| \ \Theta_i \ |}$$
$$d_i^j = || \ \omega_i - \omega_j \ || \quad \forall \omega_j \in \Omega_{visited} \ \ \& \ \ \omega_i \in \Omega_{unvisited}$$
$$\nu(\Theta) = \{\nu(\Theta_i) \quad \forall \omega_i \in \Omega_{unvisited}\}$$

Our high-level exploration method, therefore, chooses waypoints that optimize the following:

$$\omega_{next} = \underset{\omega \in \Omega_{unvisited}}{\arg\max} \quad \mathcal{D}(\omega, \Omega_{visited}) + \frac{1}{\nu(\Theta)} \tag{6}$$

This optimization ensures that the selected waypoint from $\Omega_{unvisited}$ maximizes distance from all waypoints in the set while attempting to be equidistant with the waypoints in $\Omega_{visited}$ (see Figure 5 Bottom). In practice, the robot selects a high-level waypoint from $\Omega_{unvisited}$ using Equation (6), and then executes a trajectory in the environment that attempts to grasp the object at that waypoint. We use the reward model from Equation (1) to assess the performance of this grasp.

Low-Level Grasp Exploitation. So far we have established a method for the robot to divide the bounding box $\mathcal{B}$ into equally distributed waypoints. As the robot iteratively executes trajectories that reach these high-level waypoints, it obtains rewards using Equation (1). At visited waypoints where the measured reward varies — either increasing or decreasing — the robot may be close to an optimal grasp location. Accordingly, our next step is to employ a local, low-level search algorithm to explore the regions around promising high-level waypoints, and fine-tune these waypoints to eventually perform a successful grasp.

Our first step in this process is determining which of the high-level waypoints the robot should explore around. Intuitively, we are interested in waypoints where the robot's measured reward has varied, since at these waypoints the robot must be interacting in some way with the target object. We therefore sample from the visited high-level waypoints in proportion to how much the robot's reward has varied at these waypoints. More specifically, we sample a waypoint ω_{local} from $\Omega_{visited}$ with the probability distribution:

$$p_i = \frac{e^{\gamma \sigma_i}}{\sum e^{\gamma \sigma_j}} \tag{7}$$

Algorithm 3 Grasp Exploration

Input: Prior trajectory of grasping ξ^h_{grasp}
object location in robot coordinates ω^o

Initialize $\delta, \epsilon, p_{explore}$
Initialize flag $local = False$
Initialize an empty list $\Omega^r_{visited}$
Get the point where human grasps the object ω^h_{grasp} from prior ξ^h_{grasp}
Calculate unit vector using Equation (3)
Define bounding box $\mathcal{B}$ that circumscribes the points $\omega^o_{grasp} - \Delta\hat{j}$ and $\omega^h_{grasp} + \Delta\hat{j}$
Sample random *points* from $\mathcal{B}$ and initialize the set $\Omega^r_{unvisited} =$ K-Means$(points)$
Initialize Bayesian optimizer BO

function Ask
    Generate p from uniform distribution $[0, 1]$
    if $p < p_{explore}$ **then**
        Sample high-level waypoint ω^r from $\Omega^r_{unvisited}$ using Equation (6)
        Change flag $local = False$
        return ω^r
    else
        Sample high-level waypoint from $\Omega^r_{visited}$ with probability distribution from Equation (7)
        Start low-level search by defining a bounding box around this waypoint with limits ϵ
        Query BO for ω^r
        Change flag $local = True$
        return ω^r
    end if
end function

function Tell(ω^r_i, R_i)
    if *local* **then**
        Update BO with ω^r_i, R_i
    else
        Remove ω^r_i from $\Omega^r_{unvisited}$
        Add (ω^r_i, R_i) to $\Omega^r_{visited}$
    end if
end function

while *grasp* is not successful **do**
    Sample a waypoint $\omega^r =$ Ask
    Execute trajectory $\xi^r = \{\omega^r_{t_1}, \omega^r_{t_2}, \ldots \omega^r, \omega^r_{grasp+1}\}$
    Get the reward R using Equation (1)
    Inform the explorer Tell(ω^r, R)
end while

where the denominator is summed across all visited waypoints, and σ_i is the normalized variation in reward between the high-level waypoint i and the reward R_0 from the initial trajectory:

$$\sigma_i = \frac{|| R_i - R_0 ||}{\max_j || R_j - R_0 ||} \tag{8}$$

In practice, using Equation (7) causes the robot to bias its low-level search towards the high-level waypoints that produced the largest changes in reward.

Once the high-level waypoint the robot wants to explore is selected, our next step is leveraging a local search procedure to identify the optimal grasp location within the region around that waypoint. Note that during the high-level search we are looking for regions that cause changes in reward; by contrast, within this low-level search we are purely trying to maximize the robot's reward. Here we can use existing optimization algorithms such as Bayesian optimization (BO)[1] [70]. We start by defining a smaller bounding box $\mathcal{B}_{local} \subset \mathcal{B}$ around the sampled high-level waypoint ω_{local} with distance ϵ. This bounding box covers the region of interest around centroid ω_{local}. BO then optimizes the reward function within this region by sampling waypoints ω_{opt} from $\mathcal{B}_{local}$. If the sampled waypoint gets a higher reward than ω_{local}, we substitute ω_{opt} into $\Omega_{visited}$.

Trading-off Between High- and Low-Level Search. Our overall exploration process for identifying a successful grasp trades-off between testing new high-level waypoints from $\Omega_{unvisited}$ and then exploiting the regions around relevant waypoints from $\Omega_{visited}$. We balance this exploration of new regions and exploitation of sampled regions using probability $p_{explore}$. Looking at Algorithm 3, with probability $p_{explore}$ we test an $\Omega_{unvisited}$ waypoint, and rollout a trajectory in the environment that includes that waypoint. Similarly, with probability $1-p_{explore}$ the robot executes a trajectory that explores the region around a waypoint from $\Omega_{visited}$. This search process ends once the robot identifies a waypoint that successfully grasps the target item.

In summary, grasp exploration works in a hierarchical manner. First the robot conducts a broad search across the bounding box $\mathcal{B}$ by dividing it into M evenly distributed high-level waypoints. The robot then conducts a more refined search in the vicinity of waypoints that are potentially close to the object — i.e., waypoints that incur a high variation in reward. Overall, our grasp exploration approach has similarities to the QD algorithm CMA-ME [16]. The primary novelty of

[1] VIEW is not tied to a specific local optimization algorithm. While we use BO in our experiments, it can be replaced with any other optimizer.

our approach as compared to [16] is the sampling technique used for selecting the high-level waypoints. While CMA-ME relies on randomness to select a point from $\Omega_{unvisited}$, we propose a regularized entropy metric for selecting points that are evenly spaced across the search space. In Figure 5 we show an example of why this high-level sampling approach is important, and how our proposed approach can more rapidly identify the grasp location. We also test this difference in our experiments.

4.3.2 Correcting the Task Waypoints

Once the robot has grasped the target object, it can now proceed to replicate how the human manipulated that object in the demonstration video. This process is more straightforward than identifying the correct grasp because here the rewards are *dense*: any change in the way the robot moves the object will lead to a change in the object's position, and therefore a change in the measured rewards from Equation (1). Accordingly, we can use off-the-shelf optimization methods to iteratively improve the waypoints along the initial trajectory $\xi^h_{task} = \{\omega^h_{grasp+1}, \ldots, \omega^h_{t_n}\}$ after the robot has learned to successfully grasp the target item.

Similar to our approach for grasp optimization, we start by drawing bounding boxes $\mathcal{B}$ around each waypoint in ξ^h_{task}. Here it is important to remember that the reward function from Equation (1) is the distance between the object position in the human's video demonstration and the object position in the robot's task execution. Hence, the rewards associated with each waypoint are independent, and the robot can simultaneously explore and improve each task waypoint without affecting the results across other task waypoints. We therefore conduct $n-grasp$ search processes in *parallel*, one for each waypoint from $\omega^h_{grasp+1}$ to the final waypoint $\omega^h_{t_n}$. Let us denote the robot's updated trajectory as $\xi^r_{task} = \{\omega^r_{grasp+1}, \ldots, \omega^r_{t_n}\}$. To find an optimal waypoint ω_r, the samples a point within the corresponding bounding box and then rolls-out a trajectory that moves through that point in the environment. Here we use Bayesian optimization (although other methods are possible): for each $\omega^r_t \in \xi^r_{task}$, a separate instance of BO updates the robot's waypoint to better match the video demonstration. See Algorithm 4 for a summary.

Algorithm 4 Task Exploration

Input: Prior trajectory of task ξ^h_{task}

Define bounding box for each waypoint $\omega^r \in \xi^r_{task}$
Initialize a separate Bayesian optimizer BO for each waypoint in task

function ASK
  Initialize an empty list ξ^r_{task}
  for $\omega^h_i \in \xi^h_{task}$ **do**
    Query BO for ω^r_i
    Add ω^r_i to ξ^r_{task}
  end for
  return ξ^r_{task}
end function

function TELL(ξ^r_{task}, R)
  for $i = 1, \ldots, n$ **do**
    Update corresponding BO with $\omega^r_i \in \xi^r_{task}, R_i \in R$
  end for
end function

while *task* is not successful **do**
  Sample trajectory ξ^r_{task} = ASK
  Execute trajectory ξ^r_{task} in environment
  Get the reward R for each waypoint in the trajectory using Equation (1)
  Inform the explorer TELL(ξ^r_{task}, R)
end while

4.4 Residual Network

The process we have described so far in Section 4 enables the robot to learn a task from a *single* video. This spans compressing the video to extract a prior trajectory and reward function, as well as exploring those trajectory waypoints to improve the object grasping and manipulation. But when the robot gets a new video demonstration for a different task, we are faced with the question: does the robot need to restart VIEW from scratch, or can the robot leverage what it has learned on one task to accelerate learning on another task? Here we return to our example of learning to pick up a cup. Initially the robot is given a video of the human performing the task, which the robot compresses to extract an initial trajectory ξ^h. This trajectory is often wrong (e.g., misses the cup), and so the robot iteratively improves that trajectory during exploration to eventually reach a successful trajectory ξ^* (that grasps and lifts the cup). At this point we can compare the initial trajectory ξ^h to the final trajectory ξ^* — instead of extracting ξ^h, an ideal robot should have extracted ξ^*.

