



Learning latent actions to control assistive robots

Dylan P. Losey¹ · Hong Jun Jeon² · Mengxi Li² · Krishnan Srinivasan² · Ajay Mandlekar² · Animesh Garg³ · Jeannette Bohg² · Dorsa Sadigh²

Received: 30 January 2021 / Accepted: 6 July 2021 / Published online: 4 August 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

Assistive robot arms enable people with disabilities to conduct everyday tasks on their own. These arms are dexterous and *high-dimensional*; however, the interfaces people must use to control their robots are *low-dimensional*. Consider teleoperating a 7-DoF robot arm with a 2-DoF joystick. The robot is helping you eat dinner, and currently you want to cut a piece of tofu. Today's robots assume a pre-defined mapping between joystick inputs and robot actions: in one mode the joystick controls the robot's motion in the x - y plane, in another mode the joystick controls the robot's z - yaw motion, and so on. But this mapping misses out on the task you are trying to perform! Ideally, one joystick axis should control how the robot stabs the tofu, and the other axis should control different cutting motions. Our insight is that we can achieve intuitive, user-friendly control of assistive robots by *embedding* the robot's high-dimensional actions into low-dimensional and human-controllable *latent actions*. We divide this process into three parts. First, we explore models for learning latent actions from offline task demonstrations, and formalize the properties that latent actions should satisfy. Next, we combine learned latent actions with autonomous robot assistance to help the user reach and maintain their high-level goals. Finally, we learn a personalized alignment model between joystick inputs and latent actions. We evaluate our resulting approach in four user studies where non-disabled participants reach marshmallows, cook apple pie, cut tofu, and assemble dessert. We then test our approach with two disabled adults who leverage assistive devices on a daily basis.

Keywords Assistive robotics · Teleoperation · Shared autonomy · Latent representations

1 Introduction

For over one million American adults living with physical disabilities, performing everyday tasks like grabbing a bite of food or pouring a glass of water presents a significant challenge (Taylor 2018). Assistive devices—such as wheelchair-mounted robot arms—have the potential to improve these people's independence and quality of life

(Argall 2018; Jacobsson et al. 2000; Mitzner et al. 2018; Carlson and Millan 2013). A key advantage of these robots is their *dexterity*: assistive arms move along multiple degrees-of-freedom (DoFs), orchestrating complex motions like stabbing a piece of tofu or pouring a glass of water. Unfortunately, this very dexterity makes assistive arms hard to control.

Imagine that you are leveraging an assistive robot arm to eat dinner (see Fig. 1). You want the robot to reach for some tofu on the table in front of you, cut off a piece, and then pick it up with its fork. Non-disabled persons can use their own body to show the robot how to perform this task: for instance, the human grabs the tofu with their own arm, and the robot mimics the human's motion (Rakita et al. 2017, 2019). But mimicking is not feasible for people living with physical disabilities—instead, these users are limited to *low-dimensional* controllers. Today's assistive robot arms leverage joysticks (Herlant et al. 2016), sip-and-puff devices (Argall 2018), or brain-computer interfaces (Muelling et al. 2017). So to get a bite of tofu, you must carefully coordinate the dexterous robot arm while only pressing up-down-left-

This is one of the several papers published in Autonomous Robots comprising the Special Issue on Robotics: Science and Systems 2020.

✉ Dylan P. Losey
losey@vt.edu

¹ Mechanical Engineering Department, Virginia Tech, Blacksburg, USA

² Computer Science Department, Stanford University, Stanford, USA

³ Computer Science Department, University of Toronto, Toronto, Canada

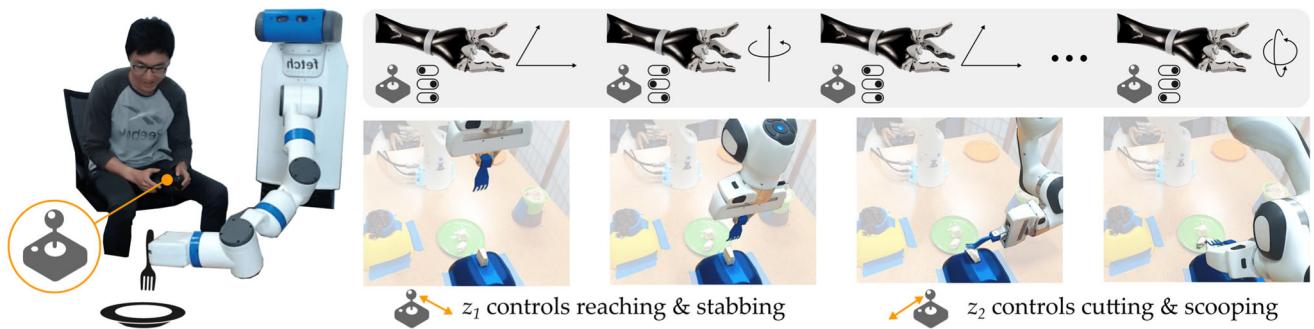


Fig. 1 Our approach makes it easier for users to control assistive robots. (Left) assistive robot arms are dexterous and high-dimensional, but humans must teleoperate these robots with low-dimensional interfaces, such as 2-DoF joysticks. (Right) we focus on assistive eating tasks; for example, trying to get a piece of tofu. (Top) existing work maps joystick inputs to end-effector motion. Here the user must toggle back and forth between multiple modes to control their desired end-effector

motion. (Bottom) we learn task specific mapping that embeds the robot's high-dimensional actions into low-dimensional *latent actions* z . Now pressing up and down controls the robot along a reaching and stabbing motion, while pressing right and left moves the robot arm through precise cutting and scooping motions. The user no longer needs to change modes

right on a joystick. Put another way, users are challenged by an inherent mismatch between low-dimensional interfaces and high-dimensional robots.

Existing work on assistive robots tackles this problem with *pre-defined* mappings between user inputs and robot actions. These mappings incorporate modes, and the user switches between modes to control different robot DoFs (Herlant et al. 2016; Aronson et al. 2018; Newman et al. 2018). For instance, in one mode the user's 2-DoF joystick controls the x - y position of the end-effector, in a second mode the joystick controls the z - yaw position of the end-effector, and so on. Importantly, these pre-defined mappings miss out on the human's underlying task. Consider teleoperating the robot to cut off a piece of tofu and then stab it with its fork. First you must use the x - y mode to align the fork above the tofu, then *roll-pitch* to orient the fork for cutting, then z - yaw to move the fork down into the tofu, and then back to *roll-pitch* to return the fork upright, and finally z - yaw to stab the tofu—and this is assuming you never undo a motion or make a correction!

Controlling assistive robots becomes easier when the joystick inputs map directly to task-related motions. Within our example, one joystick DoF could produce a spectrum of stabbing motions, while the other DoF teleoperates the robot through different cutting motions. To address the fundamental mismatch between high-DoF robot arms and low-DoF control interfaces, we learn a mapping between these spaces:

We make it easier to control high-dimensional robots by embedding the robot's actions into low-dimensional and human-controllable latent actions.

Latent actions here refer to a low-DoF representation that captures the most salient aspects of the robot's motions. Intuitively, we can think of these latent actions as similar to the eigenvectors of a matrix composed of high-dimensional robot

motions. Returning to our motivation, imagine that you have eaten dinner with the assistive robot many times. Across all of these meals there are some common motions: reaching for food items, cutting, pouring, scooping, etc. At the heart of our approach we *learn* an embedding that captures these underlying motion patterns, and enables the human to *control* via these learned embeddings, which we refer to as latent actions.

Overall, we make the following contributions:¹

Learning latent actions Given a dataset of task-related robot motions, we develop a framework for learning to map the user's low-dimensional inputs to high-dimensional robot actions. For instance, imagine using a 2-DoF joystick to teleoperate a 7-DoF assistive robot arm. To reach for a piece of tofu, you need a mapping function—something that interprets your joystick inputs into robot actions. Of course, not just any mapping will do; you need something that is intuitive and meaningful, so that you can easily coordinate all the robot's joints to move towards your tofu. In Sect. 6 we introduce a set of properties that user-friendly latent actions must satisfy, and formulate learning models that capture these properties.

Integrating shared autonomy But what happens once you've guided the robot to reach the tofu (i.e., your high-level goal)? Next, you need to precisely manipulate the robot arm in order to cut off a piece and pick it up with your fork. Here relying on latent actions alone is challenging, since small changes in your joystick input may accidentally move the robot away from your goal. To address this problem,

¹ Parts of this work have been published at the International Conference on Robotics and Automation (Losey et al. 2020), Robotics: Science and Systems (Jeon et al. 2020), and the International Conference on Intelligent Robots and Systems (Li et al. 2020).

in Sect. 7 we incorporate shared autonomy, where both the human and robot arbitrate control over the robot’s motion. Here the robot autonomously helps the user reach and maintain their desired high-level goals, while the user leverages latent actions to perform precise manipulation tasks (e.g., cutting, stabbing, and scooping). We show convergence bounds on the robot’s distance to the most likely goal, and develop a training procedure to ensure the human can still guide the robot to different goals if they change their mind.

Personalizing alignment Throughout the assistive eating task your joystick inputs have produced different robot motions. To guide the robot towards the tofu, you pressed the joystick up; to orient the fork for cutting, you pressed the joystick right; and to stab your piece of tofu, you pressed the joystick down. This alignment between joystick inputs and robot outputs may make sense to you—but different users will inevitably have different preferences! Accordingly, in Sect. 6 we leverage user expectations to personalize the alignment between joystick directions and latent actions. Part of this alignment process involves asking the human what they prefer (e.g., what joystick direction should correspond to scooping?). We minimize the number of queries by formalizing and leveraging the priors that humans expect when controlling robotic systems.

Conducting user studies In order to compare our approach to the state-of-the-art, we performed four user studies inspired by assistive eating tasks. Non-disabled participants teleoperated a 7-DoF robot arm using a 2-DoF joystick to reach marshmallows, make a simplified apple pie, cut tofu, and assemble dessert. We compared our latent action approach to both pre-defined mappings and shared autonomy baselines, including the HARMONIC dataset (Newman et al. 2018). We found that latent actions help users complete high-level reaching and precise manipulations with their preferred alignment, resulting in improved objective and subjective performance.

Evaluating with disabled users We applied our proposed approach with two disabled adults who leverage assistive devices when eating on a daily basis. These adults have a combined five years of experience with assistive robot arms, and typically control their arms with pre-defined mappings. In our case study both participants cut tofu and assembled a marshmallow dessert using either learned latent actions or a pre-defined mapping. Similar to our results with non-disabled users, here latent actions helped these disabled participants more quickly and accurately perform eating tasks.

Unifying previous research This paper combines our earlier work from (Losey et al. 2020; Jeon et al. 2020; Li et al. 2020). We build on these preliminary results by integrating each part into an overarching formalism (Sect. 7), demonstrating how each component relates to the overall approach, and

evaluating the resulting approach with disabled members of our target population (Sect. 10).

2 Related work

Our approach learns latent representations of dexterous robot motions, and then combines those representations with shared autonomy to facilitate both coarse reaching and precise manipulation tasks. We apply this approach to assistive robot arms—specifically for assistive eating—so that users intuitively teleoperate their robot through a spectrum of eating-related tasks.

Assistive eating Making and eating dinner without the help of a caretaker is particularly important to people living with physical disabilities (Mitzner et al. 2018; Jacobsson et al. 2000). As a result, a variety of robotic devices and algorithms have been developed for assistive eating (Brose et al. 2010; Naotunna et al. 2015). We emphasize that these devices are *high-dimensional* in order to reach and manipulate food items in 3D space (Argall 2018). When considering how to control these devices, prior works break the assistive eating task into three parts: (i) *reaching* for the human’s desired food item, (ii) *manipulating* the food item to get a bite, and then (iii) *returning* that bite back to the human’s mouth. Recent research on assistive eating has explored *automating* this process: here the human indicates what type of food they would like using a visual or audio interface, and then the robot autonomously reaches, manipulates, and returns a bite of the desired food to the user (Feng et al. 2019; Park et al. 2020; Gallenberger et al. 2019; Gordon et al. 2020; Canal et al. 2016). However, designing a fully autonomous system to handle a task as variable and personalized as eating is exceedingly challenging: consider aspects like bite size or motion timing. Indeed—when surveyed in Bhattacharjee et al. (2020)—users with physical disabilities indicated that they preferred *partially autonomy* during eating tasks, since this better enables the user to convey their own preferences. In line with these findings, we develop a partially autonomous algorithm that assists the human while letting them maintain control over the robot’s motion.

Latent representations Carefully orchestrating complex movements of high-dimensional robots is difficult for humans, especially when users are limited to a low-dimensional control interface (Bajcsy et al. 2018). Prior work has tried to prune away unnecessary control axes in a data-driven fashion through Principal Component Analysis (Ciocarlie and Allen 2009; Artemiadis and Kyriakopoulos 2010; Matrone et al. 2012). Here the robot records demonstrated motions, identifies the first few eigenvectors, and leverages these eigenvectors to map the human’s inputs to high-dimensional motions. But PCA produces a *linear* embedding—and this

embedding remains constant, regardless of where the robot is or what the human is trying to accomplish. To capture intricate *non-linear* embeddings, we turn to recent works that learn latent representations from data (Jonschkowski and Brock 2014). Robots can learn low-dimensional models of states (Pacelli and Majumdar 2020), dynamics (Watter et al. 2015; Xie et al. 2020), movement primitives (Noseworthy et al. 2020), trajectories (Co-Reyes et al. 2018), plans (Lynch et al. 2019), policies (Edwards et al. 2019), skills (Pertsch et al. 2020), and action representations for reinforcement learning (Chandak et al. 2019). One common theme across all of these works is that there are underlying patterns in high-DoF data, and the robot can succinctly capture these patterns with a low-DoF latent space. A second connection is that these works typically leverage *autoencoders* (Kingma and Welling 2014; Doersch 2016) to learn the latent space. Inspired by these latent representation methods, we similarly adapt an autoencoder model to extract the underlying pattern in high-dimensional robot motions. But unlike prior methods, we give the human control over this embedding—putting a human-in-the-loop for assistive teleoperation.

Shared autonomy Learning latent representations provides a mapping from low-dimensional inputs to high-dimensional actions. But how do we *combine* this learned mapping with control theory to ensure that the human can accurately complete their desired task? Prior work on assistive arms leverages shared autonomy, where the robot’s action is a combination of the human’s input and autonomous assistance (Dragan and Srinivasa 2013; Javdani et al. 2018; Jain and Argall 2019; Broad et al. 2020). Here the human controls the robot with a low-DoF interface (typically a joystick), and the robot leverages a *pre-defined* mapping with toggled modes to convert the human’s inputs into end-effector motion (Aronson et al. 2018; Herlant et al. 2016; Newman et al. 2018). To assist the human, the robot maintains a belief over a discrete set of possible goal objects in the environment: the robot continually updates this belief by leveraging the human’s joystick inputs as evidence in a Bayesian framework (Dragan and Srinivasa 2013; Javdani et al. 2018; Jain and Argall 2019; Gopinath et al. 2016; Nikolaidis et al. 2017). As the robot becomes increasingly confident in the human’s goal, it provides assistance to autonomously guide the end-effector towards that target. We emphasize that so far the robot has employed a pre-defined input mapping—but more related to our approach are (Reddy et al. 2018; Broad et al. 2020; Reddy et al. 2018), where the robot proposes or learns suitable dynamics models to translate user inputs to robot actions. For instance, in Reddy et al. (2018), Broad et al. (2020) the robot leverages a reinforcement learning framework to identify how to interpret and assist human inputs. Importantly, here the input space has the same number of dimensions as the action space, and so no embedding is required. We build

Table 1 Key variables and their definition

s	Robot’s state (or the world’s state)
b	Robot’s belief over high-level goals $g \in \mathcal{G}$
c	Robot’s context: we consider $c = s$, $c = (s, b)$
u	Human’s joystick input
z	Latent action commanded by the human
f	Alignment model $z = f(u, c)$
a_h	Human’s commanded high-DoF robot action
ϕ	Learned decoder $a_h = \phi(z, c)$
a_r	Autonomous assistive action
a	Robot’s action, where $a = (1 - \alpha) \cdot a_h + \alpha \cdot a_r$

upon this previous research in shared autonomy by helping the user reach and maintain their high-level goals, but we do so by leveraging latent representations to learn a mapping from *low*-DoF human inputs to *high*-DoF robot outputs.

3 Problem setting

We consider settings where a human user is teleoperating an assistive robot arm. The human interacts with the robot using a low-dimensional interface: this could be a joystick, sip-and-puff device, or brain-computer interface. We specifically focus on interfaces with a continuous control input (or an input that could be treated as continuous). For clarity, we will assume the teleoperation interface is a *joystick* throughout the rest of the paper, and we will use a joystick input in all our experiments. The assistive robot’s first objective is to *map* these joystick inputs to meaningful high-dimensional motions. But assistive robots can do more than just interpret the human’s inputs—they can also act autonomously to help the user reach and maintain their goals. Hence, the robot’s second objective is to *integrate* the learned mapping with shared autonomy. In practice, the mappings that the robot learns for one user may be counter-intuitive for another. Our final objective is to *align* the human’s joystick inputs with the latent actions, so that users can intuitively convey their desired motions through the control interface.

In this section we formalize our problem setting, and outline our proposed solutions to each objective. We emphasize the main variables in Table 1.

Task The human operator has a task in mind that they want the robot to accomplish. We formulate this task as a Markov decision process: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma, \rho_0)$. Here $s \in \mathcal{S} \subseteq \mathbb{R}^n$ is the state and $a \in \mathcal{A} \subseteq \mathbb{R}^m$ is the robot’s high-DoF action. Because we are focusing on the high-dimensional robot arm, we refer to s as the robot’s state, but in practice

the state s may contain both the robot’s arm position and the location of other objects in the environment (e.g., the position of the tofu).

The robot transitions between states according to $T(s, a)$, and receives reward $R(s)$ at each timestep. We let $\gamma \in [0, 1]$ denote the discount factor, and ρ_0 captures the initial state distribution. During each interaction the robot is not sure what the human wants to accomplish (i.e., the robot does not know R). Returning to our running example, the robot does not know whether the human wants a bite of tofu, a drink of water, or something else entirely. The human communicates their desired task through joystick inputs $u \in \mathbb{R}^d$. Because the human’s input is of lower dimension than the robot’s action, we know that $d < m$.