We propose to use this error between initial and final trajectories to improve the accuracy of the robot's prior. Our underlying hypothesis is that the sources of error are *constant* between tasks: e.g., any sensor inaccuracies, model misalignment, or morphological differences are approximately constant from one video demonstration to another. More formally, we treat these errors as

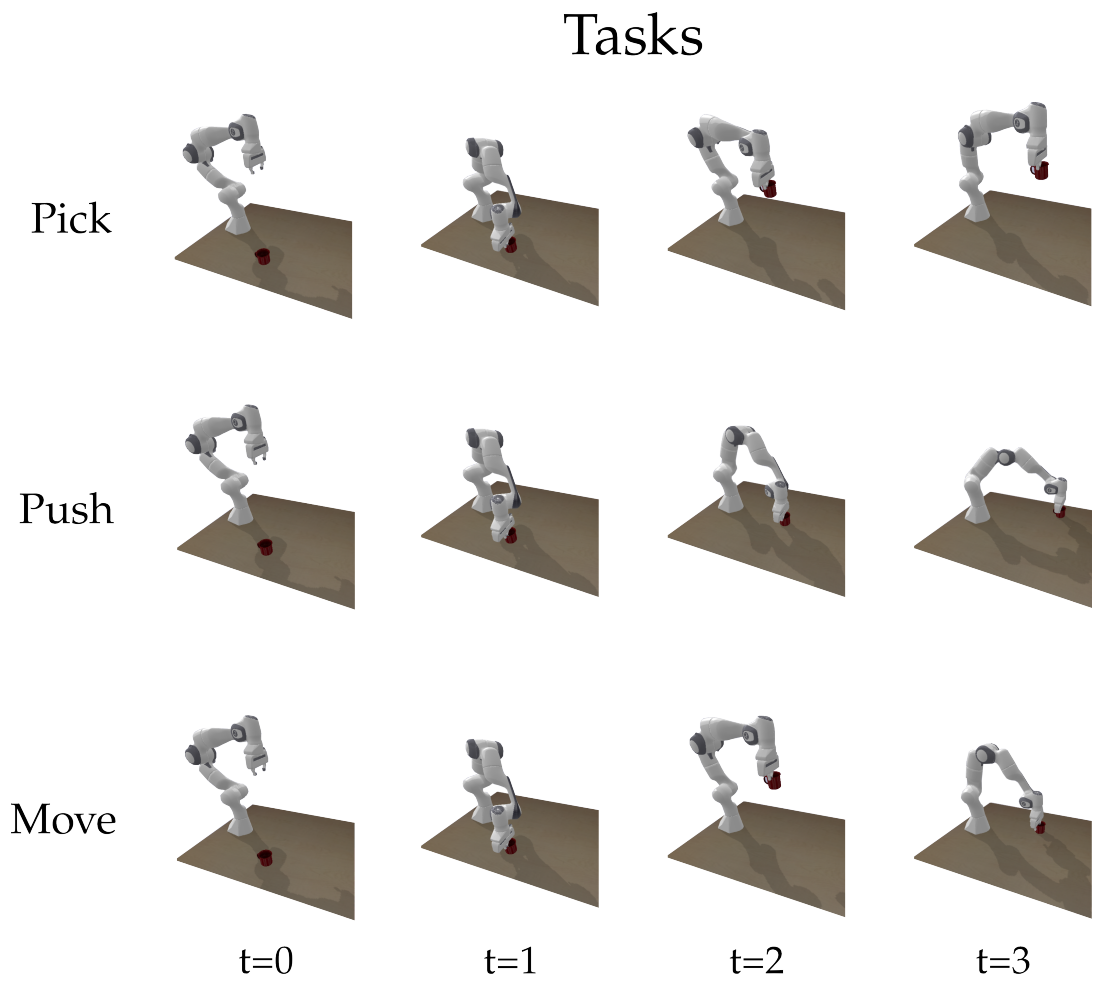

Fig. 6 Task demonstrations used in our simulations. (Top) During **Pick** the robot is teleoperated to pick up a cup placed on the table. (Middle) During **Push** the robot is teleoperated to grasp the cup and push it to a new specified location on the table. (Bottom) During **Move** the robot picks up the cup and places it at a specified new location on the table. In our ablation studies examining the effects of noise, trajectory compression, and exploration techniques, we utilize a single demonstration that is then perturbed using Gaussian noise. For assessing the influence of residual learning in our final simulation, we uniformly sample start and end points for each task and collect teleoperated demonstrations accordingly. These demonstrations are then perturbed using a deterministic noise function. We compile a dataset of 50 demonstrations for each task, either by introducing Gaussian noise to a single trajectory or by generating 50 distinct trajectories through uniform sampling.

an additive noise, so that the final trajectory ξ^* is equal to the initial trajectory plus this error: $\xi^* = \xi^h + \eta$. To de-noise the prior extraction process we propose to train a residual network across the data from previously solved tasks. Given a dataset of k previously solved tasks $\mathcal{D} = (\xi_k^h, \xi_k^*)$, we train a model to estimate the noise η. More specifically, we train a residual $\Phi(\xi^h) = \eta$ to minimize the loss $\|\xi^* - \xi^h + \Phi(\xi)\|^2$ across the dataset. The robot then deploys this residual when it receives the $k+1$ video demonstration. The robot starts by compressing the new video using the steps from Section 4.1 to get ξ_{k+1}^h; we then add the residual $\Phi(\xi_{k+1}^h)$ to push this prior towards the correct trajectory. In practice, we will show that the residual can improve the accuracy of the prior and reduce the number of iterations the robot needs to learn new tasks.

4.5 Incorporating Multi-Object Scenarios

Our discussions thus far have dealt with scenarios where the human interacts with a single object. However, many real-world manipulation tasks involve handling multiple objects. For example, when making tea the human might carry a cup to a specific location, and then pick up and pour tea from a kettle into the cup.

The proposed method VIEW readily adapts to such multi-object scenarios. Recall that our prior extraction process outputs a hand trajectory, providing the wrist location and contact information throughout the video.

We can use the changes in the contact information to divide long trajectories — involving multiple objects — into distinct sub-trajectories for each subtask. Each subtask then involves interaction with only one object, which we can solve using the algorithms described above.

Consider the example of making tea. By segmenting the trajectory at points where contact changes, we can create separate, manageable segments: one for moving the cup and another for pouring the tea. Each subtask is then solved separately using our algorithm, which includes dividing each individual subtask into grasp and task phases and solving them using our methods in Section 4.3.1 and Section 4.3.2. For example, we first address the cup's movement, and once complete, proceed to handle the kettle in a similar manner [2].Overall, this modular strategy allows the robot to systematically learn long, multi-step tasks with visual imitation learning by concentrating on one subtask at a time.

5 Simulations

We proposed VIEW, a waypoint-based algorithm that can imitate humans by watching video demonstrations. Our algorithm is comprised of three main parts: extracting a useful prior, exploring around this prior, and learning a residual from previously solved tasks. We hypothesize that each of these components will significantly impact the overall success of the robot. To test this hypothesis, in this section we conduct an ablation study that investigates how each part of VIEW contributes to the overall robot performance. We perform these experiments using a simulated robot arm and a simulated human demonstrator.

Experimental Setup. The simulations are conducted in a Pybullet environment. To collect demonstrations, we move a simulated FrankaEmika robot arm and record frames at the rate of 20Hz. Demonstrations are collected for three tasks: picking up an object (**pick**), pushing an object (**push**), and picking and placing an object (**move**). A single object (a cup) is used for all evaluations (See Figure 6). To condense trajectories for each task, the robot executes the demonstration in the simulated environment and then compresses the demonstration to extract an initial trajectory (see Section 4.1). For the push and pick tasks, this initial trajectory has three waypoints after compression, while the move task yields four waypoints. To simulate real world conditions, we distort these initial trajectories using either Gaussian noise or a fixed noise matrix. The robot then follows the exploration procedure laid out in Section 4.3

[2] See https://collab.me.vt.edu/view/ for videos showcasing VIEW learning these multi-object tasks.

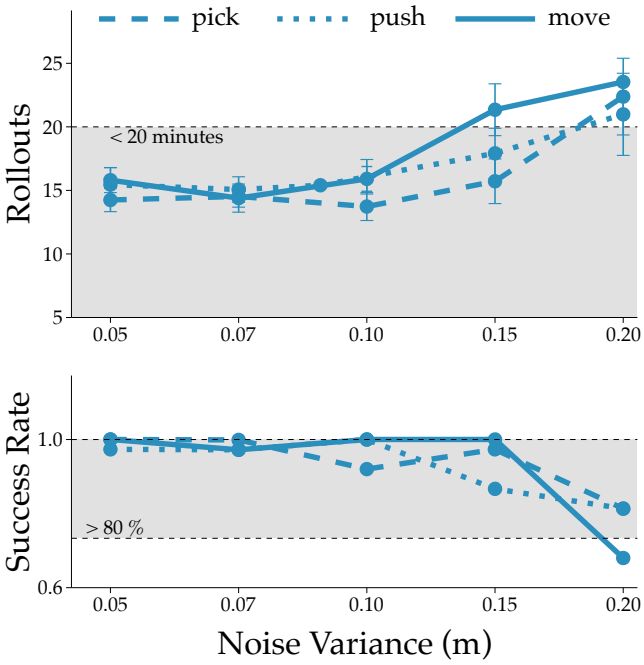

Fig. 7 Simulation results demonstrating the impact of noise on our algorithm. We test VIEW on three tasks — pick, push, move. For each task we collect the true initial trajectory, and then add Gaussian noise to distort that trajectory. This captures scenarios where the robot's prior is incorrect (e.g., misses the cup entirely), and the robot must explore around this prior to imitate the demonstrated task. Our results are shown across 50 trials. The shaded region in the top plot indicates less than 20 minutes of learning time to successfully imitate the task. The shaded region in the bottom plot indicates more than 80% success rate. The bars indicate standard error of the mean.

to explore around the distorted trajectory and tries to solve the task. To understand our algorithm's performance in ideal conditions, no noise is injected into the reward function. The success criteria vary slightly between tasks: for **pick**, success means the robot has picked up the cup; for **push**, it is successful if it pushes the cup to the correct location; and for **move**, success requires the robot to pick up the cup and place it at the correct location on the table.