Dataset Importantly, this is not the first time the user has guided their robot through the process of eating dinner. We assume access to a dataset of task demonstrations: these demonstrations can be kinesthetically provided by a caregiver or collected beforehand by the disabled user with a baseline teleoperation scheme. For example, the disabled user leverages their standard, pre-defined teleoperation mapping to guide the robot through the process of reaching for objects on the table (e.g., the plate, a glass of water) and manipulating these objects (e.g., scooping rice, picking up the glass). We collect these demonstrations and employ them to train our latent action approach. Formally, we have a dataset $\mathcal{D} = \{(c_0, a_0), (c_1, a_1), \dots\}$ of context-action pairs that demonstrate high-dimensional robot actions. Notice that here we introduce the *context* $c \in \mathcal{C}$: this context captures the information available to the robot. For now we can think of the context c as the same as the robot’s state (i.e., $c = s$), but later we will explore how the robot can also incorporate its understanding of the human’s goal into this context.

Latent actions Given this dataset, we first learn a latent action space $\mathcal{Z} \subset \mathbb{R}^d$, as well as a *decoder* function $\phi : \mathcal{Z} \times \mathcal{C} \rightarrow \mathcal{A}$. Here \mathcal{Z} is a low-dimensional embedding of \mathcal{D} —we specify the dimensionality of \mathcal{Z} to match the number of degrees-of-freedom of the joystick, so that the user can directly input latent actions $z \in \mathcal{Z}$. Based on the human’s latent action z as well as the current *context* $c \in \mathcal{C}$, the robot leverages the decoder ϕ to reconstruct a high-dimensional action (see Figure 2):

$$a_h = \phi(z, c) \quad (1)$$

Notice that we use a_h here: this is because this robot action is commanded by the human’s input. Consider pressing the joystick right to cause the robot arm to cut some tofu. The joystick input is a low-DoF input u , we map this input to a latent action $z \in \mathcal{Z}$, and then leverage ϕ to decode z into a high-DoF commanded action a_h that cuts the tofu. We formalize properties of \mathcal{Z} and models for learning ϕ in Sect. 6.

Shared autonomy The human provides joystick inputs u —which we treat as latent actions z —and these latent actions map to high-dimensional robot actions a_h . But how can the robot assist the human through its own autonomous behavior? More formally, how should the robot choose autonomous actions a_r that help guide the user? Similar to recent work on shared autonomy (Jain and Argall 2019; Dragan and Srinivasa 2013; Newman et al. 2018), we define the robot’s overall action as the linear combination of a_h (the human’s commanded action) and a_r (the robot’s autonomous guidance):

$$a = (1 - \alpha) \cdot a_h + \alpha \cdot a_r \quad (2)$$

In the above, $\alpha \in [0, 1]$ parameterizes the trade-off between direct human teleoperation ($\alpha = 0$) and complete robot autonomy ($\alpha = 1$). We specifically focus on autonomous actions that help the user reach and maintain their high-level goals. Let \mathcal{G} be a discrete set of goal positions the human might want to reach (e.g., their tofu, the rice, or a glass of water), and let $g^* \in \mathcal{G}$ be the human’s true goal (e.g., the tofu). The robot assists the user towards goals it thinks are likely:

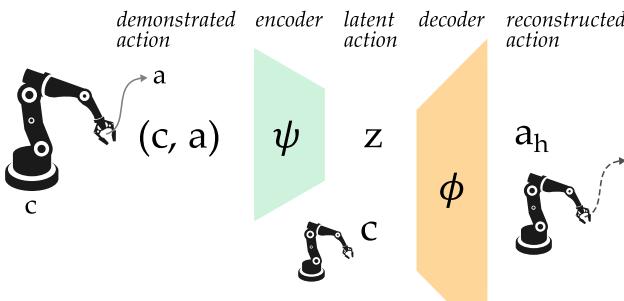
$$a_r = \sum_{g \in \mathcal{G}} b(g) \cdot (g - s) \quad (3)$$

Here a_r is a change of state (i.e., a joint velocity) that moves from s towards the mode of the inferred goal position, and b denotes the robot’s *belief*. This belief is a probability distribution over the candidate goals, where $b(g) = 1$ indicates that the robot is completely convinced that g is what the human wants. We analyze dynamics of combining Eqs. (1–3) in Sect. 7.

Alignment Recall that the human’s joystick input is u , and that our approach treats this joystick input as a latent actions z . A naive robot will simply set $z = u$. But this misses out on how different users *expect* the robot to interpret their commands. For example, let us say the assistive robot is directly above some tofu. One user might expect pressing right to cause a *stabbing* motion, while a second user expects pressing right to *cut* the tofu. To personalize our approach to match individual user expectations, we learn an alignment function:

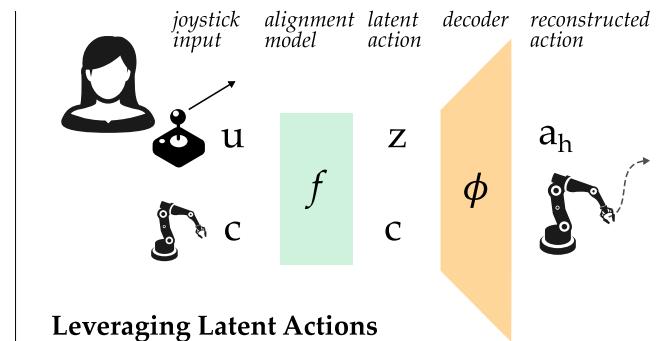
$$z = f(u, c) \quad (4)$$

Unlike Equation (1), this is *not* an embedding, since both u and z have d dimensions. But like Equation (1), the alignment model *does* depend on the robot’s current context. A user might expect pressing right to stab the tofu when they are directly above it—but when they are interacting with a glass of water, that same user expects pressing right to tilt and pour the glass. We learn f across different contexts in Sect. 6.



Learning Latent Actions

Fig. 2 Model for learning and leveraging latent actions. (Left) given a dataset of context-action pairs, we embed the robot’s high-dimensional behavior (c, a) into a low-dimensional latent space $z \in \mathcal{Z}$. The encoder and decoder are trained to ensure user-friendly properties while minimizing the error between the commanded human action a_h and the demonstrated action a . As a result, the decoder $\phi(z, c)$ provides an intu-



Leveraging Latent Actions

itive mapping from low-dimensional latent actions to high-dimensional robot actions. (Right) at run time the human controls the robot via these low-dimensional latent actions. For now we simplify the alignment model so that $z = u$, meaning that the human’s joystick inputs directly map to latent actions

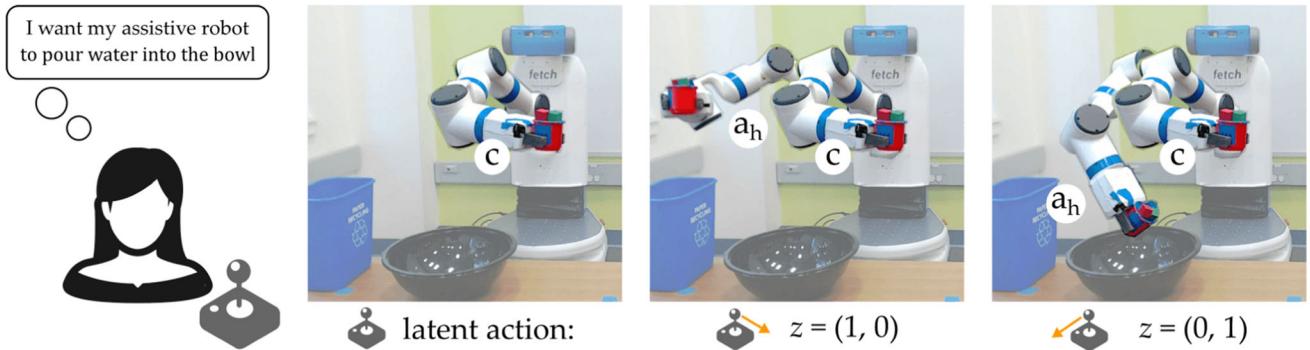


Fig. 3 Controlling an assistive robot with learned latent actions. The robot has been trained on demonstrations of pouring tasks, and learns a 2-DoF latent space. One axis of the latent space moves the cup across the table, and the other latent dimension pours the cup. This latent space satisfies our conditioning property because the decoded action a_h depends

on the current context c . This latent space also satisfies controllability because the human can leverage the two learned latent dimensions to complete the demonstrated pouring tasks (e.g., pour water into the bowl)

4 Learning latent actions

In this section we focus on learning latent actions (see Figs. 2, 3). We define latent actions as a low-dimensional embedding of high-dimensional robot actions, and we learn this embedding from the dataset \mathcal{D} of offline demonstrations. Overall, we will search for two things: i) a latent action space $\mathcal{Z} \subset \mathbb{R}^d$ that is of lower dimension than the robot’s action space $\mathcal{A} \subseteq \mathbb{R}^m$, and ii) a decoder function ϕ that maps from latent actions to robot actions. In practice, latent actions provide users a non-linear mapping for robot control: e.g., pressing right on the joystick causes the robot to perform a stabbing motion. We emphasize that these latent actions do not always have semantic meanings—they are not always “stabbing” or “cutting”—but generally embed the robot’s high-dimensional motion into a low-dimensional space.

Recall our motivating example, where the human is trying to teleoperate their assistive robot using a joystick to reach and manipulate food items. When controlling the robot, there are several *properties* that the human expects: e.g., smooth changes in the joystick input should not cause abrupt changes in robot motion, and when the human holds the joystick in a constant direction, the robot’s motion should not suddenly switch direction. In what follows, we first *formalize* the properties necessary for latent actions to be intuitive. These properties will guide our approach, and provide a principled way of assessing the usefulness of latent actions with humans-in-the-loop. Next, we will explore different *models* for learning latent actions that capture our intuitive properties.

4.1 Latent action properties

We identified four properties that user-friendly latent actions should have: conditioning, controllability, consistency, and scalability.

Conditioning Because ϕ maps from latent actions to robot actions, at first glance it may seem intuitive for ϕ to *only* depend on z , i.e., $a_h = \phi(z)$. But this quickly breaks down in practice. Imagine that you are controlling the robot arm to get some tofu: at the start of the task, you press right and left on the joystick to move the robot towards your target. But as the robot approaches the tofu, you no longer need to keep moving towards a goal; instead, you need to use those same joystick inputs to carefully align the orientation of the fork, so that you can cut off a piece. Hence, latent actions must convey different meanings in different contexts. This is especially true because the latent action space is smaller than the robot action space, and so there are more actions to convey than we can capture with z alone. We therefore introduce $c \in \mathcal{C}$, the robot’s current context, and *condition* the decoder on c , so that $a_h = \phi(z, c)$. In the rest of this section we will treat the robot’s state s as its context, so that $c = s$.

Controllability For latent actions to be useful, human operators must be able to leverage these actions to control the robot through their desired task. Recall that the dataset \mathcal{D} includes relevant task demonstrations, such as picking up a glass, reaching for the kitchen shelf, and scooping rice. We want the user to be able to control the robot through these same tasks when leveraging latent actions. Let $s_i, s_j \in \mathcal{D}$ be two states from the dataset of demonstrations, and let s_1, s_2, \dots, s_K be the sequence of states that the robot visits when starting in state $s_0 = s_i$ and taking latent actions z_1, \dots, z_K . The robot transitions between the visited states using its transition function \mathcal{T} and the learned decoder ϕ : $s_k = \mathcal{T}(s_{k-1}, \phi(z_{k-1}, s_{k-1}))$. Formally, we say that a latent action space \mathcal{Z} is *controllable* if for every pair of states (s_i, s_j) there exists a sequence of latent actions $\{z_k\}_{k=1}^K, z_k \in \mathcal{Z}$ such that $s_j = s_K$. In other words, a latent action space is controllable if it can move the robot between pairs of start and goal states from the demonstrated tasks.

Consistency Let us say you are using a one-DoF joystick to guide the robot arm along a line. When you hold the joystick to the right, you expect the robot to immediately move right—but more than that, you expect the robot to move right at every point along the line! For example, the robot should not move right for a while, then suddenly go left, and switch back to going right again. To capture this, we define a latent action space \mathcal{Z} as *consistent* if the same latent action $z \in \mathcal{Z}$ has a similar effect on how the robot behaves in nearby states. We formulate this similarity via a task-dependent metric d_M : e.g., in pouring tasks d_M could measure the orientation of the robot’s end-effector, and in reaching tasks d_M could measure

the position of the end-effector. Applying this metric, consistent latent actions should satisfy:

$$d_M(\mathcal{T}(s_1, \phi(z, s_1)), \mathcal{T}(s_2, \phi(z, s_2))) < \epsilon \quad (5)$$

when $\|s_1 - s_2\| < \delta$ for some $\epsilon, \delta > 0$. We emphasize that we do not need to know d_M for our approach; we only introduce this metric as a way of quantifying similarity.

Scalability Our last property is complementary to consistency. Thinking again about the example of teleoperating a robot along a line, when you press the joystick slightly to the right, you expect the robot to move slowly; and when you hold the joystick all the way to the right, you anticipate that the robot will move quickly. Smaller inputs should cause smaller motions, and larger inputs should cause larger motions. Formally, we say that a latent action space \mathcal{Z} is *scalable* if $\|s - s'\| \rightarrow \infty$ as $\|z\| \rightarrow \infty$, where $s' = \mathcal{T}(s, \phi(z, s))$. When put together, our consistency and scalability properties ensure that the decoder function ϕ is Lipschitz continuous.

4.2 Models for learning latent actions

Now that we have formally introduced the properties that a user-friendly latent space should satisfy, we will explore low-DoF embeddings that capture these properties; specifically, models which learn $\phi : \mathcal{Z} \times \mathcal{C} \rightarrow \mathcal{A}$ from offline demonstrations \mathcal{D} . We are interested in models that balance *expressiveness* with *intuition*: the embedding must reconstruct high-DoF actions while remaining controllable, consistent, and scalable. We assert that only models which reason over the robot’s context when decoding the human’s inputs can accurately and intuitively interpret the latent action. Our overall model structure is outlined in Fig. 2.

Reconstructing Actions Let us return to our assistive eating example: when the person applies a low-dimensional joystick input, the robot completes a high-dimensional action. We use *autoencoders* to move between these low- and high-DoF action spaces. Define $\psi : \mathcal{C} \times \mathcal{A} \rightarrow \mathcal{Z}$ as an *encoder* that embeds the robot’s behavior into a latent space, and define $\phi : \mathcal{Z} \rightarrow \mathcal{A}$ as a *decoder* that reconstructs a high-DoF robot action a_h from this latent space. Intuitively, the reconstructed robot action a_h should match the demonstrated action a . To encourage models to learn latent actions that accurately reconstruct high-DoF robot behavior, we incorporate the reconstruction error $\|a - a_h\|^2$ into the model’s loss function. Let \mathcal{L} denote the *loss function* our model is trying to minimize; when we only focus on reconstructing actions, our loss function is:

$$\mathcal{L} = \|a - \phi(\psi(c, a))\|^2 \quad (6)$$

Both principal component analysis (**PCA**) and autoencoder (**AE**) models minimize this loss function.

Regularizing Latent Actions When the user slightly tilts the joystick, the robot should not suddenly cut the entire block of tofu. To better ensure this consistency and scalability, we incorporate a *normalization* term into the model’s loss function. Let us define $\psi : \mathcal{C} \times \mathcal{A} \rightarrow \mathbb{R}^d \times \mathbb{R}_+^d$ as an encoder that outputs the mean μ and covariance σ over the latent action space. We penalize the divergence between this latent action space and a normal distribution: $KL(\mathcal{N}(\mu, \sigma) \parallel \mathcal{N}(0, 1))$. When we incorporate this normalizer, our loss function becomes:

$$\mathcal{L} = \|a - \phi(z)\|^2 + \lambda \cdot KL[\mathcal{N}(\mu, \sigma) \parallel \mathcal{N}(0, 1)] \quad (7)$$

Variational autoencoder (**VAE**) models Kingma and Welling (2014); Doersch (2016) minimize this loss function by trading-off between reconstruction error and normalization.

Conditioning on State Importantly, we recognize that the *meaning* of the human’s joystick input often depends on what part of the task the robot is performing. When the robot is above a block of tofu, pressing down on the joystick indicates that the robot should stab the food; but—when the robot is far away from the tofu—it does not make sense for the robot to stab! So that robots can associate meanings with latent actions, we *condition* the interpretation of the latent action on the robot’s current context. Define $\phi : \mathcal{Z} \times \mathcal{C} \rightarrow \mathcal{A}$ as a decoder that now makes decisions based on both z and c . Leveraging this conditioned decoder, our final loss function is:

$$\mathcal{L} = \|a - \phi(z, c)\|^2 + \lambda \cdot KL[\mathcal{N}(\mu, \sigma) \parallel \mathcal{N}(0, 1)] \quad (8)$$

We expect that conditional autoencoders (**cAE**) and conditional variational autoencoders (**cVAE**) which use ϕ will learn more expressive and controllable actions than their non-context conditioned counterparts. Note that cVAEs minimize Eq. (8), while cAEs do not include the normalization term (i.e., $\lambda = 0$).

Relation to Properties Models trained to minimize the listed loss functions are encouraged to satisfy our user-friendly properties. For example, in Eq. (8) the decoder is *conditioned* on the current context, while minimizing the reconstruction loss $\|a - \phi(z, c)\|^2$ ensures that the robot can reproduce the demonstrations, and is therefore *controllable*. Enforcing *consistency* and *scalability* are more challenging—particularly when we do not know the similarity metric d_M —but including the normalization term prevents the latent space from assigning arbitrary and irregular values to z . To better understand how these models enforce our desired properties, we conduct a set of controlled simulations in Sect. 8.1.

5 Combining latent actions with shared autonomy

The latent actions learned in Sect. 4 provide an expressive mapping between low-dimensional user inputs and high-dimensional robot actions. But controlling an assistive robot with latent actions alone still presents a challenge: any *imprecision* or *noise* in either the user’s inputs or latent space is reflected in the decoded actions. Recall our eating example: at the start of the task, the human uses latent actions to guide the robot towards their *high-level goal* (i.e., reaching the tofu). Once the robot is close to the tofu, however, the human no longer needs to control reaching motions—instead, the human leverages latent actions to *precisely manipulate* the tofu, performing *low-level* cutting and stabbing tasks. Here the human’s inputs should not unintentionally cause the robot arm to drift away from the tofu or suddenly jerk into the table. Instead, the robot should *maintain* the human’s high-level goal. In this section we incorporate shared autonomy alongside latent actions: this approach assists the human towards their high-level goals, and then maintains these goals as the human focuses on low-level manipulation. We visualize shared autonomy with learned latent actions in Fig. 4.