5.1 Impact of Noise

In our first simulation, we study how noise in the extracted prior influences our algorithm's capability to converge to the correct behavior. Here increasing noise means that the robot's extraction of the human's hand trajectory is farther from the actual trajectory that the human followed in their video. Given the variable nature of deviations between the prior and ground truth — which can be significant and unpredictable depend-

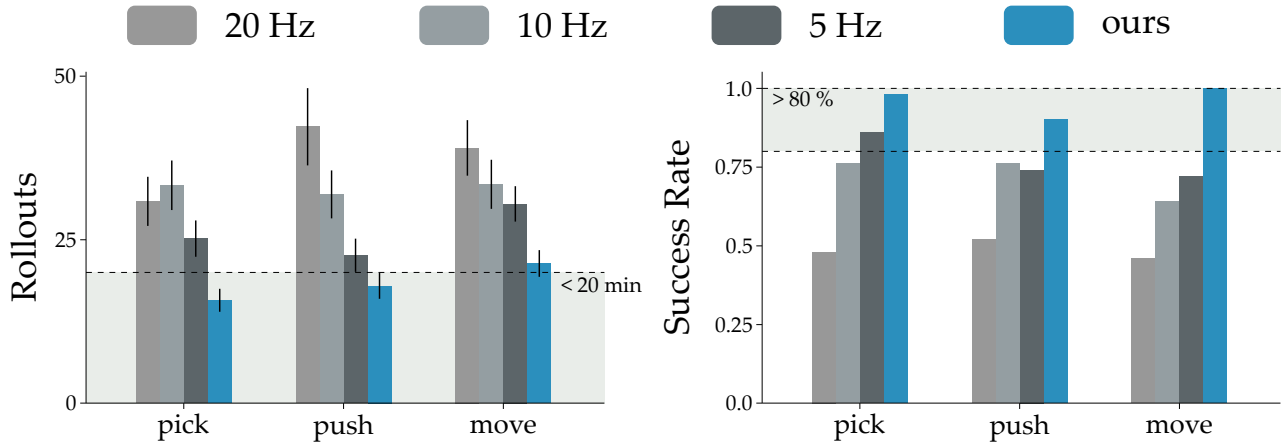

Fig. 8 Simulation results examining the impact of trajectory compression. Within VIEW we compress the prior trajectory to minimize the number of waypoints while maximizing the accuracy of the compressed trajectory. We compare this method with an alternative approach in which the prior is sampled at a lower frequency to limit the number of points in the trajectory. We vary the sampling frequency of the prior trajectory to be 5Hz, 10Hz, or 20Hz. The plot on the left shows the average number of rollouts it takes to learn each task over 50 trials, and the shaded region indicates less than 20 minutes of training time. The plot on the right shows the success rate for each task across 50 trials, and the shaded region shows a success rate higher than 80%. The bars indicate standard error of the mean.

ing on the physical setup — we want to ensure that our exploration strategy remains effective even in the presence of an incorrect initial trajectory.

To simulate an incorrect prior, we distort the correct initial trajectory by adding Gaussian noise. We analyze the impact of this noise across all three tasks: **pick**, **push**, and **move**. For each task we carry out a series of 50 trials: at each trial, we apply distortions sampled using Gaussian noise to the collected demonstration. These distorted demonstrations then become the initial trajectory ξ^h that our robot needs to correct to solve the task. Since this simulation is designed to isolate the impact of noise on our exploration scheme, we do not include the residual network during these trials.

Our findings (refer to Figure 7) reveal that VIEW can use exploration to overcome an incorrect prior. Specifically, for noise variance between 0.05m and 0.15m, the robot is able to successfully imitate the simulated human demonstration in almost 100% of the trials. However, as the prior is distorted farther away from the correct trajectory, the performance of VIEW eventually decreases. At a noise variance of 0.2m we observed a notable decrease in the success rate. This observation aligns with our expectation that larger distortions lead to longer search times, potentially resulting in timeouts before solutions are found. This pattern is also evident in the number of exploration rollouts required for convergence: in general, the more noise in the prior the more exploration rollouts the robot needed to correct its waypoints. Finally, we note that the number of waypoints can impact performance: tasks involving more waypoints (**move**) generally required more rollouts than tasks with fewer waypoints (**pick** and **push**). In practice, this simulation suggests that VIEW's exploration steps are critical to success, and the robot can use exploration to overcome errors in its initial guess of the correct trajectory.

5.2 Impact of Trajectory Compression

In our second simulation we assess the importance of trajectory compression within our algorithm. In Section 4.1 we outlined an optimization approach for identifying a small set of waypoints that capture the human's demonstration. However, simpler methods for compression are also possible: for instance, we could simply sample the demonstration at a reduced rate, and use the sampled points as the initial trajectory (e.g., downsample the video every 100 frames). Here we compare the impact of this alternative method against our compression algorithm.

To generate these alternative compressions, we first distort the correct initial trajectory using a noise variance of 0.15m. We then resample this modified demonstration at a fixed sample rate. We tested sampling rates from 20Hz to 5Hz. Our original demonstrations comprised approximately 40 waypoints: hence, the compression could range from 40 waypoints at the highest sampling rate to 10 waypoints at the lowest sampling rate. We did not sample lower than 10 waypoints using a fixed sampling rate because this caused the trajectory to skip critical waypoints, such as the pick point. Similar to the previous subsection, we assessed the impact

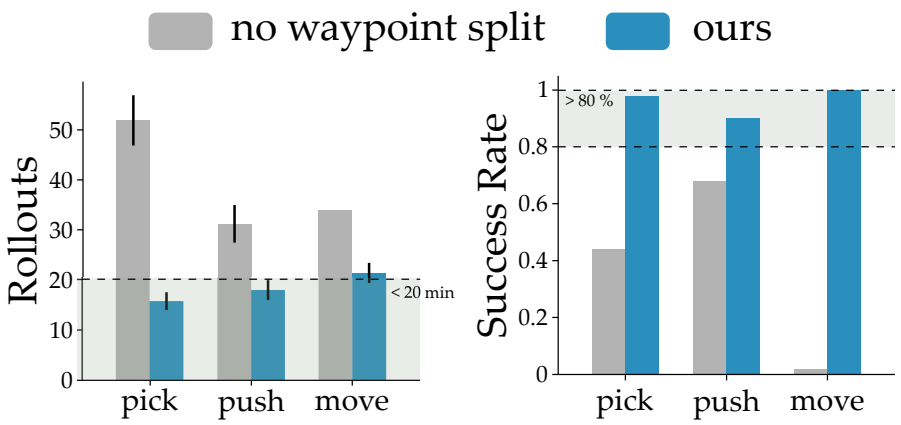

Fig. 9 Simulation results examining how separating the waypoints into grasping and manipulation phases affects performance. Under VIEW the robot autonomously splits the task into separate parts: first the robot learns to grasp the object, and then it learns how to manipulate that object and complete the task. We compare this division against a unified approach that solves the entire task simultaneously. We measure the average number of rollouts taken to solve the task (Left) and the success rate (Right) over 50 trials. The shaded regions indicate less than 20 minutes of training time and over 80% success rate, respectively. The bars indicate standard error of the mean.

of compression using the success rate and the number of rollouts required for convergence across 50 trials.

Our results, depicted in Figure 8, provide two important outcomes. First, as the number of waypoints in the robot's trajectory increases (higher sampling rate), the number of rollouts required for convergence also rises. This suggests that compression is indeed important — we can accelerated the robot's visual imitation learning by focusing on a smaller number of waypoints. Second, using simplistic compression algorithms that down-sample the demonstration at a fixed rate perform worse than our VIEW approach. The key difference here is that sampling at a fixed rate may cause the robot to miss a critical point along the demonstration (such as the frame where the human grasps the cup). Using VIEW, the robot minimizes the number of waypoints, while also ensuring that those waypoints retain critical aspects of the demonstration.

5.3 Impact of our Exploration Approach

In our third simulation we delve into the choice of exploration strategies. In particular, we study whether splitting the task into separate parts for grasping and manipulation is necessary. In Section 4.3 we developed separate exploration schemes for each of these phases, with the rationale that the robot must first learn to grasp the object before it can imitate the rest of the human's demonstration. While we discussed the reasons behind this approach in Section 4.3, we now aim to empirically assess its effectiveness.

As an alternative to our proposed method of task segmentation, we examine the performance of a unified optimization approach. This baseline does not separate the task into grasp and manipulation phases; instead, it performs Bayesian Optimization [70] to de-noise the *entire* trajectory. We maintain the noise level at 0.15m, and evaluate success rates and convergence rollouts for all 50 trials, similar to our previous simulations.

The outcomes of this experiment are depicted in Figure 9. As anticipated, we observe a substantial reduction in success rates when the waypoints are not split into grasping and manipulation phases. The highest success rate achieved without splitting is approximately 70% for the **push** task, whereas our segmented approach with VIEW achieves a minimum of 92% on the same task. VIEW also decreases the number of environmental rollouts required for convergence. Finally, we noticed that these results are impacted by the number of waypoints in the trajectory. For instance, in the **move** task — which has four waypoints instead of the three in **pick** and **push** — the success rate of the baseline approaches zero. These results indicate that forgoing waypoint segmentation during exploration not only leads to inferior performance, but also fails to scale effectively with an increasing number of waypoints.

5.4 Impact of Residual

In our final simulation, we move beyond learning a single task, and explore how VIEW performs across multiple tasks. Specifically, we test how the residual network from Section 4.4 can accelerate the robot's learning on one task given that the robot has previously solved other tasks in the same environment. We con-

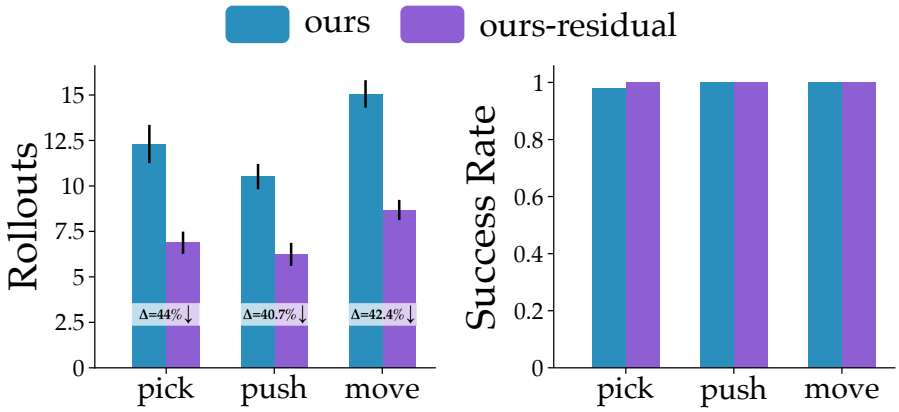

Fig. 10 Simulation results for VIEW with and without the residual. We examine if the robot can utilize previous experiences to more rapidly imitate new tasks. In this simulation we sample 50 random locations from the robot's workspace and their corresponding distortions from a noise matrix (Equation (9)). We then use these samples to train a residual network that denoises the distorted prior. The plots above compare the performance of our approach with and without the usingthis residual. (Left) The number of rollouts taken to solve the task averaged over 50 trials. We also list the percentage decrease in rollouts when the residual is present. (Right) The success rate for each task. The bars indicate standard error of the mean.

trast this to our previous simulations, where each new task or demonstration was approached from scratch.

In the previous simulations we employed Gaussian noise and sampled various distortions for a fixed cup location. This is not feasible here because repeatedly sampling from a Gaussian distribution would cause the environmental noise to be inconsistent. Put another way, a given xyz coordinate could be distorted in different ways between each task; this inconsistency would not match realistic conditions. Instead, real-world noise factors — such as the de-projection inaccuracies and morphological differences — impose a consistent offset at each waypoint. To better simulate these real-world conditions, we introduce a nonlinear noise matrix to distort the robot's entire workspace:

$$\eta = \tanh \frac{\xi - \mathcal{C}}{\lambda} \tag{9}$$

We utilize tanh to introduce distortions into the trajectory waypoints, adjusting their positions based on their proximity to a centroid ($\mathcal{C}$). The degree of distortion is modulated by the regularizer λ. This noise is then added to the demonstration to get a distorted initial trajectory ξ^h. We adjust the location of $\mathcal{C}$ and the value of λ to ensure that the distortions range from 4cm to 30cm across all waypoints. To mitigate against any bias, we do not use a fixed cup location; instead, we sample the cup's location from a uniform distribution across the table, and then collect demonstrations for each task: **pick**, **push**, and **move**. We gather a total of 50 random demonstrations from the environment, each distorted via the noise matrix, to form our dataset. Specifically, our dataset $\mathcal{D}$ for training the residual consists of 50 pairs of initial trajectories ξ^h and their corresponding ground truths ξ^*. Our algorithm's performance — with and without the integration of the residual network — is then evaluated across 50 new and unexplored cup locations for each task.

Our results are illustrated in Figure 10. Across all tasks, VIEW with the residual demonstrated the ability to few-shot learn new object locations. We observed a reduction of over 40% in the number of rollouts required for the robot to learn each task. Indeed, VIEW with the residual network needed fewer than 10 trials on average to learn the correct behavior from distorted input trajectories. These results suggest that VIEW is not only effective when learning from scratch; we can also leverage the tasks that VIEW learned across previous video demonstrations to accelerate learning on a new video demonstration in the same workspace.

6 Experiments

In the previous section we explored the components of VIEW through an ablation study in a simulated environment. In this section we now test our overall method in the real-world with human video demonstrations. We start by collecting video demonstrations for various tasks such as picking up a cup or moving a basket. We then apply VIEW to extract the human hand and object priors from these videos (see Figure 12), and explore waypoints around these priors while repeatedly interacting with the environment until the robot successfully imitates the task. To see videos of these demonstrations and VIEW's learning process, visit: https://collab.me.vt.edu/view/

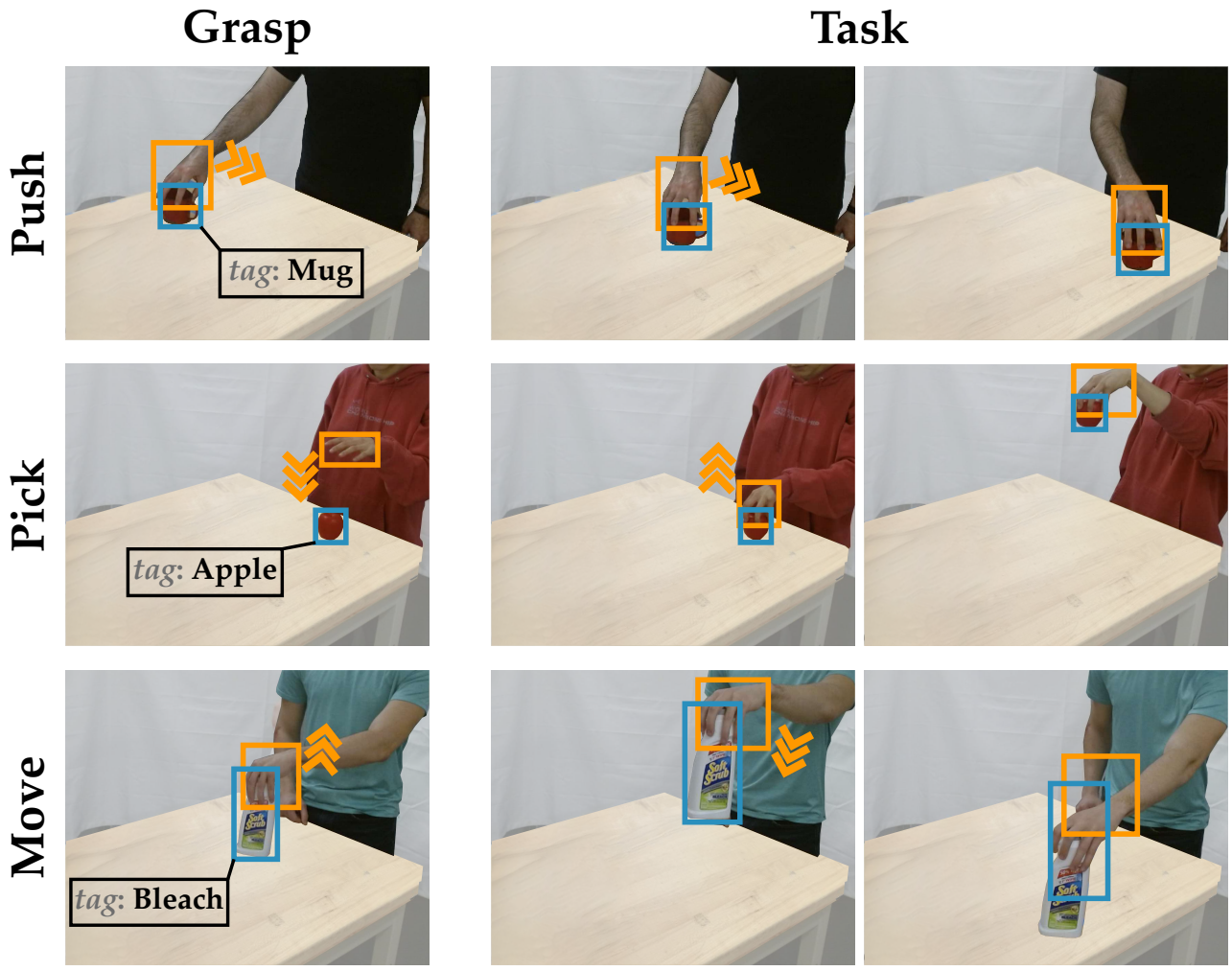

Fig. 11 Manipulation tasks from our experiments. People (including the authors and external participants) provided video demonstrations of three fundamental skills necessary for more complex tasks [47]: **push**, **pick**, **move**. Here we show examples frames where VIEW detected the human hand and the intended object, i.e., the object human is interacting with. VIEW used these frames to extract a prior trajectory for the human hand and object (also see Figure 12).

Tasks. Our real-world tasks span different skills and objects. We focus on three primitive skills — push, pick, move — since these primitive skills are fundamental across manipulation tasks [47]. A full list of the objects used in our experiments is found in the Appendix: these objects include household items such as foods, cups, and containers. We start with simple tasks where the robot must learn to push, pick, and move objects in *Uncluttered* environments where no other items are present. Next, we provide video demonstrations in *Cluttered* settings with multiple objects, and the robot must learn to imitate the demonstrated task despite this environmental clutter.

In *Uncluttered* tasks we test the three fundamental skills. **Push**: the robot must reach for the object and push it to a randomly assigned goal position; **Pick**: the robot must learn how to pick up an object; and **Move**: the robot must reach for an object, pick it up, and place it at a randomly assigned goal location. Figure 11 shows a demonstration for each skill.

In *Cluttered* tasks we test only the **Move** skill: here the robot must reach for and move the correct object while avoiding and ignoring the environmental clutter. Different objects may be placed close together in the environment to visually saturate the robot's camera or constrain the grasp locations for the target item.