5.1 Latent actions with shared autonomy

We first explain how to combine latent actions with shared autonomy. Remember that the human’s joystick input is u and the latent action is z . For now we assume some pre-defined mapping from u to z (i.e., $z = u$) so that the human’s joystick inputs are treated as latent actions. In the last section we learned a decoder $a_h = \phi(z, c)$, where the output of this decoder is a high-dimensional robot action *commanded* by the human. Here we combine this commanded action with a_r , an autonomous *assistive* action that helps the user reach and maintain their high-level goals.

Belief over Goals Similar to Dragan and Srinivasa (2013); Javdani et al. (2018); Gopinath et al. (2016), we assume access to a discrete set of high-level goals \mathcal{G} that the human may want to reach. Within our eating scenario these goals are food items (e.g., the tofu, rice, a plate, marshmallows). Although the robot knows which goals are possible, the robot does not know the human’s current goal $g^* \in \mathcal{G}$. We let $b = P(g \mid s^{0:t}, u^{0:t})$ denote the robot’s belief over this space of candidate goals, where $b(g) = 1$ indicates that the robot is convinced that g is the human’s desired goal. Here $s^{0:t}$ is the history of states and $u^{0:t}$ is the history of human inputs: we use Bayesian inference to update the robot’s belief b given the human’s past decisions:

$$b^{t+1}(g) \propto P(u^t \mid s^t, g) \cdot b^t(g) \quad (9)$$

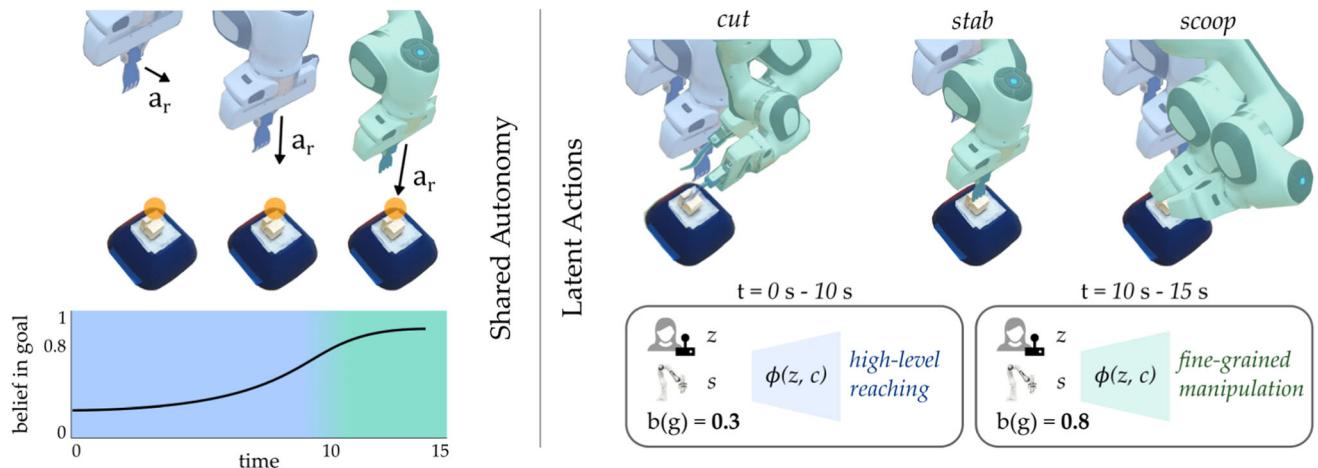


Fig. 4 Shared autonomy with learned latent actions. (Left) as the human teleoperates the robot towards their desired goal, the robot’s belief in that goal increases, and the robot selects assistive actions a_r to help the human autonomously reach and maintain their high-level goal. (Right) the meaning of the latent actions changes as a function of the robot’s

belief. At the start of the task—when the robot is not sure about any goal—the latent actions z produce high-level reaching motions (shown in blue). As the robot becomes confident in the human’s goal the meaning of the latent actions becomes more refined, and z increasingly controls fine-grained manipulation (shown in green)

This Bayesian inference approach for updating b is explored by prior work on shared autonomy Jain and Argall (2019).

Importantly, the *meaning* of the human’s joystick inputs changes as a function of the robot’s belief. Imagine that you are using a 1-DoF joystick to get the tofu in our eating example. At the start of the task—when the robot is unsure of your goal—you press left and right on the joystick to move towards your high-level goal. Once you’ve reached the tofu—and the robot is confident in your goal—you need to use those same joystick inputs to carefully align the orientation of the fork. In order to learn latent action spaces that can continuously alternate along a spectrum of high-level goals and fine-grained preferences, we now condition ϕ on the robot’s current state *as well as its belief*. Hence, instead of $c = s$, we now have $c = (s, b)$. Conditioning on belief enables the meaning of latent actions to change based on the robot’s confidence. As a result of this proposed structure, latent actions *purely indicate the desired goal* when the robot is unsure; and once the robot is confident about the human’s goal, latent actions gradually change to *convey the precise manipulation*. We note that b is available when collecting demonstrations \mathcal{D} , since the robot can compute its belief using the Bayesian update above based on the demonstrated trajectory. Take a demonstration that moves the robot to the tofu: initially the robot has a uniform belief over goals, but as the demonstration moves towards the tofu, the robot applies Eq. (9) to increase its belief b over the tofu goal.

Shared Autonomy Recall that the robot applies assistance via action a_r in Eq. (2). In order to assist the human, the robot needs to understand the human’s *intent*—i.e., which goal they want to reach. The robot’s understanding of the

human’s intended goal is captured by belief b , and we leverage this belief to select an assistive action a_r . As shown in Eq. (3), the robot selects a_r to guide the robot towards each discrete goal $g \in \mathcal{G}$ in proportion to the robot’s confidence in that goal². Combining these equations with our learned latent actions, we find the robot’s overall action a :

$$a = (1 - \alpha) \cdot \phi(z, c) + \alpha \cdot \sum_{g \in \mathcal{G}} b(g) \cdot (g - s) \quad (10)$$

Recall that $\alpha \in [0, 1]$ arbitrates between human control ($\alpha = 0$) and assistive guidance ($\alpha = 1$). In practice, if the robot has a uniform prior over which morsel of food the human wants to eat, a_r guides the robot to the center of these morsels. And—when the human indicates a desired morsel— a_r moves the robot towards that target before maintaining the target position.

5.2 Reaching and changing goals

In Eq. (10) we incorporated shared autonomy with latent actions to tackle assistive eating tasks that require high-level reaching and precise manipulation. Both latent actions and shared autonomy have an *independent* role within this method: but how can we be sure that the *combination* of these tools will remain effective? Returning to our eating example—if the human inputs latent actions, will shared autonomy correctly guide the robot to the desired morsel of food? And what if the human has multiple goals in mind

² Our approach is not tied to this particular instantiation of shared autonomy. Other instances of shared autonomy can similarly be used.

(e.g., getting a chip and then dipping it in salsa)—can the human leverage latent actions to change goals even when shared autonomy is confident in the original goal?

Converging to the Desired Goal We first explore how our approach ensures that the human reaches their desired goal. Consider the Lyapunov function:

$$V(t) = \frac{1}{2} \|e(t)\|^2, \quad e(t) = g^* - s(t) \quad (11)$$

where e denotes the error between the robot's current state s and the human's goal g^* . We want the robot to choose actions that minimize Eq. (11) across a spectrum of user skill levels and teleoperation strategies. Let us focus on the common setting in which s is the robot's joint position and a is the joint velocity, so that $\dot{s}(t) = a(t)$. Taking the derivative of Eq. (11) and substituting in this transition function, we reach³:

$$\dot{V}(t) = -\frac{1}{2} e^\top \left[\phi(z, c) + \sum_{g \in \mathcal{G}} b(g) \cdot (g - s) \right] \quad (12)$$

We want Eq. (12) to be negative, so that V (and thus the error e) decrease over time. A sufficient condition for $\dot{V} < 0$ is:

$$b(g^*) \cdot \|e\| > \|\phi(z, c)\| + \sum_{g \in \mathcal{G}'} b(g) \cdot \|g - s\| \quad (13)$$

where \mathcal{G}' is the set of all goals except g^* . As a final step, we bound the magnitude of the decoded action, such that $\|\phi(\cdot)\| < \sigma_h$, and we define σ_r as the distance between s and the furthest goal: $\sigma_r = \max_{g \in \mathcal{G}'} \|g - s\|$. Now we have $\dot{V} < 0$ if:

$$b(g^*) \cdot \|e\| > \sigma_h + (1 - b(g^*)) \cdot \sigma_r \quad (14)$$

We define $\delta := \sigma_h + (1 - b(g^*)) \cdot \sigma_r$. We therefore conclude that our approach in Eq. (10) yields *uniformly ultimately bounded stability* about the human's goal, where δ affects the *radius* of this bound Spong et al. (2006). As the robot's confidence in g^* increases, $\delta \rightarrow \sigma_h$, and the robot's error e decreases so long as $\|e(t)\| > \sigma_h$. Intuitively, this guarantees that the robot will move to some ball around the human's goal g^* (even if we treat the human input as a disturbance), and the radius of that ball decreases as the robot becomes more confident.

Changing Goals Our analysis so far suggests that the robot becomes *constrained* to a region about the most likely goal. This works well when the human correctly conveys their intentions to the robot—but what if the human makes a mistake, or changes their mind? How do we ensure that the robot

³ For notational simplicity we choose $\alpha = 0.5$, so that both human and robot inputs are equally weighted. Our results generalize to other α .

is not *trapped* at an undesired goal? Re-examining Eq. (14), it is key that—in every context c —the human can convey sufficiently large actions $\|\phi(z, c)\|$ towards their preferred goal, ensuring that σ_h does not decrease to zero. Put another way, the human must be able to *increase* the radius of the bounding ball, reducing the constraint imposed by shared autonomy.

To encourage the robot to learn latent actions that increase this radius, we introduce an additional term into our model's loss function \mathcal{L} . We reward the robot for learning latent actions that have high *entropy* with respect to the goals; i.e., in a given context c there exist latent actions z that cause the robot to move towards *each* of the goals $g \in \mathcal{G}$. Define $p_c(g)$ as proportional to the total *score* η accumulated for goal g :

$$p_c(g) \propto \sum_{z \in \mathcal{Z}} \eta(g, c, z) \quad (15)$$

where the score function η indicates how well action z taken from context c conveys the intent of moving to goal g , and the distribution p_c over \mathcal{G} captures the proportion of latent actions z at context c that move the robot toward each goal. Intuitively, p_c captures the comparative ease of moving toward each goal: when $p_c(g) \rightarrow 1$, the human can easily move towards goal g since *all* latent actions at c induce movement towards goal g and consequently, *no* latent actions guide the robot towards any other goals. We seek to *avoid* learning latent actions where $p_c(g) \rightarrow 1$, because in these scenarios the teleoperator *cannot* correct their mistakes or move towards a different goal. Recall from Sect. 4 that the model should minimize the reconstruction error while regularizing the latent space. We now argue that the model should additionally maximize the Shannon entropy of p , so that the loss function becomes:

$$\begin{aligned} \mathcal{L} = & \|a - \phi(z, c)\|^2 + \lambda_1 \cdot KL[\mathcal{N}(\mu, \sigma) \parallel \mathcal{N}(0, 1)] \\ & + \lambda_2 \cdot \sum_{g \in \mathcal{G}} p(g) \log p(g) \end{aligned} \quad (16)$$

Here the hyperparameter $\lambda_2 > 0$ determines how much importance is assigned to maximizing the entropy over goals. When combining shared autonomy with latent actions, we employ this loss function to train the decoder ϕ from dataset \mathcal{D} .

6 Aligning latent actions with user preferences

In Sect. 6 we learned latent actions, and in Sect. 7 we combined these latent actions with shared autonomy to handle precise manipulation tasks. Throughout these sections we treated the human's joystick inputs as the latent actions (i.e.,

$z = u$) since both the joystick inputs and the latent actions are the same dimensionality. However, different users have different expectations for how the robot will interpret their inputs! Imagine that the shared autonomy has assisted us to the tofu, and now we want to control the robot through a cutting motion. One user expects $u = \text{down}$ to cause the robot to cut, but another person thinks $u = \text{down}$ should cause a stabbing motion. Accordingly, in this section we learn a *personalized* alignment $z = f(u, c)$ that converts the human’s joystick inputs u to their preferred latent action z (see Fig. 5). Our goal is to make the robotic system easier to control: instead of forcing the human to adapt to ϕ , we want the robot to adapt to the user’s preferences (without fundamentally changing the latent action space or decoder ϕ).

To learn the human’s preference we will query the user, showing them example robot motions and then asking them for the corresponding joystick input. But training our alignment model f may require a large number of motion-joystick pairs, particularly in complex tasks where the user must leverage the same joystick input to accomplish several things. It is impractical to ask the human to provide all of these labels. Accordingly, to address the challenge of insufficient training data, we employ a *semi-supervised* learning method (Chapelle et al. 2006). In this section we first outline our approach, and then formulate a set of intuitive priors that facilitate semi-supervised learning from limited human feedback.

6.1 Alignment model

Recall from Eq. (4) that we seek to learn a function approximator $f : \mathcal{U} \times \mathcal{C} \rightarrow \mathcal{Z}$. Importantly, this alignment model is conditioned on the current context $c \in \mathcal{C}$. Consider the person in our motivating example, who is using a 2-axis joystick to control a high-DoF assistive robot arm to reach and cut tofu. The user’s preferred way to control the robot is unclear: what does the user mean if they push the joystick right? When the robot is left of the tofu, the user might intend to move the robot towards the tofu—but when the robot is directly above the tofu, pressing right now indicates that the robot should rotate and start a cutting motion! This mapping from the user input to intended action is not only person dependent, but it is also *context dependent*. In practice, this context dependency prevents us from learning a single transformation to uniformly apply across the robot’s workspace; instead, we need an intelligent strategy for understanding the human’s preferences in different contexts.

Model To capture this interdependence we employ a general Multi-Layer Perceptron (MLP). The MLP f takes in the current user input u^t and context c^t , and outputs a latent action z^t . Combining f with our latent action model, we now have

a *two-step* mapping between the human’s low-dimensional input and the human’s high-dimensional command:

$$a_h^t = \phi(z^t, c^t) = \phi(f(u^t, c^t), c^t) \quad (17)$$

When using our alignment model *online*, we get the human’s commanded action using Eq. (17), and then combine this with shared autonomy to provide the overall robot action a . But *offline*—when we are learning f —we set $a = a_h$, so that the robot directly executes the human’s commanded action. This disentangles the effects of shared autonomy and latent actions, and lets us focus on learning the preferred mapping from joystick inputs u to robot actions a . Given that the robot takes action a^t at the current timestep t , the state s^{t+1} at the next timestep follows our transition model: $s^{t+1} = T(s^t, a^t)$. Letting $a = a_h$, and plugging in Equation (17), we get the following relationship between joystick inputs and robot motion:

$$s^{t+1} = T(s^t, \phi(f(u^t, c^t), c^t)) = T(s^t, u^t) \quad (18)$$

Our objective is to learn f so that $s^{t+1} = T(s^t, u^t)$ matches the human’s expectations. The overall training process is visualized in Fig. 6.

Loss function We train f to minimize the loss function $\mathcal{L}_{\text{align}}$. We emphasize that this loss function (used for training the alignment model f) is different than the loss function described in Sects. 6 and 7 (which was used for training the

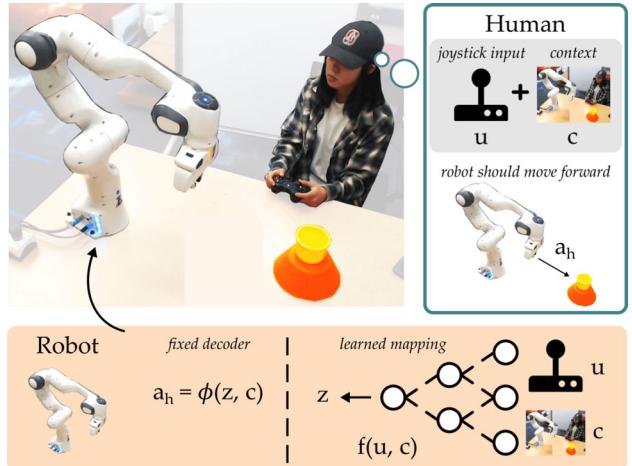


Fig. 5 Overview of alignment model. The human has in mind a preferred mapping between their joystick inputs u and their commanded actions a_h . We break this into a two step process: *aligning* the joystick inputs with latent actions z , and then decoding z into a high-DoF action a_h . In previous sections we focused on the decoder ϕ ; now we learn a personalized alignment model f . The robot learns f offline in a *semi-supervised* manner by combining labeled queries with intuitive priors that capture the human’s underlying expectations of how the control mapping should behave

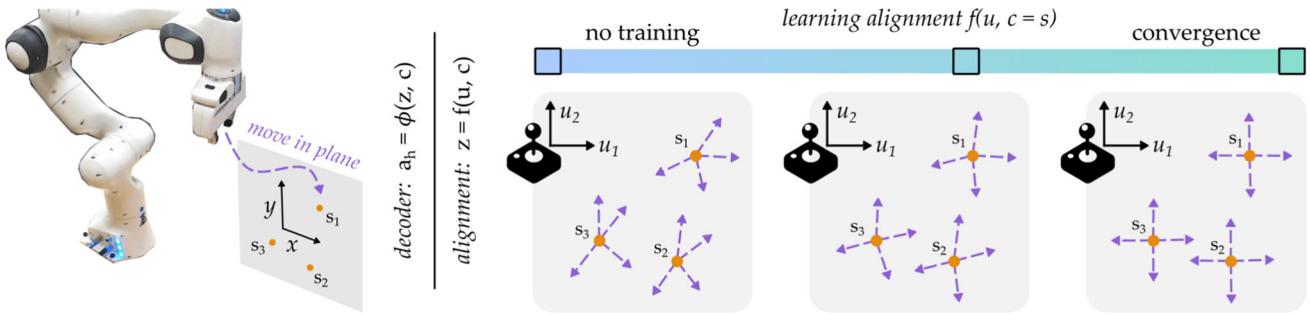


Fig. 6 Training our alignment model $z = f(u, c)$. Here the context c is equal to the robot’s state s . (Left) the example task is to move the robot’s end-effector in a 2D plane, and the current user prefers for the robot’s end-effector motion to align with their joystick axes, so that u_1 moves the robot in the x -axis and u_2 moves the robot in the y -axis. (Right) we take snapshots at three different points during training, and plot how

the robot actually moves when the human presses up, down, left, and right. Note that this alignment is context dependent. As training progresses, the robot learns the alignment f , and the robot’s motions are gradually and consistently pushed to match with the human’s individual preferences

decoder ϕ). Importantly, $\mathcal{L}_{\text{align}}$ must capture the individual user’s expected joystick mapping—and to understand what the user expects, we start by asking a set of questions. In each separate query, the robot starts in a state s^t and moves to some state s^* . We then ask the user to *label* this motion with their preferred joystick input u , resulting in the labeled data tuple (s^t, u^t, s^*) . For instance, the robot arm starts above the tofu, and then stabs down to break off a piece: you might label this motion by holding down on the joystick (i.e., $u = \text{down}$).