Our method can scale to arbitrarily long tasks that involve manipulating multiple objects, as shown in our supplemental videos. However, for the purposes of this experiment, we only focus on single object manipulation tasks. Our primary aim is to test VIEW's ability to imitate manipulation tasks from a single video demonstration, and to compare VIEW to relevant baselines. As discussed in Section 4.5, we can use changes in the contact information to segment a demonstration that involves handling multiple objects into a sequence

of single object manipulation tasks. Hence, by exploring VIEW's performance on single object manipulation tasks, we obtain fundamental knowledge about the effectiveness of our proposed method.

Baselines. The primary baseline in our experiments is **WHIRL** [3], a state-of-the-art method for visual imitation learning from human demonstrations. However, the version of WHIRL implemented in our experiments differs in one way from the method described by [3]. Within the original work, WHIRL calculates rewards by comparing agent-agnostic representations of the human demonstration and robot interaction (similar to VIEW). WHIRL finds this agent-agnostic representation by inpainting the human and the robot from the videos using Copy-Paste Networks [34], and then using the action-recognition model of [44] to calculate its representation. But in our experiments we found that the Copy-Paste networks could not successfully inpaint the robot, despite careful fine-tuning on a custom dataset [3]. Accordingly, to create a fair comparison, we replaced the original reward model in WHIRL with our object-centric reward model from Section 4.2. We believe this is a reasonable change because our reward model actually provides more explicit feedback: it directly compares the movement of the target object across videos, rather than comparing a high-dimensional action representation as done in [3]. The rest of the WHIRL algorithm matches the original manuscript.

Our other experimental baselines are ablations of our approach. At one extreme we have **Prior**, a method that extracts the human hand trajectory from the video demonstration and then replays that trajectory on the robot arm. This corresponds to VIEW without any exploration or residual. In practice, **Prior** will only succeed if the initial trajectory the robot extracts is sufficient to successfully imitate the task. At the other extreme we tested **ours-BO**, an ablation of VIEW that leverages a different exploration scheme. Recall from Section 4.3 that VIEW divides exploration into two phases: learning to grasp and then matching the human's behavior. When learning to grasp we proposed a QD algorithm with high-level and low-level searches. **ours-BO** is a variant of VIEW that does not use this QD algorithm: instead, the robot employs Bayesian Optimization to separately identify the grasp and match the human's behavior. Finally, when the robot is learning multiple tasks, we also test **ours-residual**. This is our full VIEW algorithm that leverages previously solved tasks to improve its prior extraction.

Experimental Setup and Procedure. Experiments were conducted on a Universal Robots UR10 manipulator with 6 Degrees-of-Freedom. The human's video demonstrations were recorded with a RealSense D435 RGB-D camera at 60 frames per second. We also recorded the robot's interactions with the environment as it iteratively tried to imitate the demonstrated behavior. Note that we used the same camera angle for recording the human demonstrations and the robot interactions.

We recorded a single human video demonstration per task. Overall, we collected 13 video demonstrations, where 9 were in *Uncluttered* environments and 4 were in *Cluttered* environments. For the *Uncluttered* tasks the 9 total demonstrations were divided into 3 videos for each skill — *move-uncluttered*, *pick-uncluttered*, and *push-uncluttered*. The 4 videos in four different *Cluttered* environments all demonstrated the same skill *move-cluttered*. We conducted three trials on the robot for every demonstration, totalling 39 trials per method.

Between each trial **Prior**, **WHIRL**, and **ours** all reset and then learned the new task entirely from scratch. Put another way, even if the robot had learned to pick up a cup in the previous trial, the robot discarded that successful trajectory when starting the next trial. Here **ours-residual** was the exception: after we trained **ours** on the *Uncluttered* tasks, we used the data from these solved tasks to train our residual policy. We then tested **ours-residual** on the four *Cluttered* tasks and compared its performance to **ours** (i.e., our VIEW algorithm without including the residual).

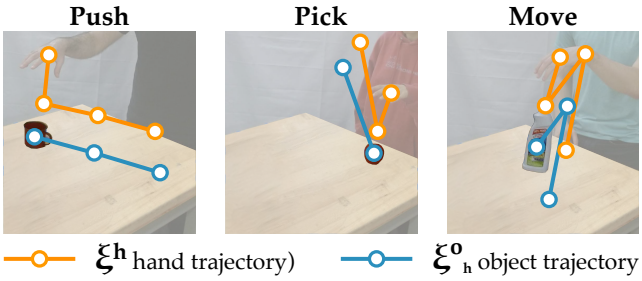

Fig. 12 Examples of the prior extracted from the human video demonstrations. Our method outputs how the human hand moves (ξ^h) and how the object moves (ξ^o_h) throughout the video. Each trajectory is compressed such that it only consists of waypoints that mark significant changes in the motion (like change in direction, change in contact, etc.).

6.1 Results

Uncluttered Tasks. The results from our experiments on *Uncluttered* tasks are shown in Figure 13. These plots display the results across the two phases of each task: the success rate for learning to grasp the object, and the success rate for learning to correctly manipu-

[3] See the Appendix for more detailed analysis

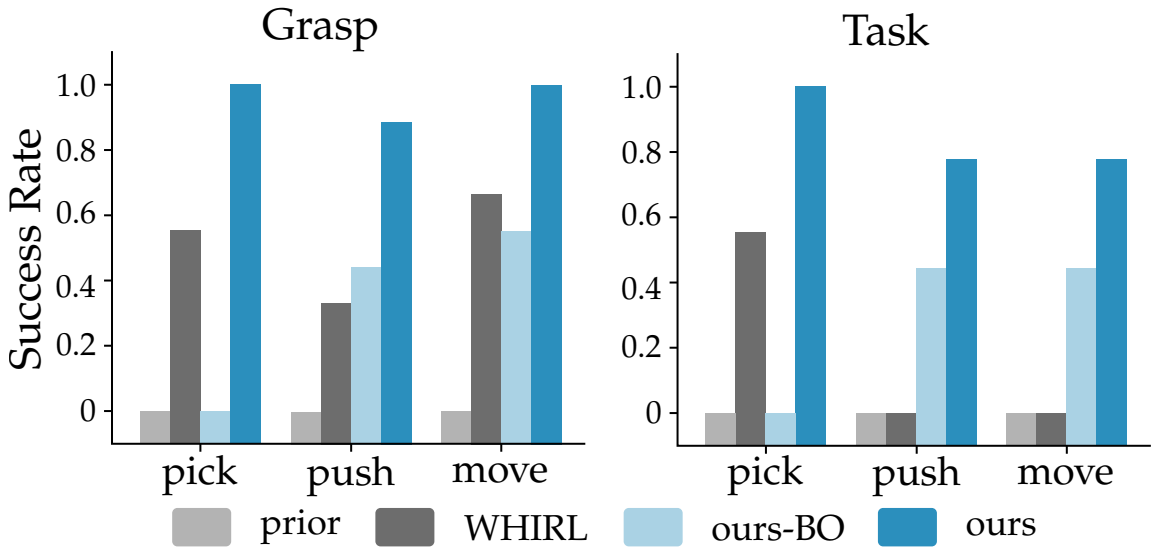

Fig. 13 Experiment results for *Uncluttered* tasks. (Left) How frequently the robot grasped the object from the human's video demonstration. (Right) How frequently the robot learned to imitate the human's video demonstration. Results are calculated across 9 separate video demonstrations and 3 trials per video demonstration. Note that the results for **pick** are the same in both grasping and task exploration, since here the objective is just to pick up (i.e., grasp) an item.

late that object. The robot is said to have "succeeded" if it was able to pick up the object and approximately imitate the human. For instance, in a moving task a successful grasp would mean that the robot held the object without dropping it. The definition of successful task completion varied between the different tasks: for *move-uncluttered*, the task was considered a success if the robot placed the object close to the same location as the human. In *Pick-uncluttered*, the robot was successful it if grasped the target object and lifted it off the table, while in *push-uncluttered*, the robot successfully completed the task if it pushed the item to the human's demonstrated location.

We notice that simply replaying the trajectory that the robot extracted from the human's video demonstration was never successful. Across all tasks, **Prior** was not able to either grasp or manipulate the target object. The state-of-the-art visual imitation learning baseline **WHIRL** was more effective, particularly in learning to grasp the target item. But **our** proposed VIEW algorithm surpassed this baseline, reaching more than twice the success rate of **WHIRL** for the push task and achieving a 100% success percentage in the pick task. For *push-uncluttered* and *move-uncluttered*, **WHIRL** was able to grasp the target object in some trials, but it did not learn to correctly manipulate that object within the limit of 100 rollouts in the environment (roughly 45 minutes). For these same tasks **our** VIEW algorithm reached an 80% success rate, learning to replicate the human's video demonstrations in less than 30 minutes[4].

Finally, we compared the performance of **ours-BO** and **ours**. Across the board, we found that **ours-BO** is less effective at visual imitation learning than our full VIEW algorithm, and in the *pick-uncluttered* environment this baseline performs significantly worse than **WHIRL**. These results highlight the importance of our high-level and low-level QD search algorithms for exploring how to grasp the object: without the ability to learn effective grasps, **ours-BO** struggles to imitate the rest of the manipulation task.

Cluttered Tasks. We present the results for the *Cluttered* task trials in Figure 14. As before, the plot on the left shows the grasp success rate, and the plot on the right shows the full task success rate. The key difference between these *Cluttered* experiments and the previous results was that — in this case — there were multiple objects on the table, and the robot had to determine which of these objects the human manipulated in order to accurately match their video demonstration. Our results followed the same trends as in *Uncluttered* tasks. Directly executing the extracted prior never led to success for either grasping or manipulation. Again, this suggests that the robot must refine its estimate of the human's hand trajectory to successfully replicate

[4] WHIRL was shown to work for similar tasks in the original paper [3]. However, we were unable to reproduce these results. We acknowledge that we replaced WHIRL's original reward model with our own agent-agnostic reward. However, this new reward provides more explicit feedback about the task and exploration. See Appendix for more details.