Given a start state s^t and input u^t from the user’s labeled data, the robot learns f to minimize the distance between $T(s^t, u^t)$ and s^* . Letting N denote the number of queries that the human has answered, and letting d be the distance metric, our alignment function should minimize:

$$L_{\text{sup}} = \frac{1}{N} \sum_{i=1}^N d(s^{*,i}, T(s^i, u^i)) \quad (19)$$

If we could ask the human as many questions as necessary, then $\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{sup}}$, and our alignment function only needs to minimize the supervised loss. But collecting this large dataset is impractical. Accordingly, to minimize the number of questions the human must answer, we introduce additional loss terms in $\mathcal{L}_{\text{align}}$ that capture underlying priors in human expectations.

6.2 Reducing human data with intuitive priors

Our insight here is that humans share some underlying expectations of how the control mapping should behave (Jonschkowski and Brock 2014). We will formulate these common expectations—i.e., *priors*—as loss terms that f minimizes within semi-supervised learning.

When introducing these priors, it helps to refine our notation. Recall that s is the system state: here we use s

to specifically refer to the robot’s joint position, and we denote the forward kinematics of the robot arm as $x = \Psi(s)$. End-effector pose x is particularly important, since humans often focus on the robot’s gripper during eating tasks. When the human applies joystick input u at state s , the corresponding change in end-effector pose x is: $\Delta x = \Psi(T(u, s)) - \Psi(s)$. With these definitions in mind, we argue an intuitive controller should satisfy the properties listed below. We emphasize that—although these properties share some common themes with the latent action properties from Sect. 6—the purpose of these properties is different. When formalizing the properties for latent actions we focused on enabling the human to complete tasks using these latent actions. By contrast, here we focus on intuitive and task-agnostic expectations for controller mappings.

Proportionality The amount of change in the position and orientation of the robot’s end-effector should be proportional to the scale of the human’s input. In other words, for scalar α , we expect:

$$\alpha \cdot |\Psi(T(u, s)) - \Psi(s)| = |\Psi(T(\alpha \cdot u, s)) - \Psi(s)|$$

We accordingly define the proportionality loss $\mathcal{L}_{\text{prop}}$ as:

$$\mathcal{L}_{\text{prop}} = \|\Psi(T(\alpha \cdot u, s)) - \Psi(s) - \alpha \cdot \Delta x\|^2 \quad (20)$$

where α is sampled from our range of joystick inputs.

Reversability If a joystick input u makes the robot move *forward* from s_1 to s_2 , then the opposite input $(-u)$ should move the robot *back* from s_2 to its original end-effector position. In other words, we expect:

$$\Psi(s_1) = \Psi\left(T(-u, T(u, s_1))\right)$$

This property ensures users can recover from their mistakes. We define the reversability loss $\mathcal{L}_{\text{reverse}}$ as:

$$\mathcal{L}_{\text{reverse}} = \|\Psi(s) - \Psi(T(-u, T(u, s)))\|^2 \quad (21)$$

Here $\Psi(s)$ is the current position and orientation of the robot's end-effector, and the right term is the pose of the end-effector after executing human input u followed by the opposite input $(-u)$.

Consistency The same input taken at nearby states should lead to similar changes in robot pose. We previously discussed a similar property in Sect. 6 when formalizing latent actions. Here we specifically focus on the input-output relationship between joystick input u and end-effector position $x = \Psi(s)$:

$$\begin{aligned} \Delta x_1 &= \|\Psi(T(u, s_1)) - \Psi(s_1)\|^2 \\ \Delta x_2 &= \|\Psi(T(u, s_2)) - \Psi(s_2)\|^2 \end{aligned}$$

We expect $\|\Delta x_1 - \Delta x_2\| \rightarrow 0$ as $\|s_1 - s_2\| \rightarrow 0$. Consistency prevents sudden changes in the alignment mapping. We define the consistency loss \mathcal{L}_{con} as:

$$\mathcal{L}_{\text{con}} = \|\Delta x(s_1) - \Delta x(s_2)\| \cdot \exp\{-\gamma \|s_1 - s_2\|\} \quad (22)$$

When the hyperparameter $\gamma \rightarrow 0$, the robot only enforces consistency at local states, and when $\gamma \rightarrow \infty$, the robot tries to enforce consistency at all states.

Semi-supervised learning When learning our alignment model f , we first collect a batch of robot motions (s, s^*) . The human labels N of these (start-state, end-state) pairs with their preferred joystick input u , so that we have labeled data (s, u, s^*) . We then train the alignment model to minimize the supervised loss for the *labeled* data, as well as the semi-supervised loss for the *unlabeled* data. Hence, the cumulative loss function is:

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{sup}} + \lambda_1 \mathcal{L}_{\text{prop}} + \lambda_2 \mathcal{L}_{\text{reverse}} + \lambda_3 \mathcal{L}_{\text{con}} \quad (23)$$

Importantly, incorporating these different loss terms—which are inspired by human priors over controllable spaces (Jonschkowski and Brock 2014)—enables the robot to generalize the labeled human data (which it performs supervised learning on) to unlabeled states (which it can now perform semi-supervised learning on).

7 Algorithm

Sections 4, 5, and 6 developed parts of our approach. Here we put these pieces together to present our general algorithm for controlling assistive eating robots with learned latent actions.

Algorithm 1 Latent Control of Assistive Robots

Offline:

- 1: Select a discrete set of goals \mathcal{G}
- 2: Collect a dataset $\mathcal{D} = \{(c_0, a_0), (c_1, a_1), \dots\}$ from kinesthetic demonstrations
- 3: Train latent action model ϕ to minimize \mathcal{L} on \mathcal{D}
- 4: Query the user to label example motions $\{(s, s^*)\}$ with their preferred joystick direction u
- 5: Train alignment model f to minimize loss $\mathcal{L}_{\text{align}}$ using labeled and unlabeled motions $\{(s, s^*)\}$

Online, at each timestep t :

- 6: $z^t \leftarrow f(u^t, c^t)$ \triangleright Align latent action with user input
- 7: $a_h^t \leftarrow \phi(z^t, c^t)$ \triangleright Decode z^t to high-DoF action
- 8: $a_r^t \leftarrow \sum_{g \in \mathcal{G}} b^t(g) \cdot (g - s^t)$ \triangleright get robot assistance
- 9: $a^t \leftarrow (1 - \alpha) \cdot a_h^t + \alpha \cdot a_r^t$ \triangleright blend both a_h and a_r
- 10: $b^{t+1} \propto P(u^t | c^t, g) P(g)$ \triangleright update belief over goals
- 11: $s^{t+1} \sim \mathcal{T}(s^t, a^t)$ \triangleright take action

Our approach is summarized in Algorithm 1 and explained below.

Given an assistive eating scenario, we start by identifying the food items and other potential goals that the human may want to reach (Line 1). We then collect high-dimensional kinesthetic demonstrations, where a caretaker backdrives the robot through task-related motions that interact with these potential goals (Line 2). Leveraging the properties and models from Sect. 6, we then train our latent action space and learn decoder ϕ (Line 3). Next, we show the user example robot motions—e.g., by sampling values of z —and ask the user to label these motions with their preferred joystick input (Line 4). Applying the priors developed in Sect. 6, we generalize from a small number of human labels to learn the alignment f between joystick inputs and latent actions (Line 5).

Once we have learned ϕ and f , we are ready for the human-in-the-loop. At each timestep the human presses their low-DoF joystick to provide input u . We find the latent action z that is aligned with the human's input (Line 6), and then decode that low-DoF latent action to get a high-DoF robot action a_h (Line 7). In order to help the human reach and maintain their high-level goals, we incorporate the shared autonomy approach from Sect. 7. Shared autonomy selects an assistive action a_r based on the current belief over the human's goal (Line 8), and the robot blends a_r and a_h to take overall action a (Line 9). Finally, the robot applies Bayesian inference to update its understanding of the human's desired goal based on their joystick input (Line 10). We repeat this process until the human has finished eating.

How Practical is Our Approach? One concern is the amount of data required to learn latent actions. In all of the studies reported below—where the assistive robot makes a

mock-up apple pie, assembles dessert, and cuts tofu—the robot was trained with a maximum of **twenty minutes** of kinesthetic demonstrations, and all training was done **on-board** the robot computer. We recognize that this short training time is likely due to the structure of the cVAE model used in these tasks and may not hold true in general; however, this easy implementation holds promise for future use. In the following sections, we demonstrate the objective and subjective benefits of Algorithm 1, as well as highlighting some of its shortcomings.

Where Do the Goals Come From? Another question is how the robot detects the discrete set of goals \mathcal{G} that the human may want to reach. Here we turn to *perception*, where recent assistive eating work shows how robot arms can estimate the pose of various objects of interest (Feng et al. 2019; Park et al. 2020). Determining which objects are potential goals is simplified in eating settings, since the target items are largely consistent (e.g., food items, cups, plates, and bowls). The location of these goals is included in the state s and the robot uses this information when decoding the human’s joystick inputs. Although not covered in this paper, it is also possible to condition latent actions directly on the robot’s perception, so that s becomes the visual inputs (Karamcheti et al. 2021).

8 Simulations

We performed three separate simulations, one for each key aspect of our proposed method. First we leverage different autoencoder models to learn latent actions, and determine which types of models best capture the user-friendly properties formalized in Sect. 6. Our second simulation then compares learned latent actions alone to latent actions with shared autonomy (Sect. 7), and focuses on how shared autonomy helps users reach, maintain, and change their high-level goals. Finally, we learn the alignment model from Sect. 6 between joystick inputs and latent actions. We compare versions of our semi-supervised approach with intuitive priors, and see how these priors improve the alignment when we only have access to limited and imperfect human feedback. All three simulations were performed in controlled conditions with simulated humans and simulated or real robot arms. These simulated humans chose joystick inputs according to mathematical models of human decision making, as detailed below.

8.1 Do learned latent actions capture our user-friendly properties?

Here we explore how well our proposed models for learning latent actions capture the user-friendly properties formalized

in Sect. 6. These properties include *controllability*, *consistency*, and *scalability*.

Setup We simulate one-arm and two-arm planar robots, where each arm has $n = 5$ degrees-of-freedom. The state $s \in \mathbb{R}^n$ is the robot’s joint position, and the action $a \in \mathbb{R}^n$ is the robot’s joint velocity. Hence, the robot transitions according to: $s^{t+1} = s^t + a^t \cdot dt$, where dt is the step size. Demonstrations consist of trajectories of state-action pairs: in each of different simulated tasks, the robot trains with a total of 10,000 state-action pairs.

Tasks The simulated robots perform four different tasks.

1. *Sine*: one 5-DoF robot arm moves its end-effector along a sine wave with a 1-DoF latent action
2. *Rotate*: two 5-DoF robot arms are holding a box, and rotate that box about a fixed point using a 1-DoF latent action
3. *Circle*: one 5-DoF robot moves back and forth along circles of different radii with a 2-DoF latent action
4. *Reach*: one 5-DoF robot arm reaches from a start location to a goal region with a 1-DoF latent action

Model details We test latent action models which minimize the different loss function described in Sect. 6. Specifically, we test:

- Principal Component Analysis (**PCA**)
- Autoencoders (**AE**)
- Variational autoencoders (**AE**)
- Conditioned autoencoders (**cAE**)
- Conditioned variational autoencoders (**cVAE**)

The encoders and decoders contain between two and four linear layers (depending on the task). The loss function is optimized using Adam with a learning rate of $1e^{-2}$. Within the VAE and cVAE, we set the normalization weight < 1 to avoid posterior collapse.

Dependent measures To determine *accuracy*, we measure the mean-squared error between the intended actions a and reconstructed actions \hat{a} on a test set of state-action pairs (s, a) drawn from the same distribution as the training set.

To test model *controllability*, we select pairs of start and goal states (s_i, s_j) from the test set, and solve for the latent actions z that minimize the error between the robot’s current state and s_j . We then report this minimum state error.

We jointly measure *consistency* and *scalability*: to do this, we select 25 states along the task, and apply a fixed grid of latent actions z_i from $[-1, +1]$ at each state. For every (s, z) pair we record the distance and direction that the end-effector

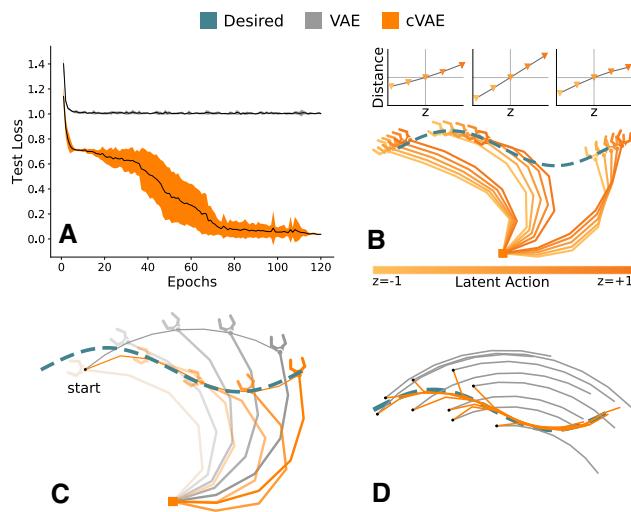


Fig. 7 Results for the *Sine* task. (A) Mean-squared error between intended and reconstructed actions normalized by PCA test loss. (B) Effect of the latent action z at three states along the sine wave for the cVAE model. Darker colors correspond to $z > 0$ and lighter colors signify $z < 0$. Above we plot the distance that the end effector moves along the sine wave as a function of z at each state. (C) Rollout of robot behavior when applying a constant latent input $z = +1$, where both VAE and cVAE start at the same state. (D) End-effector trajectories for multiple rollouts of VAE and cVAE

travels (e.g., the direction is $+1$ if the end-effector moves right). We then find the best-fit line relating z to distance times direction, and report its R^2 error.

Our results are averaged across 10 trained models of the same type, and are listed in the form $mean \pm SD$.

Hypotheses We have the following two hypotheses:

- H1.** Only latent action models conditioned on the context will accurately reconstruct actions from low-DoF inputs.
- H2.** Conditioned autoencoders and conditioned variational autoencoders will learn a latent space that is controllable, consistent, and scalable.

Sine task This task and our results are shown in Fig. 7. We find that conditioning the decoder on the current context, i.e., $\phi(z, c)$, greatly improves *accuracy* when compared to the PCA baseline, i.e., $\phi(z)$. Here AE and VAE incur $98.0 \pm 0.6\%$ and $100 \pm 0.8\%$ of the PCA loss, while cAE and cVAE obtain $1.37 \pm 1.2\%$ and $3.74 \pm 0.4\%$ of the PCA loss, respectively.

We likewise observe that cAE and cVAE are more *controllable* than their alternatives. When using the learned latent actions to move between 1000 randomly selected start and end states along the sine wave, cAE and cVAE have an average end-effector error of 0.05 ± 0.01 and 0.10 ± 0.01 . Models

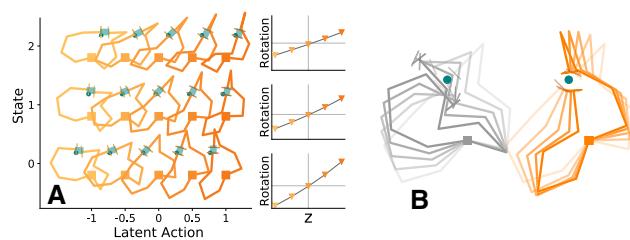


Fig. 8 Results for the *Rotate* task. (A) The robot uses two arms to hold a light blue box, and learns to rotate this box around the fixed point shown in teal. Each state corresponds to a different fixed point, and positive z causes counterclockwise rotation. On right we show how z affects the rotation of the box at each state. (B) rollout of the robot's trajectory when the user applies $z = +1$ for VAE and cVAE models, where both models start in the same state. Unlike the VAE, the cVAE model coordinates its two arms

without state conditioning—PCA, AE, and VAE—have average errors 0.90 , 0.94 ± 0.01 , and 0.95 ± 0.01 .

When evaluating *consistency* and *scalability*, every tested model has a roughly linear relationship between latent actions and robot behavior: PCA has the highest $R^2 = 0.99$, while cAE and cVAE have the lowest $R^2 = 0.94 \pm 0.04$ and $R^2 = 0.95 \pm 0.01$.

Rotate task We summarize the results for this two-arm task in Fig. 8. Like in the *Sine* task, the models conditioned on the current context are more *accurate* than their non-conditioned counterparts: AE and VAE have $28.7 \pm 4.8\%$ and $38.0 \pm 5.8\%$ of the PCA baseline loss, while cAE and cVAE reduce this to $0.65 \pm 0.05\%$ and $0.84 \pm 0.07\%$. The context conditioned models are also more *controllable*: when using the learned z to rotate the box, AE and VAE have $56.8 \pm 9\%$ and $71.5 \pm 8\%$ as much end-effector error as the PCA baseline, whereas cAE and cVAE achieve $5.4 \pm 0.1\%$ and $5.9 \pm 0.1\%$ error.

When testing for *consistency* and *scalability*, we measure the relationship between the latent action z and the change in orientation for the end-effectors of both arms (i.e., ignoring their location). Each model exhibits a linear relationship between z and orientation: $R^2 = 0.995 \pm 0.004$ for cVAE and $R^2 = 0.996 \pm 0.002$ for cVAE. In other words, there is an approximately linear mapping between z and the orientation of the box that the two arms are holding.

Circle task Next, consider the one-arm task in Fig. 9 where the robot has a 2-DoF latent action space. We here focus on the learned latent dimensions $z = [z_1, z_2]$, and examine how these latent dimensions correspond to the underlying task. Recall that the training data consists of state-action pairs which translate the robot's end-effector along (and between) circles of different radii. Ideally, the learned latent dimensions correspond to these axes, e.g., z_1 controls tangential motion while z_2 controls orthogonal motion. Interestingly, we found that this intuitive mapping is *only* captured by

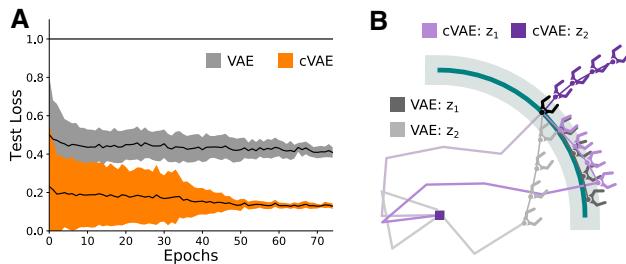


Fig. 9 Results for the *Circle* task. (A) mean-squared error between desired and reconstructed actions normalized by the PCA test loss. (B) 2-DoF latent action space $z = [z_1, z_2]$ for VAE and cVAE models. The current end-effector position is shown in black, and the colored grippers depict how changing z_1 or z_2 affects the robot’s state. Under the cVAE model, these latent dimensions move the end-effector tangent or orthogonal to the circle

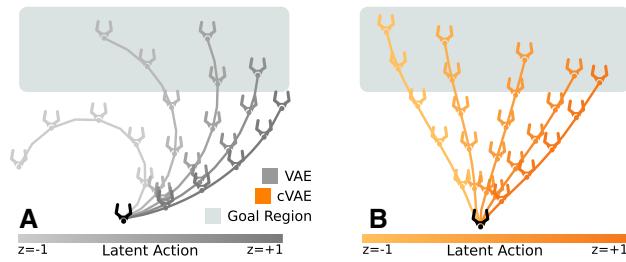


Fig. 10 Results for the *Reach* task. In both plots, we show the end-effector trajectory when applying constant inputs $z \in [-1, +1]$. The lightest color corresponds to $z = -1$ and the darkest color is $z = +1$. The goal region is highlighted, and the initial end-effector position is black. (a) trajectories with the VAE model. (b) trajectories with the cVAE model. The latent action z controls which part of the goal region the trajectory moves towards

the state conditioned models. The average angle between the directions that the end-effector moves for z_1 and z_2 is $27 \pm 20^\circ$ and $34 \pm 15^\circ$ for AE and VAE models, but this angle increases to $72 \pm 9^\circ$ and $74 \pm 12^\circ$ for the cAE and cVAE (ideally 90°). The state conditioned models better *disentangle* their low-dimensional embeddings, supporting our hypotheses and demonstrating how these models produce user-friendly latent spaces.

Reach task In the final task, a one-arm robot trains on trajectories that move towards a goal region (see Fig. 10). The robot learns a 1-DoF latent space, where z controls the direction that the trajectory moves (i.e., to the left or right of the goal region). We focus on *controllability*: can robots utilize latent actions to reach their desired goal? In order to test controllability, we sample 100 goals randomly from the goal region, and compare robots that attempt to reach these goals with either VAE or cVAE latent spaces. The cVAE robot more accurately reaches its goal: the L_2 distance between the goal and the robot’s final end-effector position is 0.57 ± 0.38 under VAE and 0.48 ± 0.5 with cVAE. Importantly, using conditioning also improves the movement *quality*. The aver-

age start-to-goal trajectory is 5.1 ± 2.8 units when using the VAE, and this length drops to 3.1 ± 0.5 with the cVAE model.

Summary The results of our *Sine*, *Rotate*, *Circle*, and *Reach* tasks support hypotheses **H1** and **H2**. Latent action models that are conditioned on the context more *accurately* reconstruct high-DoF actions from low-DoF embeddings (**H1**). Moreover, conditioned autoencoders and conditioned variational autoencoders learn latent action spaces which capture our desired properties: *controllability*, *consistency*, and *scalability* (**H2**).

8.2 Do latent actions with shared autonomy help users reach and change goals?

Now that we have tested our method for learning latent actions, the next step is to combine these latent actions with shared autonomy (see Sect. 7). Here we explore how this approach works with a *spectrum of different simulated users*. We simulate human teleoperators with various levels of expertise and adaptability, and measure whether these users can interact with our algorithm to *reach* and *change* high-level goals.

Incorporating shared autonomy In the previous simulations we used latent actions by themselves to control the robot. Now we compare this approach with and without shared autonomy:

- Latent actions with no assistance (**LA**)
- Latent actions with shared autonomy (**LA+SA**)
- Latent actions trained to maximize entropy with shared autonomy (**LA+SA+Entropy**)

For both LA and LA+SA we learn the latent space with a conditioned autoencoder (i.e., cAE in the previous section). However, here the context includes both *state and belief*. In other words, $c = (s, b)$. We also test LA+SA+Entropy, where the model uses Equation (16) to reward entropy in the learned latent space.

Environments We implement these models on both a *simulated* and a *real* robot. The simulated robot is a 5-DoF planar arm, and the real robot is a 7-DoF Franka Emika. For both robots, the state s captures the current joint position, and the action a is a change in joint position, so that: $s^{t+1} = s^t + a^t \cdot dt$.

Task We consider a manipulation task where there are two coffee cups in front of a robot arm (see Fig. 12). The human may want to reach and grasp either cup (i.e., these cups are the potential goals). We embed the robot’s high-DoF actions into a 1-DoF input space: the simulated users had to convey

both their goal and preference *only by pressing left and right* on the joystick.

Simulated humans The users attempting to complete this task are *approximately optimal*, and make decisions that guide the robot accordingly to their goal g^* . Remember that x is the position of the robot's end-effector and Ψ is the forward kinematics. The humans have reward function $R = -\|g^* - x\|^2$, and choose latent actions z to move the robot towards g^* :

$$p(z) \propto \exp \left\{ -\beta(t) \cdot \|g^* - \Psi(s + \phi(z, c) \cdot dt)\|^2 \right\} \quad (24)$$

Here $\beta \geq 0$ is a temperature constant that affects the user's *rationality*. When $\beta \rightarrow 0$, the human selects increasingly random z , and when $\beta \rightarrow \infty$, the human always chooses the z that moves towards g^* . We simulate different types of users by varying $\beta(t)$.

Users with fixed expertise We first simulate humans that have *fixed* levels of expertise. Here expertise is captured by β from Eq. (24): users with high β are proficient, and rarely make mistakes with noisy inputs. We anticipate that all algorithms will perform similarly when humans are always perfect or completely random—but we are particularly interested in the spectrum of users *between* these extremes, who frequently *mis-control* the robot.

Our results relating β to performance are shown in Fig. 11. In accordance with our convergence result from Sect. 7, we find that introducing shared autonomy helps humans reach their desired grasp more quickly, and with less final state error. The performance difference between LA and LA+SA decreases as the human's expertise increases—looking specifically at the real robot simulations, LA takes 45% more time to complete the task than LA+SA at $\beta = 75$, but only 30% more time when $\beta = 1000$. We conclude that shared autonomy improves performance across all levels of expertise, both when latent actions are trained with and without entropy.

Users that change their mind One downside of shared autonomy is over-assistance: the robot may become *constrained* at likely (but incorrect) goals. To examine this adverse scenario we simulate humans that *change* which coffee cup they want to grasp after N timesteps. These simulated users intentionally move towards the *wrong* cup while $t \leq N$, and then try to reach the correct cup for the rest of the task. We model humans as near-optimal immediately after changing their mind about the goal.

We visualize our results in Fig. 12. When the latent action space is trained only to minimize reconstruction loss (LA + SA), users cannot escape the shared autonomy constraint around the wrong goal as N increases. Intuitively, this occurs

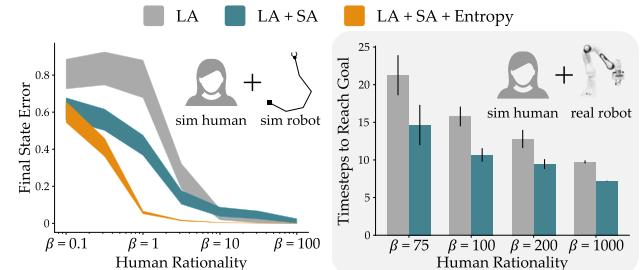


Fig. 11 Simulated humans for different levels of rationality. As $\beta \rightarrow \infty$, the human's choices approach optimal inputs. *Final State Error* (in all plots) is normalized by the distance between goals. Introducing shared autonomy (SA) improves the convergence of latent actions (LA), particularly when the human teleoperator is noisy and imperfect

because the latent space controls the intended goal when the belief b is roughly uniform, and then switches to controlling the preferred trajectory once the robot is confident. So if users change their goal after first convincing the robot, the latent space no longer contains actions that move towards this correct goal! We find that our proposed entropy loss function addresses this shortcoming: LA + SA + Entropy users are able to input actions z that alter the robot's goal. Our results suggest that encouraging entropy at training time improves the robustness of the latent space.

Users that learn within the task We not only expect real users to change their mind when collaborating with the robot, but we also anticipate that these teleoperators will *learn and improve* as they gain experience during the task. For instance, the user might learn that holding left on the joystick causes the robot to grasp the cup from the side, while holding right guides the robot towards a top grasp. To simulate this in-task learning, we set $\beta(t) = m \cdot t$, where the slope m determines how quickly the user learns. All users start with random actions ($\beta = 0$), and either learn *quickly* (high m) or *slowly* (low m). We point out that slow learners may effectively “change their mind” multiple times, since they are unsure of how to control the robot.

Our findings are plotted in Fig. 13. We see that—for both fast and slow learners—LA+SA+Entropy improves in-task performance. We attribute this improvement to the inherent *versatility* of latent spaces that maximize entropy: as humans gain expertise, they can use these latent actions to quickly undo their mistakes and correct the robot's behavior.

Summary Overall, these simulations show that users are not able to precisely reach and maintain goals when controlling robots with *only* latent actions. Including shared autonomy improves convergence, while training latent actions to maximize entropy ensures that this convergence does not become a burden. When using our proposed combination of shared autonomy and learned latent actions, naïve and experienced

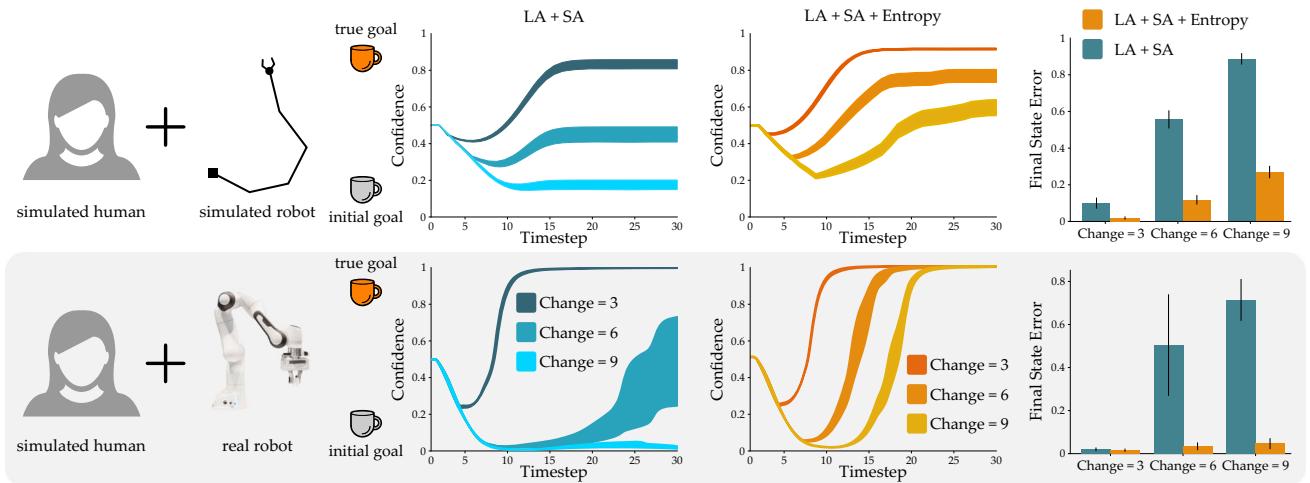


Fig. 12 Simulated humans that change their intended goal part-way through the task. *Change* is the timestep where this change occurs, and *Confidence* refers to the robot's belief in the human's true goal. Because of the constraints imposed by shared autonomy, users need latent actions

that can overcome misguided assistance and move towards a less likely (but correct) goal. Encouraging entropy in the learned latent space (*LA + SA + Entropy*) enables users to switch goals

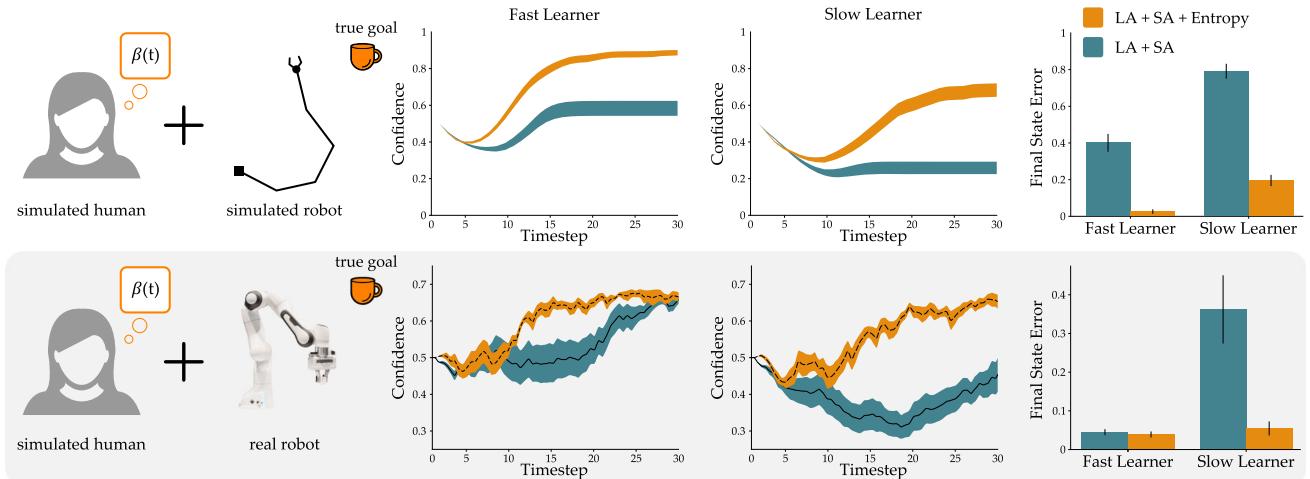


Fig. 13 Simulated humans that learn how to teleoperate the robot. The human's rationality $\beta(t)$ is linear in time, and either increases with a high slope (*Fast Learner*) or low slope (*Slow Learner*). As the human learns, they get better at choosing inputs that best guide the robot towards their

true goal. We find that latent actions learned with the entropy reward (*LA+SA+Entropy*) are more versatile, so that the human can quickly undo mistakes made while learning

users are able to reach their preferred goal and change their mind.

8.3 Can we efficiently align latent actions with joystick inputs?

By combining shared autonomy with latent actions, we have a way for humans to precisely control high-DoF robots. But currently the mapping between joystick inputs and latent actions is arbitrary—and this makes it challenging for users to know how to leverage latent actions. Here we test our

proposed alignment method from Sect. 6. We explore how efficiently we can learn a simulated human's preferred alignment by using semi-supervised learning and underlying priors.

Setup Simulated users control the FrankaEmika arm from the previous section. The latent action space and decoder $\phi(z, c)$ were trained using conditional autoencoders. This decoder ϕ maps from a 2-DoF latent action z to a 7-DoF robot arm motion. We now leave the decoder ϕ fixed, and focus on learning the alignment model f . This alignment

model $z = f(u, c)$, takes 2-DoF inputs from the simulated human and converts them to latent actions for the robot to execute.

Tasks We considered three tasks of increasing complexity. In each task the simulated user interacted with a 2-DoF joystick.

1. *Plane*: The robot moves its end-effector in the x - y plane. The human prefers for one joystick DoF to move the robot along the x -axis, and the other should move the robot along the y axis.
2. *Pour*: The robot moves and rotates its end-effector along z axis. The human expects one dimension of the joystick to move the robot up and down, and the other to control a pouring motion.
3. *Reach & Pour*: The robot reaches for a bottle, carries it to a bowl, and then pours the contents. The human’s preference is divided into two parts: when reaching for the bottle in the x - y plane, the human’s preference matches *Plane*, and when pouring, the human’s preference is the same as in *Pour*.

Learning the alignment Before each of these tasks we provided the simulated human with 10 different motions to label. The human indicated which joystick input u they would expect to use to command the demonstrated motion (s, s^*) . In addition to these 10 labeled datapoints, the robot also collected 1000 unlabeled motions for self-supervised learning. Given this data, we compared our approach to different baselines. To better understand which priors are useful, we also included an ablation study where the robot learned with only one prior at a time. Overall, the conditions were:

- **No Align**: baseline where $z = u$
- **Manual Align**: the affine transformation that best matches the labeled data
- **No Priors**: trained with the supervised loss \mathcal{L}_{sup} from Equation (19)
- **Proportional**: trained with $\mathcal{L}_{sup} + \mathcal{L}_{prop}$, the proportional prior from Equation (20)
- **Reversible**: trained with $\mathcal{L}_{sup} + \mathcal{L}_{reverse}$, the reversible prior from Equation (21)
- **Consistent**: trained with $\mathcal{L}_{sup} + \mathcal{L}_{con}$, the consistency prior from Equation (22)
- **All Priors**: trained with all of the proposed priors
- **Ideal Align**: supervised loss where the simulated human answers 1000 queries (instead of 10).

Besides the type of alignment model, we also varied how imperfectly the simulated human answered our queries. We set the coefficient of variance as 0, 0.1, and 0.5 for the simulated human when they answered queries.

Dependent measures To determine the quality of each alignment model, we measured the error between where the human intended to go and where the robot actually went. Specifically, we measured the relative end-effector position and orientation. Let $x^* = \Psi(s^*)$ be the intended end-effector pose, let x^t be the start pose, and let x^{t+1} be where the robot actually ended up: we computed $\|x^* - x^{t-1}\|^2 / \|x^* - x^t\|^2$. For each experiment setting, we reported mean and standard deviation of this metric over 10 total runs.

Hypotheses We expected three things during our alignment simulations:

- H1.** With abundant labeled data, the alignment model will learn the human’s preferences.
- H2.** Compared to the fully-supervised baseline, our semi-supervised alignment models that leverage intuitive priors will achieve similar performance with far less human data.
- H3.** Semi-supervised training with proportional, reversible, and consistent priors will outperform models trained with only one of these priors.

Results We highlight results for the *Plane* and *Reach + Pour* tasks in Fig. 14. For models that do not leverage data or personalization—i.e., No Align and Manual Align—the error is significantly higher than learning alternatives. With abundant data and noise-free human annotations, Ideal Align provided the gold-standard performance. The success of Ideal Align indicates that our parametrization of the alignment model f is capable of capturing the human’s preferences.