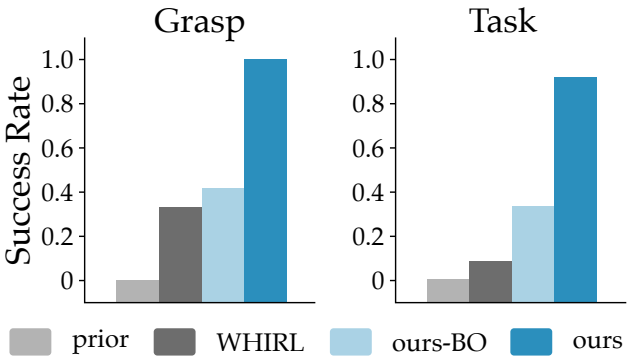

Fig. 14 Experiment results for *Cluttered* tasks. Here the environment contained multiple extraneous items in addition to the target object the human manipulated. (Left) How frequently the robot learned to grasp the correct item. (Right) How frequently the robot correctly imitated the entire video demonstration. These results were taken across 4 separate video demonstrations and 3 trials per video demonstration (for a total of 12 datapoints).

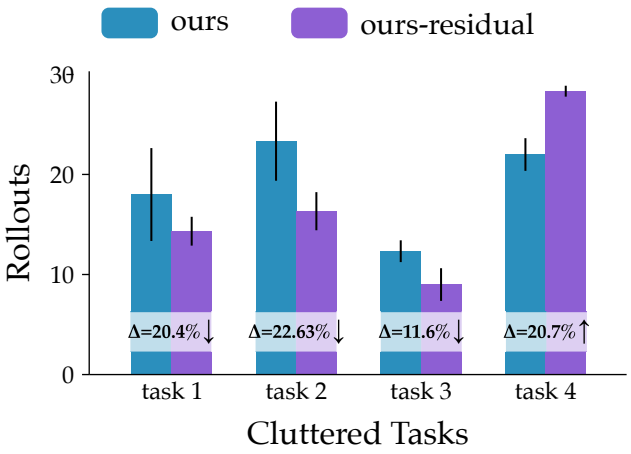

Fig. 15 Experiment results for learning from multiple tasks. Here the robot has previously solved the *Uncluttered* tasks, and it is now trying to learn a new *Cluttered* task. We compare our VIEW algorithm without the residual (*ours*) to our full VIEW algorithm with the residual (*ours-residual*). There are a total of four video demonstration for different *Cluttered* tasks. We plot the average number of rollouts needed for VIEW to solve each of these tasks. Δ is the percentage change in the number of rollouts with and without the residual. The bars indicate standard error of the mean.

their object interactions. The baselines **WHIRL** and **ours-BO** were roughly similar, reaching success percentages of less than 50% across a maximum of 100 real-world rollouts (roughly 45 minutes). We were not surprised that **WHIRL** struggled with cluttered environments: it does not split the exploration into separate parts for grasping and manipulation; even if the robot grasps the object in an interaction, it can fail to grasp it again in the subsequent repetitions[5]. Overall, **our** VIEW method was effective across the cluttered settings, grasping and manipulating the correct object to match the human's video demonstration in almost 100% of the trials. VIEW solved each task in less than 30 minutes.

Learning from Multiple Tasks. In Figure 15 we summarize the results from our final experiment. This experiment focused on how VIEW can leverage the tasks it has previously solved to improve its prior and accelerate its learning on new tasks. To quantify this acceleration, we measured the number of rollouts it took for the robot to successfully imitate a video demonstration in the *Cluttered* environment. Both **ours** and **ours-residual** used VIEW, but **ours-residual** included the full VIEW algorithm with the residual component. We found that for 3 out of the 4 *Cluttered* demonstrations, applying the residual significantly reduced the number of interactions needed to learn the task (roughly 25% fewer rollouts). Interestingly, for the fourth demonstration the residual actually had the opposite effect, and slowed down the robot's learning. On further examination, we think this decrease in performance likely occurred because the initial hand trajectory ξ^h lied outside the distribution of the data used to train the residual. Since the residual had not seen a demonstration that operated in the same part of the workspace as this video, it was not able to de-noise the prior and accelerate the robot's learning. This suggests that — while the residual can be useful — it should be carefully applied. Learning a robust residual necessitates an expansive dataset that includes waypoints spanning the workspace. Any waypoint in the regions not covered by the training data cannot be reliably de-noised by the model, and thus designers may only want to apply the residual when the human is performing a demonstration that is spatially similar to a previously solved task.

[5] It is important to note that our reward model provides explicit feedback about the *tagged* object: the rewards do not change if WHIRL moves any object other than the *tagged* object. In contrast, the original reward model in WHIRL compares the agent-agnostic action embeddings. This would pose a significant challenge in a cluttered environment since the robot would still be executing the right behavior, but manipulating the wrong object. For instance, if the robot were to move the kettle instead of the cup, it performs the same action — moving — and would receive a high reward even though it actually fails to complete the task.

7 Conclusion

State-of-the-art visual imitation learning methods rely on intricate architectures to manage the complexities present in video demonstrations. This paper introduces an alternative framework designed to streamline the learning process by compressing video data and honing in on crucial features and waypoints. We show that

by concentrating on these essential aspects, robots can more rapidly learn tasks from human video demonstrations. Our method, VIEW, incorporates distinct modules for (a) generating a condensed prior that captures the key aspects of the human demonstrator's intent, (b) facilitating targeted exploration around the waypoints in the prior through a division into grasp and task execution phases, and (c) employing a residual model to enhance learning efficiency by drawing on insights from previously completed tasks.

Through an ablation study in a simulated environment, we examine the contribution of each module to VIEW's overall efficacy. Subsequent real-world experiments, utilizing videos of human demonstrations, further validate our method's capability to effectively learn from such demonstrations. The combined results from our simulation studies and real-world testing indicate that VIEW can efficiently learn tasks demonstrated using a single video, typically requiring under 30 minutes and fewer than 20 real-world trials. Additionally, we advance the capabilities of human-to-robot visual imitation learning by showing that VIEW can learn from arbitrarily long video demonstrations involving multiple object interactions. These findings are illustrated in our supplemental videos, available here: https://collab.me.vt.edu/view/

Limitations. Our method has demonstrated the capability to expedite the learning process from human demonstrations, significantly reducing the required time from several hours [3] to less than 30 minutes. However, achieving this level of success comes with its own set of constraints. A primary limitation is the necessity for the learning environment to mirror the setup used in the human demonstrations, including the identical positioning of objects. This requirement stems from the specific mechanics of our prior extraction and reward computation processes.

Additionally, our approach is tied to a specific demonstration. For example, if a video shows a human picking up a cup from a certain location, the robot will learn to pick up the cup from that same location. If the location of the cup changes, the robot cannot adapt to perform the task at the new location. To overcome this limitation, we can integrate our method with behavior cloning [51] or another policy learning framework. This integration would allow the robot to convert human demonstrations into robot demonstrations that include state-action pairs. Using behavior cloning, the robot could then learn a more generalized policy capable of adapting to changes in the world state. This approach effectively positions our method as an intermediary layer, translating human demonstrations into a format suitable for imitation learning policies that depend on state-action pairs for training.

8 Declarations

Funding. This research was supported in part by the USDA National Institute of Food and Agriculture, Grant 2022-67021-37868.

Conflict of Interest. The authors declare that they have no conflicts of interest.

Ethical Statement. All physical experiments that relied on interactions with humans were conducted under university guidelines and followed the protocol of Virginia Tech IRB #20-755.

Author Contribution. A.J. led the algorithm development for prior extraction and agent agnostic reward computation. S.P. led the development for exploration. A.J. and S.P. wrote the first manuscript draft. A.J. ran the simulations and S.P. conducted the physical experiments. D.L. supervised the project, helped develop the method, and edited the manuscript.

Acknowledgements. We thank Heramb Nemlekar for his valuable feedback on our manuscript.

References

1. Alakuijala, M., Dulac-Arnold, G., Mairal, J., Ponce, J., Schmid, C.: Learning reward functions for robotic manipulation by observing humans. In: IEEE International Conference on Robotics and Automation (2023)
2. Amiranashvili, A., Dorka, N., Burgard, W., Koltun, V., Brox, T.: Scaling imitation learning in minecraft. arXiv preprint arXiv:2007.02701 (2020)
3. Bahl, S., Gupta, A., Pathak, D.: Human-to-robot imitation in the wild. In: Robotics: Science and Systems (2022)
4. Brown, D., Goo, W., Nagarajan, P., Niekum, S.: Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In: International Conference on Machine Learning (2019)
5. Brown, D.S., Goo, W., Niekum, S.: Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In: Conference on Robot Learning (2020)
6. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
7. Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., Dollar, A.M.: Yale-CMU-Berkeley dataset for robotic manipulation research. The International Journal of Robotics Research (2017)
8. Cetin, E., Celiktutan, O.: Domain-robust visual imitation learning with mutual information constraints. In: International Conference on Learning Representations (2021)