In practice, the amount of human feedback will always be limited. We found that our proposed priors were critical when looking at models that only had access to 10 queries during training. The three semi-supervised models with just one intuitive prior (Proportional, Reversible, and Consistent) performed twice as well as No Priors, the supervised baseline. Putting all of these priors together resulted in even better performance: across different user noise levels, All Priors consistently demonstrated the lowest mean error and standard deviation. This was particularly noticeable when the human oracle is noisy, suggesting that the three priors are indeed complementary, and including each of them together brings a performance boost!

Comparing the easier *Plane* task to the more complex *Reach & Pour* task in Figure 14, we also saw that using priors became increasingly important as the task got harder. This suggests that—in complex scenarios—simply relying on a few labeled examples during supervised learning may lead to severe overfitting. Our intuitive priors for semi-supervised learning effectively mitigate this problem.

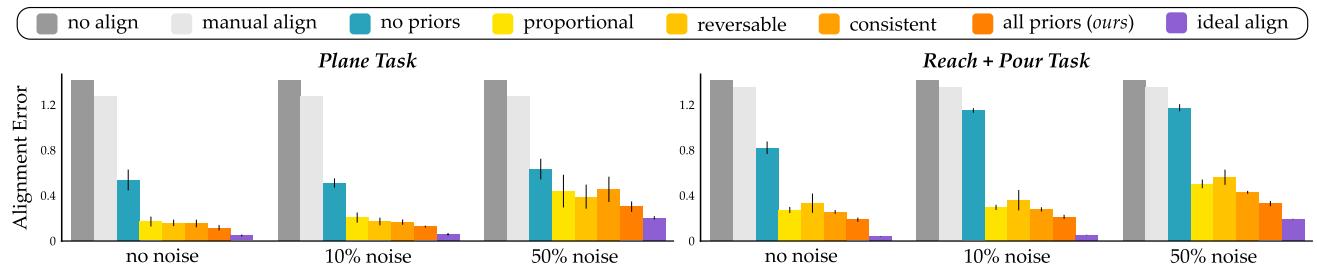


Fig. 14 Learning the joystick alignment from simulated and imperfect human feedback. We tested three different tasks with increasing complexity, and here we display the results of the easiest (*Plane*) and the hardest (*Reach + Pour*). *Alignment Error* refers to the difference between where the simulated human expected the robot to move, and

where the robot actually moved. To explore the robustness of our method we varied how noisy the human was when providing their preferences. Across different tasks and levels of human noise, All Priors (our semi-supervised approach) consistently outperformed the other methods, and almost matched an ideal alignment learned from abundant data

Summary Viewed together, the results of our simulations strongly support the hypotheses **H1**, **H2**, and **H3**. Our proposed alignment model successfully learned the mapping between the joystick inputs and latent actions (**H1**). In settings with limited labels, our proposed alignment model with intuitive control priors reached results that almost match supervised training with abundant data (**H2**). Finally, in ablation studies, we showed how combining all three proposed priors leads to superior performance and greater training stability than training with a single prior (**H3**).

9 User studies with non-disabled participants

To evaluate whether actual humans can use learned latent actions to teleoperate robots and perform assistive eating tasks, we conducted four user studies. The participants in these studies are all *non-disabled* adults (we apply our approach with disabled adults in Sect. 10). Importantly, we designed these user studies to mimic assistive teleoperation settings. In each study the human user interacts with a 2-DoF joystick, and uses this joystick to control a 7-DoF robot arm.

The order of the studies roughly parallels Sects. 6, 7, and 6. We start by comparing latent actions to an existing dataset for assistive eating tasks, and then compare latent actions to the end-effector teleoperation method commonly used on assistive robot arms. Next, we introduce shared autonomy, and perform an ablation study to understand how shared autonomy and latent actions contribute to high-level reaching and precise manipulation tasks. Finally, we learn the alignment between joystick inputs and latent actions by considering intuitive priors, and evaluate how this alignment improves human-robot co-adaptation.

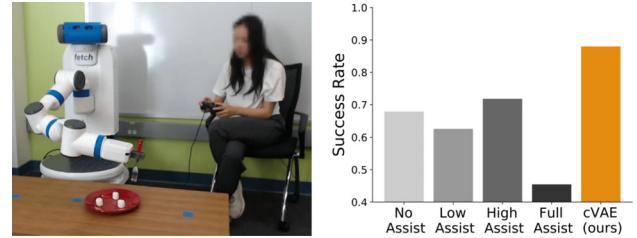


Fig. 15 Experimental setup for our user study from Sect. 9.1. (Left) in this eating task, the participant uses a two-DoF joystick to guide the robot to reach their desired marshmallow. (Right) we compare our latent action approach to shared autonomy baselines from the HARMONIC dataset

9.1 Comparing latent actions to the HARMONIC baselines

In our first user study users teleoperate an assistive robot arm using *only learned latent actions*. We baseline their performance against current state-of-the-art approaches for controlling high-DoF assistive robot arms. Specifically, we implement the same assistive eating task as in the HARMONIC dataset (Newman et al. 2018). This task is shown in Fig. 15: users guide the robot arm to stab a marshmallow of their choice. There are three different marshmallows on the plate—i.e., three possible high-level goals—and only the human knows which of these discrete goals they want to reach. The HARMONIC dataset reports the performance of 24 people who completed this task with different levels of shared autonomy. Within the HARMONIC baseline, however, the mapping from low-DoF user inputs to high-DoF robot actions is *predefined*, with separate *modes* to control the end-effector's *x-y* position, *z-yaw* position, and *roll-pitch* orientation. We compare the learned latent mapping from Sect. 6 to this set of baselines.

Independent variables We manipulated the robot's teleoperation strategy with five levels: the four conditions from the HARMONIC dataset plus our proposed learned

latent actions. For the first four conditions, the robot uses optimization-based shared autonomy (Javdani et al. 2018) to select assistive actions. The robot either provides no assistance (**No Assist**) or linearly interpolates between the human’s input and the assistive action (**Low Assist**, **High Assist**, and **Full Assist**). **High Assist** was the most effective strategy from this group: when interpolating, here the assistive action is given twice the weight of the human’s commanded action. Within our learned latent actions condition, we applied a conditional variational autoencoder (**cVAE**) to learn the decoder $\phi(z, c)$ from the demonstrations in the HARMONIC dataset. For now we treat the joystick inputs as the latent actions, so that $z = u$.

Dependent measures Before each trial users indicated which marshmallow they want to reach. We measured the fraction of trials in which the robot picks up the correct marshmallow (*Success Rate*), the amount of time needed to complete the task (*Completion Time*), the total magnitude of the human’s input (*Joystick Input*), and the distance traveled by the robot’s end-effector (*Trajectory Length*).

Hypothesis We hypothesized that:

- H1.** Compared to the baselines, latent actions will improve task success while reducing the completion time, joystick inputs, and trajectory length.

Experimental setup Participants interacted with a joystick while watching the robotic arm (see Fig. 15). The robot held a fork; during the task, users teleoperated the robot to position this fork directly above their desired marshmallow. We selected the robot’s start state, goal locations, and movement speed to be consistent with the HARMONIC dataset.

Participants and procedure Our participant pool consisted of ten Stanford University affiliates who provided informed consent (3 female, average participant age 23.9 ± 2.8 years). Following the same protocol as the HARMONIC dataset, each participant was given up to five minutes to familiarize themselves with the task and joystick, and then completed five recorded trials using our cVAE approach. At the start of each trial the participant indicates which marshmallow they want the robot to reach; the trial ends after the user indicates that the fork is above their intended marshmallow. We point out that participants only completed the task with the cVAE condition; other teleoperation strategies are benchmarked in Newman et al. (2018).

Results We display example robot trajectories in Fig. 16 and report our dependent measures in Figs. 15 and 17. Inspecting these example trajectories, we observe that the cVAE model

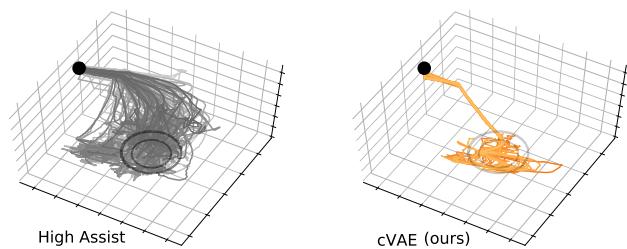


Fig. 16 End-effector trajectories from **High Assist** and **cVAE** conditions. The robot starts at the black dot, and moves to position itself over the plate

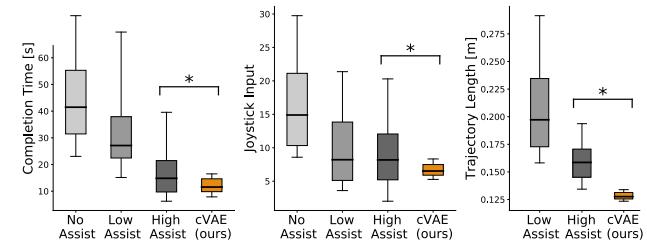


Fig. 17 Comparing our objective results to the HARMONIC baseline. We found that **cVAE** led to faster task completion with less user input and end-effector motion. The **Full Assist** condition performed worse than **High Assist** across the board (omitted for clarity). Error bars show the 10 and 90 percentiles, and * denotes statistical significance ($p < .05$)

learned latent actions that constrain the robot’s end-effector into a region above the plate. Users controlling the robot with cVAE reached their desired morsel in 44 of the 50 total trials, yielding a higher *Success Rate* than the assistance baselines. To better compare cVAE to the High Assist condition, we performed independent t-tests. Participants with the cVAE model had significantly lower *Completion Time* ($t(158) = 2.95, p < .05$), *Joystick Input* ($t(158) = 2.49, p < .05$), and *Trajectory Length* ($t(158) = 9.39, p < .001$), supporting hypothesis **H1**.

Summary We baselined our learned mapping from low-DoF to high-DoF actions against state-of-the-art shared autonomy approaches with predefined mappings. Users teleoperating an assistive robot with learned latent actions reached their high-level goals more accurately, while requiring less time, effort, and movement.

9.2 Comparing latent actions to end-effector teleoperation

Real-world assistive eating tasks involve more than just reaching for discrete goals (i.e., stabbing a food morsel). Often objects lie in continuous regions (i.e., anywhere on a shelf), and users must make continuous decisions (i.e., how much water to pour). In our second user study we therefore apply learned latent actions to *continuous* tasks.

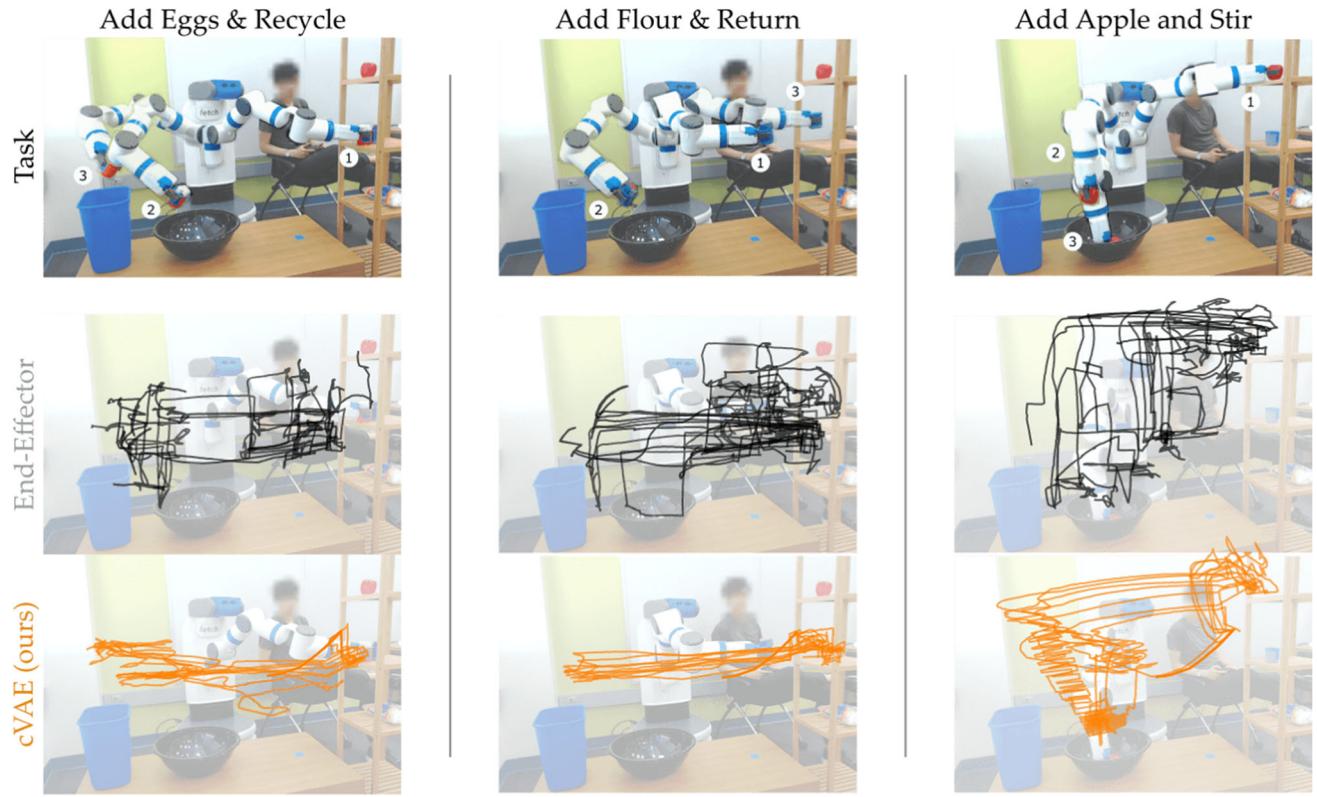


Fig. 18 Experimental setup for our user study in Sect. 9.2. (Top row) the participant is teleoperating an assistive robot to make their “apple pie” recipe. This recipe is broken down into three sub-tasks. On left the robot picks up eggs, pours them into the bowl, then drops the container into the recycling. In middle the robot picks up flour, pours it into the bowl, then returns the container to the shelf. On right the robot grasps

an apple, places it in the bowl, then stirs the mixture. (Middle row) example robot trajectories when the person directly controls the robot’s **End-Effector**. (Bottom row) example trajectories when using **cVAE** to learn latent actions. Comparing the example trajectories, we observe that **cVAE** resulted in robot motions that more smoothly and directly accomplished the task

Consider cooking the “apple pie” in Fig. 18. Assembling this recipe requires picking up ingredients from the shelf, pouring them into a bowl, recycling empty containers—or returning half-filled containers to the shelf—and then stirring the mixture. Shared autonomy approaches like (Dragan and Srinivasa 2013; Newman et al. 2018; Javdani et al. 2018) are not suitable within this setting because: (i) the goals lie in continuous regions and (ii) the user needs to control both the *high-level goal* that the robot reaches for and the *trajectory* the robot follows to reach that goal (e.g., keeping a cup upright until pouring). Hence, we compare our latent action method against *end-effector teleoperation*. End-effector teleoperation is commonly used by assistive robots (Herlant et al. 2016), where the human presses a button to switch modes and control different aspects of the end-effector’s motion. For example, in one mode the 2-DoF joystick moves the robot’s end-effector in the x - y plane. See videos of this user study here: <https://youtu.be/wjnhrzugBj4>.

Independent variables We tested two teleoperation strategies: **End-Effector** and **cVAE**. Under End-Effector the user

inputs apply a 6-DoF twist to the robot’s end-effector, controlling its linear and angular velocity. Participants interact with two 2-DoF joysticks, and are given a button to toggle between linear and angular motion (Herlant et al. 2016; Newman et al. 2018; Javdani et al. 2018). By contrast, in cVAE the participants only interact with one 2-DoF joystick, i.e., the latent action is $z = [z_1, z_2] \in \mathbb{R}^2$. We emphasize that this cVAE latent action model has the same structure as the one used in our previous user study (Sect. 9.1), and does not yet include either shared autonomy or alignment models. We trained cVAE using state-action pairs from kinesthetic demonstrations, where we guided the robot along related sub-tasks such as reaching for the shelf, pouring objects into the bowl, and stirring. Overall, the cVAE was trained with less than 7 minutes of demonstration data.

Dependent measures—objective We measured the total amount of time it took for participants to complete the entire

cooking task (*Completion Time*), as well as the magnitude of their inputs (*Joystick Input*).

Dependent measures—subjective We administered a 7-point Likert scale survey after each condition. Questions were separated into six scales, such as ease of performing the task (*Ease*) and consistency of the controller (*Consistent*). Once users had completed the task with both strategies, we asked comparative questions about which they preferred (*Prefer*), which was *Easier*, and which was more *Natural*.

Hypotheses We had the following hypotheses:

- H1.** Users controlling the robot arm with low-DoF latent actions will complete the cooking task more quickly and with less overall effort.
- H2.** Participants will perceive the robot as easier to work with in the cVAE condition, and will prefer the cVAE over End-Effector teleoperation.

Experimental setup We developed a cooking task where the person is making a simplified “apple pie.” As shown in Fig. 18, the assistive robot must sequentially pour eggs, flour, and an apple into the bowl, dispose of their containers, and stir the mixture. The user sat next to the robot and controlled its behavior with a handheld joystick. During the experiment we introduced variance by intermittently changing the location of the shelf, bowl, and recycling bin.

Participants and procedure Eleven members of the Stanford University community (4 female, age range 27.4 ± 11.8 years) provided informed consent to participate in this study. Similar to the other user studies described in this paper, we used a within-subjects design, and counterbalanced the order of our two conditions. Four of our subjects had prior experience interacting with the robot used in our experiment.

Before starting the study, participants were shown a video of the cooking task. Participants then separately completed the three parts of the task as visualized in Fig. 18; we reset the robot to its home position between each of these sub-tasks. After the user completed these sub-tasks, we re-arranged the placement of the recycling and bowl, and users performed the entire cooking task without breaks. Participants were told about the joystick interface for each condition, and could refer to a sheet that labelled the joystick inputs.

Results—objective Our objective results are summarized in Fig. 19. When using cVAE to complete the entire recipe, participants finished the task in less time ($t(10) = -6.9, p < .001$), and used the joystick less frequently ($t(10) = -5.1, p < .001$) as compared to direct End-Effector teleoperation.

Results—subjective We display the results of our 7-point Likert scale surveys in Fig. 20. Before reporting these results,

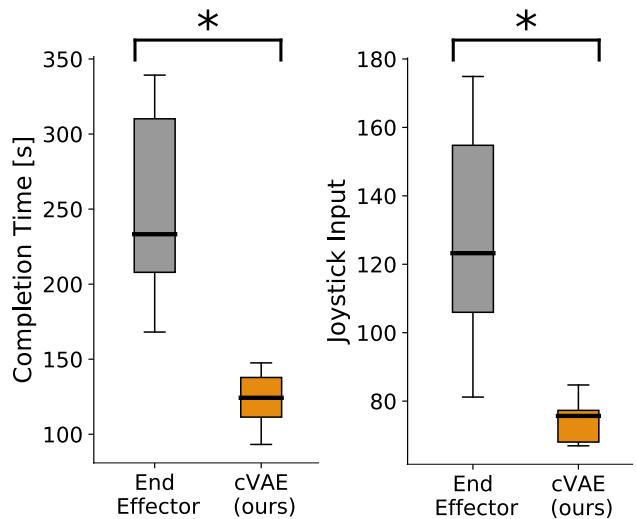


Fig. 19 Objective results from assembling an “apple pie.” These results were collected across the entire cooking task (combining each sub-task from Fig. 18). We compare using just latent actions to direct end-effector teleoperation

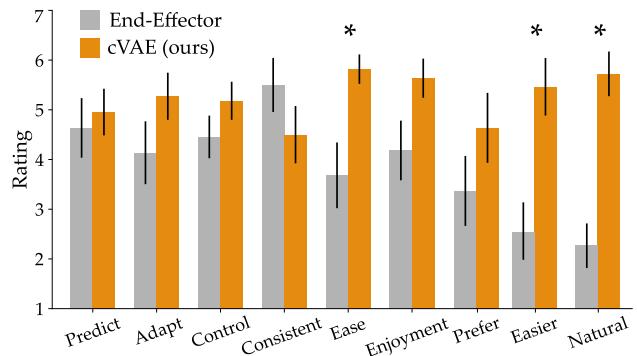


Fig. 20 Subjective results from assembling an “apple pie.” Higher ratings indicate participant agreement. Participants thought our approach required less effort (*Ease*), made it easier to complete the task (*Easier*), and produced more natural robot motion (*Natural*) as compared to End-Effector control

we first confirmed the reliability of our scales. We then leveraged paired t-tests to compare user ratings for End-Effector and cVAE conditions. We found that participants perceived cVAE as requiring less user effort ($t(10) = 2.7, p < .05$) than End-Effector. Participants also indicated that it was easier to complete the task with cVAE ($t(10) = 2.5, p < .05$), and that cVAE caused the robot to move more naturally ($t(10) = 3.8, p < .01$). The other scales were not significantly different.

Summary We focused on a cooking task with continuous high-level goals, and compared latent actions to end-effector teleoperation. When controlling the robot with latent actions, users completed the cooking task more quickly and with less effort (**H1**). Participants believed that the cVAE approach led to more natural robot motion, and indicated that it was easier

to perform the task with latent actions. However, participants did not indicate a clear preference for either strategy (**H2**). We will explore ways to improve user satisfaction in our next user study, where we combine latent actions with shared autonomy to assist the human.

9.3 Combining learned latent actions with shared autonomy

In our first two user studies we tested latent actions *by themselves*, without any robotic assistance. Our results indicate that latent actions objectively outperform both the HARMONIC baselines and end-effector control; however, the participant's subjective responses are mixed. Plus, so far we have only dealt with high-level reaching goals—but assistive eating also involves fine-grained manipulation, where the user must cut, stab, and scoop their food. Here we tackle both issues by performing a user study with *eating tasks* (see Fig. 21). Participants must control the robot towards their goal plate, and then carefully adjusted the robot's motion to cut, stab, and scoop different foods. We explore how users leverage learned latent actions *with shared autonomy* to complete this task. Specifically, we conduct an ablation study across latent actions and shared autonomy, and determine whether latent actions alone, shared autonomy alone, or their combination best results in high-level reaching and precise manipulation. See videos of this user study here: <https://youtu.be/7BouKojzVyk>.

Experimental setup Each participant attempted to complete two dishes: an *Entree task* and a *Dessert task*. In *Entree*, users had to perform multiple precise motions at the *same* goal. Here participants (i) guided the robot towards a bowl with tofu, (ii) cut off a slice of tofu, and (iii) stabbed and scooped the slice onto their plate. In *Dessert* the participants had to convey their preferences at *multiple* goals: they (i) stabbed a marshmallow in the middle plate, (ii) scooped it through icing at the right plate, and then (iii) dipped it in rice at the left plate before (iv) setting the marshmallow on their plate. In both tasks subjects sat next to the robot, mimicking a wheelchair-mounted arm.

Independent variables We conducted a 2×2 factorial design that separately varied *Control Interface* and *Robot Assistance*.

For the control interface, we tested a state-of-the-art direct teleoperation scheme (*Retargetting*), where the user's joystick inputs map to the 6-DoF end-effector twist of the robot (Rakita et al. 2017). We compared this direct teleoperation baseline to our learned latent actions: here the robot interprets the meaning of the human's inputs based on the current context.

For robot assistance, we tested *with* and *without shared autonomy*. We implemented the shared autonomy algorithm from (Javdani et al. 2018), which assists the robot towards likely human goals.

Crossing these two separate factors, we totaled four different conditions:

- **R**: Retargeting with no shared autonomy
- **R+SA**: Retargeting with shared Autonomy
- **LA**: Latent actions with no shared autonomy
- **LA+SA**: Latent actions with shared autonomy

Note that the **LA+SA** condition is our proposed approach (Algorithm 1). However, we still omit the alignment model f , so for now the latent action z is set equal to the joystick input u .

Model training We provided kinesthetic demonstrations \mathcal{D} that guided the robot towards each plate, and then performed cutting, stabbing, and scooping motions at these goals. The robot learned the latent action space from a total of 20 minutes of kinesthetic demonstrations. Because we are combining latent actions with shared autonomy, we also recorded the belief b during these demonstrations. In the LA+SA condition, the robot used context $c = (s, b)$ to decode human inputs.

Dependent measures—objective We recorded the amount of time users took to complete each task (*Total Time*), as well as the amount of time spent without providing joystick inputs (*Idle Time*). We also computed proxy measures of the high-level goal accuracy and low-level preference precision. For goals, we measured the robot's total distance to the closest plate throughout the task (*Goal Error*). For preferences, we recorded the dot product between the robot's actual end-effector direction and the true end-effector directions needed to precisely cut, stab, and scoop (*Preference Alignment*).

Dependent measures—subjective We administered a 7-point Likert scale survey after each condition. Questions were organized along five scales: how *Easy* it was to complete the tasks, how *Helpful* the robot was, how *Precise* their motions were, how *Intuitive* the robot was to control, and whether they would use this condition again (*Prefer*).

Participants and procedure We recruited 10 subjects from the Stanford University student body to participate in our study (4 female, average age 23.5 ± 2.15 years). All subjects provided informed written consent prior to the experiment. We used a within-subjects design: each participant completed both tasks with all four conditions (the order of the conditions was counterbalanced). Before every trial, users

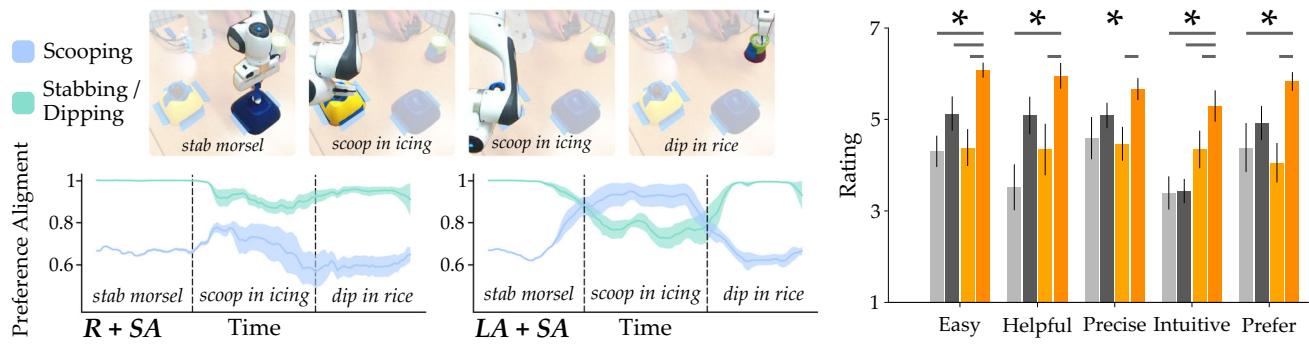


Fig. 21 Experimental setup for our user study in Sect. 9.3. (Left) The *Dessert* task consists of 3 phases: stabbing the marshmallow, scooping it in icing, and dipping it in rice. We identified the end-effector directions needed to complete these fine-grained preferences, and plotted the average dot product between the desired and actual directions (*Preference Alignment*). In the R+SA condition, users executed the entire task

in a stabbing/dipping orientation. By contrast, with LA+SA users correctly adjusted the scooping preference in the second phase of the task. (Right) We plot the results of our 7-point Likert scale surveys. Color to method mappings are consistent with Fig. 22, and * indicate statistical significance ($p < 0.05$)

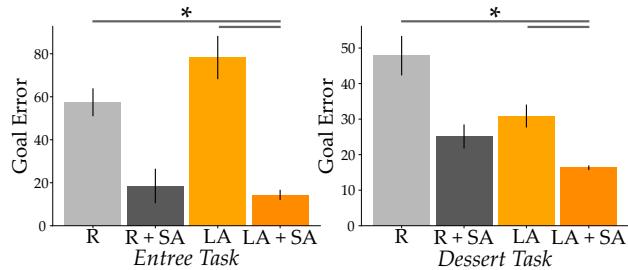


Fig. 22 Error between the end-effector and nearest goal during the eating tasks. Adding shared autonomy (SA) decreased this error across both mapping strategies (R and LA)

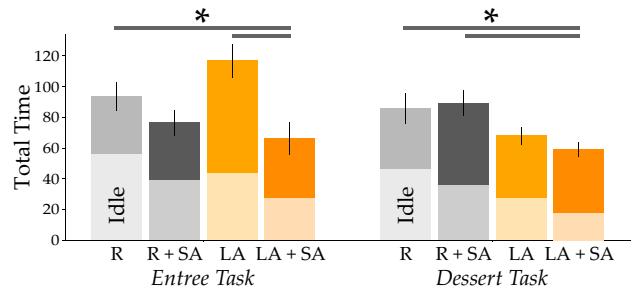


Fig. 23 Time taken to complete the eating task (solid) and time spent idle (light). Users completed both tasks most efficiently with our proposed combination of shared autonomy and latent actions (LA+SA)

practiced teleoperating the robot with the current condition for up to 5 minutes.

Hypotheses We tested three main hypotheses:

- H1.** Users controlling the robot with shared autonomy will more accurately maintain their goals.
- H2.** Latent actions will help users more precisely perform manipulation tasks.
- H3.** Participants will complete the task most efficiently with combined LA+SA.

Results—objective To explore **H1**, we analyzed the *Goal Error* for methods with and without shared autonomy (see Figure 22). Across both tasks, users interacting with shared autonomy reached their intended goals significantly more accurately ($F(1, 18) = 29.9, p < .001$). Breaking this down by condition, users incurred less error with LA+SA than with LA ($p < .001$), and—similarly—users were more accurate with R+SA than with R ($p < .05$). Building on our prior

user study results, this indicates that adding shared autonomy improves the performance of our latent action approach.

So shared autonomy helped users more accurately maintain their goals—but were participants able to complete the precise manipulation tasks at those goals? We visualize the *Preference Alignment* for *Dessert* in Figure 21, specifically comparing R+SA to LA+SA. We notice that—when using direct teleoperation—participants remained in a stabbing preference throughout the task. By contrast, users with latent actions *adjusted* between different manipulation tasks: stabbing the marshmallow, scooping it in icing, and dipping it in rice. These results support **H2**, suggesting that latent actions enable users to precisely manipulate the robot.

Now that we know the benefits of shared autonomy and latent actions individually, what happens when we focus on their combination? Inspecting Fig. 23, participants using LA+SA were able to complete both tasks more efficiently. Summing times across both tasks, and then performing pairwise comparisons between each condition, we found that LA+SA outperformed the alternatives for both *Total Time*

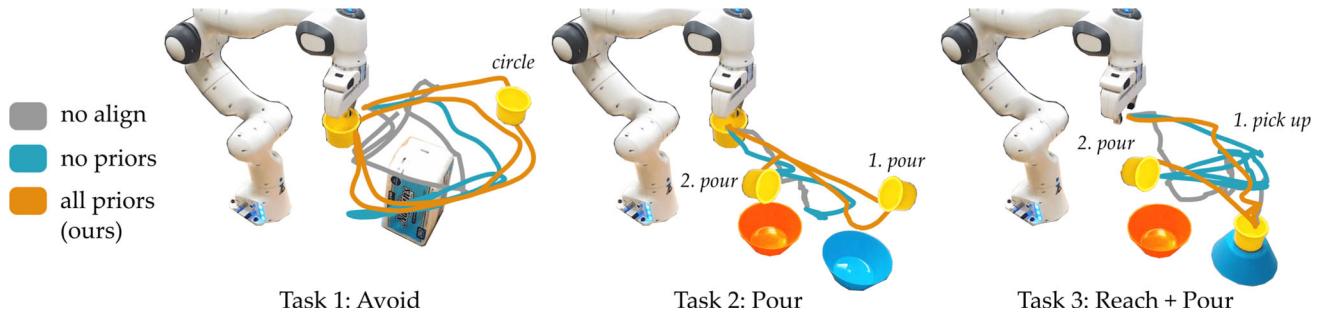


Fig. 24 Experimental setup for the user study in Section 9.4. We visualize a single user’s end-effector trajectories for the *Avoid*, *Pour* and *Reach + Pour* tasks. Participants teleoperated the 7-DoF Panda robot arm without any alignment model (No Align), with an alignment model trained only on their supervised feedback (No Priors), and with our pro-

posed method, where the robot generalizes the human’s feedback using intuitive priors (All Priors). For both No Align and No Priors baselines, we can see moments where the human gets confused, counteracts themselves, or fails to complete the task

($p < .05$) and *Idle Time* ($p < .05$). Overall, we found that LA+SA users completed the task the fastest and with the least error.

Results—subjective We find further support for H3 in the user’s feedback. The results of t-tests comparing LA+SA to the other conditions are reported in Fig. 21 (where an * denotes $p < .05$). Responses suggest that users were most “comfortable” when performing precise manipulation with LA+SA. We emphasize the improvement in these subjective results as compared to Fig. 20, where users ranked latent actions similarly to direct end-effector control. We conclude that incorporating shared autonomy improves the human’s experience when leveraging latent actions.

Summary We conducted two eating tasks where participants needed to (i) reach for high-level goals and (ii) precisely manipulate food items at those goals. We found that both shared autonomy and latent actions improved performance: including shared autonomy decreased end-effector error, while using latent actions helped users quickly transition between different fine-grained manipulations. These results compliment Sect. 9.1 (where we compare latent actions to shared autonomy) and Sect. 9.2 (where we compare latent actions to end-effector control). But now we also have that the *combination* of shared autonomy and latent actions outperforms either of these approaches alone.

9.4 Learning the alignment between joystick inputs and latent actions

Now that we have shown the benefits of latent actions—and how latent actions can incorporate shared autonomy—we finally turn our attention to aligning the human’s joystick inputs with the learned latent space. This final user study builds on Sect. 6. Prior to the study, we learn a latent action space that maps from 2-DoF inputs to 7-DoF robot actions.

During the study, we ask participants to label a few sample robot motions with their preferred joystick input. The robot uses intuitive priors to generalize from these labels and learn a personalized mapping from joystick inputs u to latent actions z . In the previous studies we have simply set $z = u$, and we leverage this as one of the baselines in this user study. We also evaluate the effects of our intuitive priors, and compare learning the alignment model *with* and *without* these priors. See videos of this user study here: https://youtu.be/rKHka0_48discretionary-Q.

Tasks Similar to our simulation experiments from Sect. 8.3, we considered three different tasks. These tasks are visualized in Fig. 24.

1. *Avoid*: The robot arm moves its end-effector in a horizontal plane. Users are asked to guide the robot around an obstacle without colliding with it.
2. *Pour*: The robot arm is holding a cup, and users want to pour this cup into two bowls. Users are asked to first pour into the farther bowl, before moving the cup back to the start and pouring into the closer bowl.
3. *Reach & Pour*: Users start by guiding the robot towards a cup and then pick it up. Once the users reach and grasp the cup, they are asked to take the cup to a target bowl, and finally pour into it.

Independent variables For each of the tasks described above, we compared three different alignment models. **No Align** corresponds to what we have done in the previous user studies: setting $z = u$, so that all users must adapt to the same latent action alignment. We compare this to two *personalized* approaches. First is **No Priors**, where we train the alignment model f just using the supervised loss (i.e., the robot only learns from the human’s labeled data). We contrast this to **All Priors**, where the robot leverages semi-supervised learning

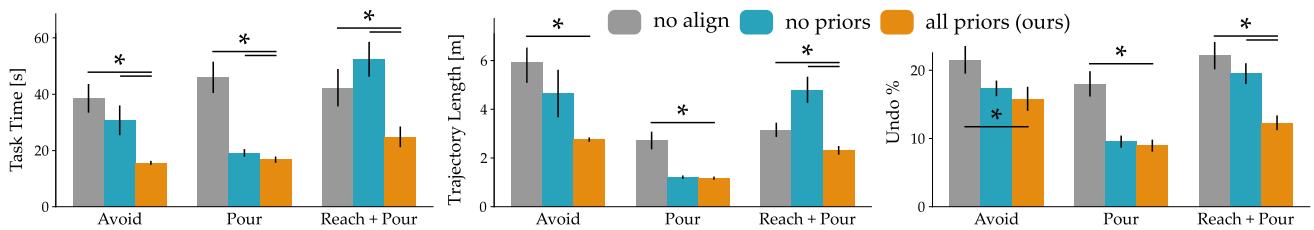


Fig. 25 Objective results from our alignment user study. (Left) Average time taken to complete each task. (Middle) Average trajectory length as measured in end-effector space. (Right) Percentage of the time people spend undoing their actions. Error bars show the standard deviation

across the 10 participants, and colors match Fig. 24. Asterisks denote statistically significant pairwise comparisons between the two marked strategies ($p < .05$)

(i.e., the robot also considers the proportional, reversible, and consistent priors we anticipate that the user will expect). Each condition learns from the same human feedback; we emphasize that All Priors uses the same number of human labels as No Priors.

Dependent measures To evaluate the effectiveness of these different alignment strategies, we recorded *Task Time* and *Trajectory Length*. We also calculated the percentage of the time that users *Undo* their actions by significantly changing the joystick direction—undoing suggests that the alignment is not quite right, and the human is still adapting to the robot’s control strategy. Besides these objective measures, we also collected subjective feedback from the participants through 7-point Likert scale surveys.