9. Chane-Sane, E., Schmid, C., Laptev, I.: Learning video-conditioned policies for unseen manipulation tasks. In: International Conference on Robotics and Automation (2023)
10. Chen, J., Yuan, B., Tomizuka, M.: Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2019)
11. Das, P., Xu, C., Doell, R.F., Corso, J.J.: A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In: IEEE Conference on Computer Vision and Pattern Recognition (2013)
12. Duan, J., Wang, Y.R., Shridhar, M., Fox, D., Krishna, R.: Ar2-d2: Training a robot without a robot. arXiv preprint arXiv:2306.13818 (2023)
13. Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning (2021)
14. Eze, C., Crick, C.: Learning by watching: A review of video-based learning approaches for robot manipulation. arXiv preprint arXiv:2402.07127 (2024)
15. Fang, B., Jia, S., Guo, D., Xu, M., Wen, S., Sun, F.: Survey of imitation learning for robotic manipulation. International Journal of Intelligent Robotics and Applications (2019)
16. Fontaine, M.C., Togelius, J., Nikolaidis, S., Hoover, A.K.: Covariance matrix adaptation for the rapid illumination of behavior space. In: Genetic and Evolutionary Computation Conference (2020)
17. Gouda, A., Ghanem, A., Reining, C.: DoPose-6D dataset for object segmentation and 6D pose estimation. In: IEEE International Conference on Machine Learning and Applications (2022)
18. Gouda, A., Roidl, M.: Dounseen: Zero-shot object detection for robotic grasping. arXiv preprint arXiv:2304.02833 (2023)
19. Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M.: The" something something" video database for learning and evaluating visual common sense. In: IEEE International Conference on Computer Vision (2017)
20. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: IEEE international Conference on Computer Vision (2017)
21. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: A survey of learning methods. ACM Computing Surveys (2017)
22. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
23. Jain, V., Attarian, M., Joshi, N.J., Wahid, A., Driess, D., Vuong, Q., Sanketi, P.R., Sermanet, P., Welker, S., Chan, C., et al.: Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers. arXiv preprint arXiv:2403.12943 (2024)
24. Jin, J., Petrich, L., Dehghan, M., Jagersand, M.: A geometric perspective on visual imitation learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2020)
25. Jonnavittula, A., Losey, D.P.: I know what you meant: Learning human objectives by (under) estimating their choice set. In: IEEE International Conference on Robotics and Automation (2021)
26. Jonnavittula, A., Losey, D.P.: Learning to share autonomy across repeated interaction. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2021)
27. Jonnavittula, A., Mehta, S.A., Losey, D.P.: SARI: Shared autonomy across repeated interaction. ACM Transactions on Human-Robot Interaction (2024)
28. Kelly, M., Sidrane, C., Driggs-Campbell, K., Kochenderfer, M.J.: HG-DAgger: Interactive imitation learning with human experts. In: IEEE International Conference on Robotics and Automation (2019)
29. Kim, M.J., Wu, J., Finn, C.: Giving robots a hand: Learning generalizable manipulation with eye-in-hand human video demonstrations. arXiv preprint arXiv:2307.05959 (2023)
30. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research (2013)
31. Koppula, H.S., Gupta, R., Saxena, A.: Learning human activities and object affordances from rgb-d videos. The International Journal of Robotics Research (2013)
32. Koppula, H.S., Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. IEEE Transactions on Pattern Analysis and Machine Intelligence (2015)
33. Lee, R., Abou-Chakra, J., Zhang, F., Corke, P.: Learning fabric manipulation in the real world with human videos. arXiv preprint arXiv:2211.02832 (2022)
34. Lee, S., Oh, S.W., Won, D., Kim, S.J.: Copy-and-paste networks for deep video inpainting. In: IEEE/CVF International Conference on Computer Vision (2019)
35. Li, J., Lu, T., Cao, X., Cai, Y., Wang, S.: Meta-imitation learning by watching video demonstrations. In: International Conference on Learning Representations (2021)
36. Liu, P., Orru, Y., Paxton, C., Shafiullah, N.M.M., Pinto, L.: Ok-robot: What really matters in integrating open-knowledge models for robotics. arXiv preprint arXiv:2401.12202 (2024)
37. Liu, Y., Gupta, A., Abbeel, P., Levine, S.: Imitation from observation: Learning to imitate behaviors from raw video via context translation. In: IEEE International Conference on Robotics and Automation (2018)
38. Luo, H., Zhai, W., Zhang, J., Cao, Y., Tao, D.: Learning visual affordance grounding from demonstration videos. IEEE Transactions on Neural Networks and Learning Systems (2023)
39. Lynch, C., Sermanet, P.: Language conditioned imitation learning over unstructured data. In: Robotics: Science and Systems (2020)
40. Ma, M., Marturi, N., Li, Y., Leonardis, A., Stolkin, R.: Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos. Pattern Recognition (2018)
41. Mehta, S.A., Habibian, S., Losey, D.P.: Waypoint-based reinforcement learning for robot manipulation tasks. arXiv preprint arXiv:2403.13281 (2024)
42. Mehta, S.A., Losey, D.P.: Unified learning from demonstrations, corrections, and preferences during physical human-robot interaction. ACM Transactions on Human-Robot Interaction (2023)
43. Menda, K., Driggs-Campbell, K., Kochenderfer, M.J.: EnsembleDAgger: A bayesian approach to safe imitation learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2019)
44. Monfort, M., Pan, B., Ramakrishnan, K., Andonian, A., McNamara, B.A., Lascelles, A., Fan, Q., Gutfreund, D., Feris, R.S., Oliva, A.: Multi-moments in time: Learning

and interpreting models for multi-action video understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
45. Morales, E.F., Murrieta-Cid, R., Becerra, I., Esquivel-Basaldua, M.A.: A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. Intelligent Service Robotics (2021)
46. Muckell, J., Olsen, P.W., Hwang, J.H., Lawson, C.T., Ravi, S.: Compression of trajectory data: A comprehensive evaluation and new approach. GeoInformatica (2014)
47. Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al.: Open x-embodiment: Robotic learning datasets and rt-x models. arXiv preprint arXiv:2310.08864 (2023)
48. Pan, Y., Cheng, C.A., Saigol, K., Lee, K., Yan, X., Theodorou, E.A., Boots, B.: Imitation learning for agile autonomous driving. The International Journal of Robotics Research (2020)
49. Pari, J., Shafiullah, N.M., Arunachalam, S.P., Pinto, L.: The surprising effectiveness of representation learning for visual imitation. In: Robotics: Science and Systems (2021)
50. Patel, A., Wang, A., Radosavovic, I., Malik, J.: Learning to imitate object interactions from internet videos. arXiv preprint arXiv:2211.13225 (2022)
51. Pomerleau, D.A.: Efficient training of artificial neural networks for autonomous navigation. Neural Computation (1991)
52. Rafailov, R., Yu, T., Rajeswaran, A., Finn, C.: Visual adversarial imitation learning using variational models. Advances in Neural Information Processing Systems (2021)
53. Ratliff, N., Bagnell, J.A., Srinivasa, S.S.: Imitation learning for locomotion and manipulation. In: IEEE-RAS International Conference on Humanoid Robots (2007)
54. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (2015)
55. Romero, J., Tzionas, D., Black, M.J.: Embodied hands: Modeling and capturing hands and bodies together. ACM Transactions on Graphics (2017)
56. Rong, Y., Shiratori, T., Joo, H.: Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In: IEEE International Conference on Computer Vision Workshops (2021)
57. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: International Conference on Artificial Intelligence and Statistics (2011)
58. Schaal, S.: Learning from demonstration. In: Advances in Neural Information Processing Systems (1996)
59. Schäfer, L., Jones, L., Kanervisto, A., Cao, Y., Rashid, T., Georgescu, R., Bignell, D., Sen, S., Gavito, A.T., Devlin, S.: Visual encoders for data-efficient imitation learning in modern video games. arXiv preprint arXiv:2312.02312 (2023)
60. Scheller, C., Schraner, Y., Vogel, M.: Sample efficient reinforcement learning through learning from demonstrations in minecraft. In: NeurIPS Competition and Demonstration Track (2020)
61. Sermanet, P., Xu, K., Levine, S.: Unsupervised perceptual rewards for imitation learning. In: Robotics: Science and Systems (2017)
62. Shafiullah, N.M.M., Rai, A., Etukuru, H., Liu, Y., Misra, I., Chintala, S., Pinto, L.: On bringing robots home. arXiv preprint arXiv:2311.16098 (2023)
63. Shan, D., Geng, J., Shu, M., Fouhey, D.F.: Understanding human hands in contact at internet scale. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
64. Sharma, P., Pathak, D., Gupta, A.: Third-person visual imitation learning via decoupled hierarchical controller. In: Advances in Neural Information Processing Systems (2019)
65. Shaw, K., Bahl, S., Sivakumar, A., Kannan, A., Pathak, D.: Learning dexterity from human hand motion in internet videos. The International Journal of Robotics Research (2024)
66. Shi, L.X., Hu, Z., Zhao, T.Z., Sharma, A., Pertsch, K., Luo, J., Levine, S., Finn, C.: Yell at your robot: Improving on-the-fly from language corrections. arXiv preprint arXiv:2403.12910 (2024)
67. Shi, L.X., Sharma, A., Zhao, T.Z., Finn, C.: Waypoint-based imitation learning for robotic manipulation. In: Conference on Robot Learning (2023)
68. Sieb, M., Xian, Z., Huang, A., Kroemer, O., Fragkiadaki, K.: Graph-structured visual imitation. In: Conference on Robot Learning (2020)
69. Smith, L., Dhawan, N., Zhang, M., Abbeel, P., Levine, S.: AVID: Learning multi-stage tasks via pixel-level translation of human videos. In: Robotics: Science and Systems (2020)
70. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems (2012)
71. Song, S., Zeng, A., Lee, J., Funkhouser, T.: Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. IEEE Robotics and Automation Letters (2020)
72. Taranovic, A., Kupcsik, A.G., Freymuth, N., Neumann, G.: Adversarial imitation learning with preferences. In: International Conference on Learning Representations (2022)
73. Tremblay, J., To, T., Birchfield, S.: Falling things: A synthetic dataset for 3d object detection and pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (2018)
74. Vassiliades, V., Chatzilygeroudis, K., Mouret, J.B.: Using centroidal voronoi tessellations to scale up the multi-dimensional archive of phenotypic elites algorithm. IEEE Transactions on Evolutionary Computation (2017)
75. Wang, J., Mueller, F., Bernard, F., Sorli, S., Sotnychenko, O., Qian, N., Otaduy, M.A., Casas, D., Theobalt, C.: Rgb2hands: Real-time tracking of 3d hand interactions from monocular rgb video. ACM Transactions on Graphics (2020)
76. Wen, B., Lian, W., Bekris, K., Schaal, S.: You only demonstrate once: Category-level manipulation from single visual demonstration. In: Robotics: Science and Systems (2022)
77. Wen, C., Lin, J., Qian, J., Gao, Y., Jayaraman, D.: Keyframe-focused visual imitation learning. In: International Conference on Machine Learning (2021)
78. Xiong, H., Li, Q., Chen, Y.C., Bharadhwaj, H., Sinha, S., Garg, A.: Learning by watching: Physical imitation of manipulation skills from human videos. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2021)
79. Young, S., Gandhi, D., Tulsiani, S., Gupta, A., Abbeel, P., Pinto, L.: Visual imitation made easy. In: Conference on Robot Learning (2021)

80. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.L., Grundmann, M.: Mediapipe hands: On-device real-time hand tracking. In: CVPR Workshop on Computer Vision for Augmented and Virtual Reality (2020)

A Appendix

A.1 Implementation Details

A public repository of our code can be found here: https://github.com/VT-Collab/view

Hand trajectory extraction. In line with the methodology described in WHIRL [3], we utilize the 100 Days of Hands (100DOH) detector from https://github.com/ddshan/hand_detector.d2 for identifying hand-object contact points. For wrist detection, we integrate this with FrankMocap, as documented in Rong *et al.* [56], without any model fine-tuning. The implementation for FrankMocap can be found here: https://github.com/facebookresearch/frankmocap. To obtain compressed trajectories, we combine the output of the FrankMocap model with SQUISHE. We develop our own version of SQUISHE based on the description provided in [46]. Our implementation can be accessed in the code repository.

Object trajectory extraction. To identify objects within the scene, we use Mask R-CNN, as detailed by He *et al.* [20], through its implementation in Detectron2 (https://github.com/facebookresearch/detectron2). Following the methodologies outlined in [18], we initially pretrain our model using the Nvidia Falling Things dataset [73] and the DoPose-6D dataset [17]. We then finetune the model on a custom dataset containing 21 objects, with a subset of 7 being directly relevant to our final evaluations. This subset includes standard objects from the YCB object dataset [7] and others that are commonly found in kitchen environments. The complete list of objects used in our evaluation is shown in Figure 16.

Residual network. For our residual network, we employ a fully connected multi-layer perceptron with two hidden layers, utilizing ReLU as the activation function and mean squared error (MSE) for loss calculation. We use the Adam optimizer and train the network for 100 epochs. The initial learning rate is set at 0.1, with a decay factor of 0.15. For more detailed information on our training parameters, please refer to our code repository.

A.2 Challenges with WHIRL

Because of the lack of publicly available implementations of WHIRL, we developed our version based on the algorithms provided in WHIRL's publication [3]. As described, we used a four-layer MLP, implemented as a Variational Autoencoder and optimized via KL divergence loss. Initially — consistent with the guidelines in WHIRL's manuscript — we employed Copy-Paste Networks for inpainting [34] and the moment model from Monfort *et al.* [44] for calculating rewards.

However, during our experiments, we encountered two major challenges with WHIRL (see Figure 17). The first issue was the inconsistency observed in the video inpainting performance, where the Copy-Paste Network failed to fully remove the robot from several frames (See Figure 17 Top). This inconsistency persisted even after we fine-tuned the model on

Fig. 16 Objects manipulated in our real-world experiments. (From left to right) We use a bottle of bleach, a kettle, a mug, a banana, an apple, a bottle of mustard, and a basket. These seven distinct items were systematically selected for assessment based on their varying shapes, sizes, and colors to provide a comprehensive evaluation of our algorithm.

400 custom images of our robot. Having consistent images with the human and the robot removed are particularly critical because WHIRL's exploration strategy relies heavily on comparing embeddings across frames. Due to the erratic inpainting results, the robot often converged to suboptimal positions, distant from the target object (See Figure 17 Bottom Left). The second issue pertained to the rewards linked with task completion. There was a marked lack of differentiation in rewards between trajectories where the robot only grasped the object and those where it successfully completed the task (see Figure 17 Bottom Right). This similarity in rewards often caused the robot to become stuck in a local minimum, proficient at object pickup but failing to complete the rest of the manipulation task.

In response to these issues, we replaced the reward model from WHIRL with the reward model described in Section 4.2. While WHIRL calculates exploration rewards based on the variance in frame embeddings and task rewards through the difference in video embeddings, our method takes a different approach. We explicitly compute exploration rewards using changes in the object's position and gauge task completion by measuring the object's proximity to the demonstrated trajectory. Our reward structure therefore capitalizes on direct and pertinent information (i.e., the location of the target object) rather than an indeterminate high-dimensional representation. We conducted a limited set of experiments to ensure that reward responses from our model were comparable to those from the original moment model. We believe that this modification does not fundamentally alter the functionality of WHIRL, and is a reasonable baseline for comparison.

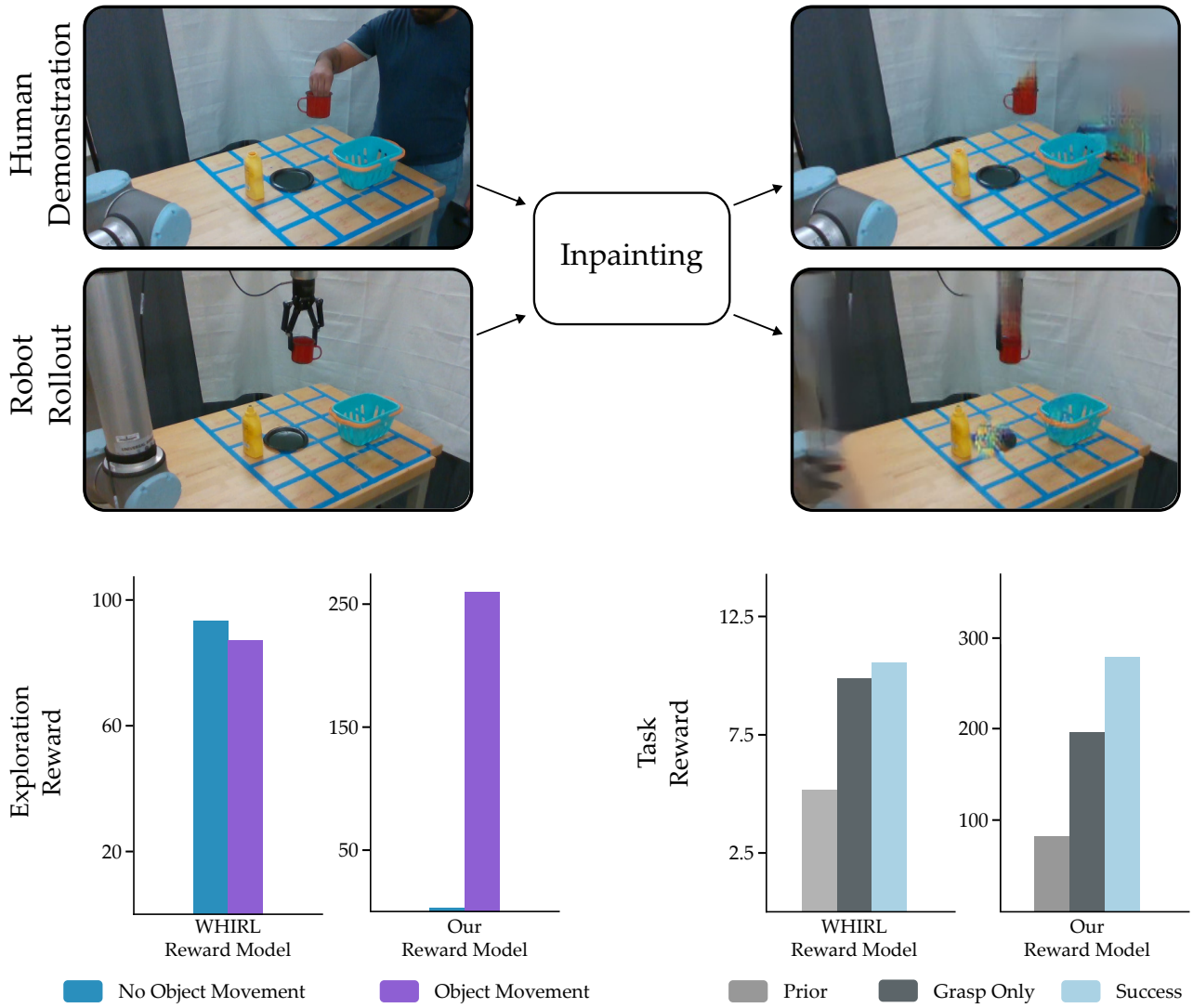

Fig. 17 Challenges with WHIRL Evaluation. (Top) As described in [3], we utilized Copy-Paste Networks [34] for the purpose of video inpainting, with the aim of removing both the human demonstrator and the robot arm from the video frames. This process is critical for enabling the comparison of frames through moment models [44], which in turn facilitates the computation of agent-agnostic rewards. However, in our evaluations we encountered consistency issues with the inpainted images, leading to highly variable reward signals. (Bottom Left) The inconsistency in reward signals led to scenarios where the robot received high exploration rewards without actually moving the object. This is problematic because the robot relies on these rewards to identify waypoints that are near the object, which are necessary for successful grasping. In contrast, WHIRL with our reward model produces low exploration rewards when there is no object movement, and rewards increase significantly only when the object is displaced. This variability in the WHIRL reward model often caused the robot's learning trajectory to converge prematurely at a suboptimal point, usually far from the target object. (Bottom Right) When the robot managed to overcome the variability in exploration rewards and successfully grasped the object, we observed that the reward difference between just grasping the object and completing the entire task was minimal. WHIRL with our reward model provided a clearer distinction between these different phases of the task. The lack of clear reward differentiation in WHIRL's reward model frequently hindered the robot's ability to fully learn the task, often resulting in the robot only learning to pick up the object without completing subsequent steps. Based on these results, in our experiments from Section 6 we used WHIRL with our proposed reward model instead of WHIRL with its original reward model.