Participants and procedure We recruited 10 volunteers that provided informed written consent (3 female, ages 23.7 ± 1.5). Participants used a 2-axis joystick to teleoperate the 7-DoF robot arm, and completed three manipulation tasks inspired by assistive settings. At the start of each task, we showed the user a set of robot movements and ask them to provide their preferred input on the joystick—i.e., “if you wanted the robot to perform the movement you just saw, what joystick input would you provide?” Users answered 7 queries for task *Avoid*, 10 queries for task *Pour* and 30 queries for task *Reach & Pour*. After the queries finished, the users started performing tasks sequentially using each of the alignment strategies. The order of alignment strategies was counterbalanced.

Hypothesis We hypothesize that:

H1. An alignment model learned from user-specific feedback and generalized through intuitive priors will make it easier for humans to control the robot and perform assistive manipulation tasks.

Results The objective results of our user study are summarized in Fig. 25. Across tasks and metrics, our model with All Priors outperforms the two baselines. In addition, our

model not only has the best average performance, but it also demonstrates the least variance. Similar to our simulation results from Sect. 8.3, when the task is difficult (i.e., *Reach & Pour*), the performance of No Priors drops significantly compared to simpler tasks, reinforcing the *importance of priors* when in learning complex alignment models (Fig. 26).

We also illustrate our survey responses in Fig. 27. Across the board, we found that users exhibited a clear preference for our proposed method. Specifically, they perceived All Priors as resulting in better alignment, more natural, accurate, and effortless control, and would elect to use it again. These subjective results highlight the importance of *personalization* when controlling high-DoF systems—we contrast these results to Fig. 20, where participants perceived the unaligned controller as somewhat unintuitive.

To better visualize the user experiences, we also display example robot end-effector trajectories from one of the participants in Fig. 24. Here we observe that the trajectories of our model (in orange) are smooth and do not detour during the tasks, while the trajectories for No Align (in grey) and No Priors (in black) have many movements that counteract themselves, indicating that this user was struggling to understand and align with the control strategy. In the worst case, participants were unable to complete the task with the No Priors model (see the *Avoid* task in Fig. 24) because no joystick inputs mapped to their intended direction, effectively causing them to get stuck at undesirable states.

To further validate that our model is learning the human’s preferences, we illustrate heatmaps over user inputs for the *Avoid* task in Fig. 26. Recall that this task requires moving the robot around an obstacle. Without the correct alignment, users default to the four cardinal directions (No Align), or warped circle-like motions (No Priors). By contrast, under All Priors the users smoothly moved the joystick around an even distribution, taking full advantage of the joystick’s $[-1, +1]$ workspace along both axes.

Summary In this user study we explored different approaches for learning the alignment model between joystick inputs and latent actions. We compared learning the alignment model

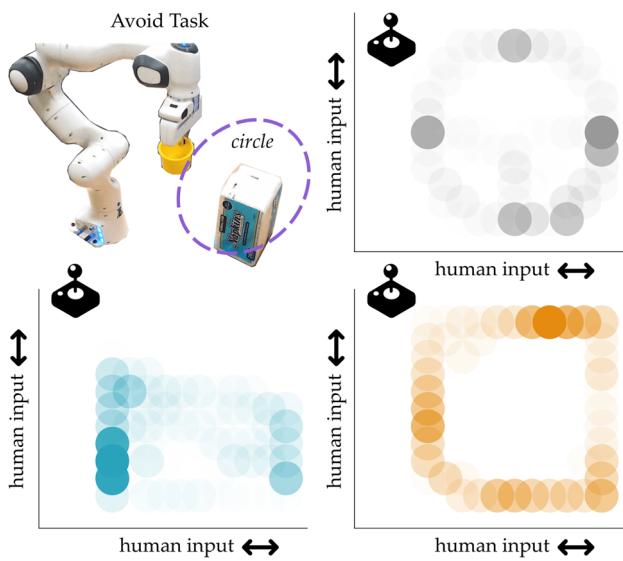


Fig. 26 Heatmaps of the participants’ joystick inputs during the *Avoid* task. For No Align in the upper right, people primarily used the cardinal directions. For No Priors in the bottom left, the joystick inputs were not clearly separated, and no clear pattern was established. For our All Priors model on the bottom right, however, we observed that the human inputs were *evenly distributed*. This indicates that the users smoothly completed the task by continuously manipulating the joystick in the range $[-1, +1]$ along both axes

with and without intuitive priors, and found that including these priors improved user performance, particularly in challenging tasks (**H1**).

10 Case study with disabled users

Our previous user studies involved non-disabled participants. Overall, these studies demonstrate the potential advantages of learned latent actions and support our proposed algorithm. But we still need to determine whether these results transfer

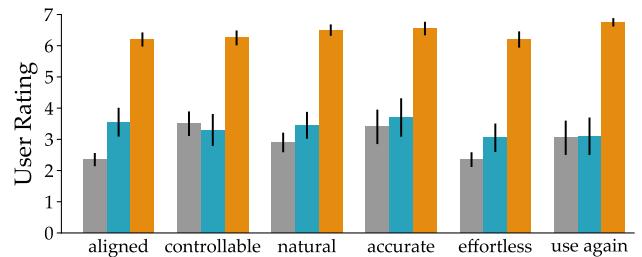


Fig. 27 Results from our 7-point Likert-scale survey after the alignment user study. The legend is the same as in Fig. 25. Higher ratings indicate agreement. Users thought that our learned model with intuitive priors aligned with their preferences, was easy to control, and improved efficiency—plus they would choose to use it again. Pairwise comparisons between our approach and the baselines are statistically significant across the board

to our target population; accordingly, here we test whether *disabled* persons can leverage latent actions to teleoperate assistive robots during eating tasks. This case study explores conditions and tasks similar to the user study from Sect. 9.3, but now with the added dimension of two disabled users that are familiar with assistive robot arms.

Participants We recruited two adult males with disabilities (ages 28 and 42) who require assistance when eating. The first participant has three years of experience with assistive robot arms, and the second participant has 2 years of experience with these robots.

Experimental setup In order to ensure safety during the COVID-19 pandemic, we developed a *remote control* interface where participants teleoperated an on-campus robot in real-time from their own homes (see Fig. 28). Participants interacted with a virtual joystick while watching live-streamed video of the robot arm. Video showed the robot and task from two angles: a front-view and a side-view. Just like the previous in-person studies, we used the participant’s joystick inputs to control the 7-DoF motion of the robot arm. We worked with both participants to minimize communica-

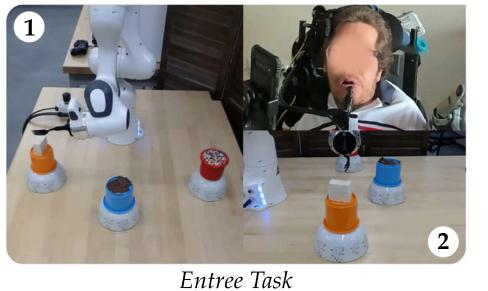
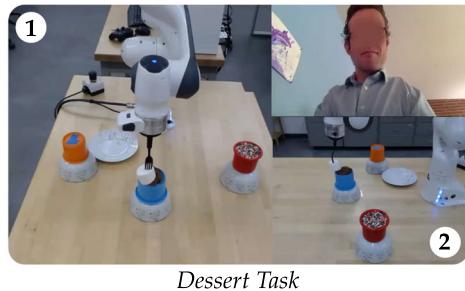


Fig. 28 Experimental setup for our case study with disabled persons (Sect. 10). Two adult males who employ assistive devices when eating volunteered to participate in the study. Due to safety restrictions, this study was conducted remotely: participants used an online joystick to teleoperate our robot in real-time while watching live-streamed video.



Here we show examples of the participants’ views during *Entree* and *Dessert* tasks. Camera (1) is a front-view of the robot and the high-level goals, and camera (2) is a side view of the same. Enlarged images of the participants are shown in the top right of each frame

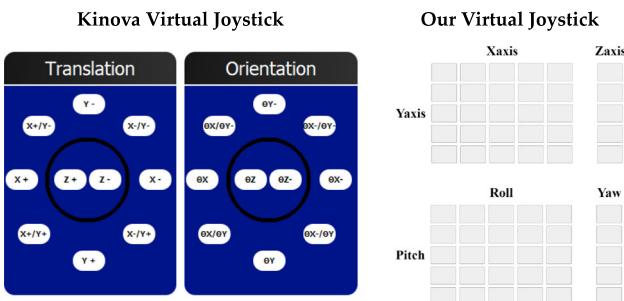


Fig. 29 Comparison between the End-Effector condition on a Kinova assistive robot arm (Kinova Software Development Kit User Guide 2021) (left) and our End-Effector implementation (right). We introduced a grid so that this virtual joystick was as close as possible to the continuous joysticks we previously used during our studies with non-disabled persons. We also consulted with our users while creating this interface to ensure that it matched their expectations

tion latency and position the cameras effectively; however, we recognize that delays and depth-perception may affect the results of these experiments.

As in Sect. 9.3, participants attempted to assemble two dishes: an *Entree* task and a *Dessert* task. The *Entree* task involved reaching for a block of tofu and then precisely cutting off a slice. The *Dessert* task had three steps: (i) reaching for and stabbing a marshmallow, (ii) carefully scooping that marshmallow in icing, and then (iii) dipping the marshmallow in sprinkles.

Independent variables We compared two different control schemes: **End-Effector** and **LA+SA**. With End-Effector the user directly controls the position and orientation of the fork attached to the end of the assistive robot arm (Herlant et al. 2016; Newman et al. 2018; Javdani et al. 2018). Participants here interact with two separate sets of joysticks, one for linear motion and a second for angular motion. We collaborated with both participants to ensure that this End-Effector setup closely resembled the control interface they typically use on their assistive robot arms (see Fig. 29). The LA+SA condition is our proposed approach (Algorithm 1), which combines latent actions with shared autonomy. We omit the alignment model f because of the time constraints of our voluntary participants; thus, the latent action z is set equal to the joystick input u .

Dependent measures To understand the effects of learned latent actions, we measured the total time taken to complete each task (*Total Time*) and the amount of time during the task where users were not providing joystick inputs (*Idle Time*). We also recorded the distance between the robot's fork and the closest goal—i.e., the tofu, marshmallow, icing, or sprinkles. This *Goal Error* serves as a proxy metric for the accuracy of the robot's high-level reaching. After participants completed the entire experiment we administered a short, open-ended survey to elicit their free-response feedback.

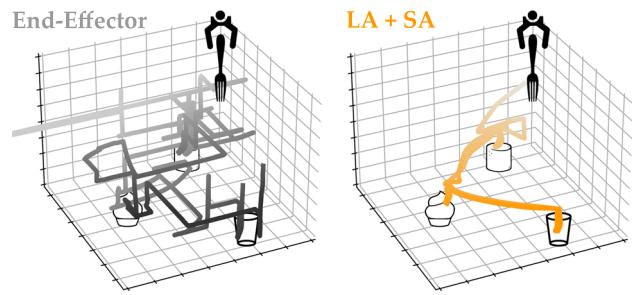


Fig. 30 The trajectories the robot's fork follows with **End-Effector** and **LA+SA** conditions in the *Dessert* task. The fork starts above the goals and moves to stab a marshmallow, scoop it in icing, and then dip it in a cup of sprinkles. We overlay the trajectories for both disabled participants

Procedure We applied a within-subjects design, where both participants completed the Dessert and Entree tasks using End-Effector and LA+SA conditions. User 1 started with the End-Effector condition and User 2 started with LA+SA. Both users completed the Dessert task first before doing the Entree task. So that the users could familiarize themselves with the controller and environment, participants were allotted up to 5 minutes of practice time with each condition.

Hypotheses We tested two main hypotheses:

- H1.** Disabled users will complete the eating tasks more quickly and accurately with our combination of latent actions and shared autonomy.
- H2.** Disabled users will subjectively prefer LA+SA to their current control approach (End-Effector).

Results The objective results of our case study are visualized in Figs. 30, 31, and 32. In Fig. 30 we display the motion of the robot's end-effector for the *Dessert* task: comparing the trajectories, we observe that End-Effector resulted in disjointed motions where users constantly switched between controlling the x , y , or z position of the robot's fork. By contrast, LA+SA helped users smoothly and directly move between goals.

To explore **H1** we specifically analyzed the *Goal Error* and *Total Time* for both tasks and participants (Figs. 31 and 32). Across disabled users we noticed a clear trend: our proposed LA+SA helped participants accurately reach their high-level goals while reducing the *Total Time* and *Idle Time* required to complete the tasks. These trends match our results with non-disabled users (see Sect. 9.3), and suggest that combining latent actions with shared autonomy improves objective performance during eating tasks.

We also asked for free-form feedback after the experiment to better understand the perspective of disabled users. The

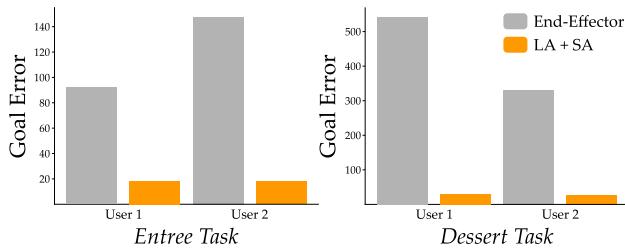


Fig. 31 Error between the robot’s fork and the closest goal during both *Entree* and *Dessert* tasks. We separately show the results for each disabled user

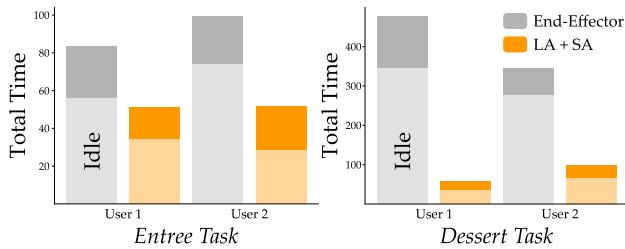


Fig. 32 Total time taken to complete the task (solid) and the idle time where users were trying to select their joystick input (light). Time is measured in seconds. As before, we separately show the results for each disabled user. Notice that the *Dessert* task took both users over 5 minutes with **End-Effector**, but less than 2 minutes with **LA+SA**

participants’ responses generally supported **H2**. User 1 stated that⁴:

Comparing End-Effector to LA+SA, the former was a lot harder, and LA+SA was way easier in probably every aspect. LA+SA was a little bit confusing in the sense that I wasn’t sure right off the bat what the joystick directions meant, but after using it for a minute or two it was really, really intuitive. Overall, LA+SA was great and I would definitely use it.

User 2 similarly mentioned that:

With LA+SA it was much easier to get those broad strokes of where I wanted to go, and more intuitive in how the robot moved, together with a simpler interface.

Summary This small-scale case study with two disabled users compared their typical end-effector control interface to our proposed latent action approach. Both participants performed better with learned latent actions (**H1**): they moved the robot closer to their high-level goals and completed the task in less total time. Users also perceived latent actions as a better approach for the two assistive eating tasks (**H2**).

⁴ We have added the condition names for clarity in these quotes. Participants did not know what the conditions were during the user study, and referred to them as “the first one” or “the second one.”

11 Conclusion

Our user and case studies separately and collectively test the key parts of our approach from Algorithm 1. We learn latent actions from kinesthetic demonstrations, and then enable users to control assistive robots with these learned actions. Our results demonstrate that controlling robots with learned latent actions outperforms the baseline shared autonomy dataset as well as direct end-effector teleoperation. Incorporating shared autonomy with latent actions further increases performance: shared autonomy helps the user reach and maintain high-level goals, while latent actions focus on low-level, precise manipulation. We also leveraged our semi-supervised approach to learn the human’s personalized alignment between joystick inputs and latent actions—in practice, this reduced the number of queries the human needed to answer, and resulted in more efficient task completion than one-size-fits-all alternatives. Overall, our latent action approach is a step towards intuitive, user-friendly control of assistive feeding robots. Our case study with two disabled users suggests that, in practice, controlling robots with learned latent actions makes assistive eating easier.

Limitations One key limitation of our approach occurs when the user encounters a new task never seen when training the latent actions. Here we cannot rely on latent actions—but we can revert to a baseline teleoperations scheme (e.g., end-effector control), and let the user complete the task using this default teleoperation mapping. We then include the demonstrated behavior within \mathcal{D} , retrain, and leverage learned latent actions *the next time* we encounter this new task. We emphasize that disabled persons can always provide new demonstrations by reverting to the standard, pre-defined teleoperation scheme to control the robot. However, directly retraining our learned latent actions on this updated dataset presents some new challenges: (i) determining if we have seen enough data so that the learned latent actions will perform robustly and (ii) ensuring that learning new latent actions does not interfere with or override previously learned latent actions. Our future work focuses on this challenge—we envision assistive robots that continuously alternate between learned latent actions and end-effector control to balance between compact, intuitive embeddings and full, high-dimensional control over the robot.

References

- Argall, B. D. (2018). Autonomy in rehabilitation robotics: An intersection. *Annual Review of Control, Robotics, and Autonomous Systems*, 1, 441–463.
- Aronson, R. M., Santini, T., Kübler, T. C., Kasneci, E., Srinivasa, S., & Admoni, H. (2018). Eye-hand behavior in human-robot shared manipulation. In *ACM/IEEE international conference on human-robot interaction* (pp. 4–13).

- Artemiadis, P. K., & Kyriakopoulos, K. J. (2010). EMG-based control of a robot arm using low-dimensional embeddings. *IEEE Transactions on Robotics*, 26(2), 393–398.
- Bajcsy, A., Losey, D. P., O’Malley, M. K., & Dragan, A. D. (2018). Learning from physical human corrections, one feature at a time. In *ACM/IEEE international conference on human-robot interaction* (pp. 141–149).
- Bhattacharjee, T., Gordon, E. K., Scalise, R., Cabrera, M. E., Caspi, A., Cakmak, M., & Srinivasa, S. S. (2020). Is more autonomy always better? Exploring preferences of users with mobility impairments in robot-assisted feeding. In *ACM/IEEE international conference on human-robot interaction* (pp. 181–190).
- Broad, A., Abraham, I., Murphey, T., & Argall, B. (2020). Data-driven Koopman operators for model-based shared control of human-machine systems. *The International Journal of Robotics Research*.
- Brose, S. W., Weber, D. J., Salatin, B. A., Grindle, G. G., Wang, H., Vazquez, J. J., & Cooper, R. A. (2010). The role of assistive robotics in the lives of persons with disability. *American Journal of Physical Medicine & Rehabilitation*, 89(6), 509–521.
- Canal, G., Alenyà, G., & Torras, C. (2016). Personalization framework for adaptive robotic feeding assistance. In *International conference on social robotics* (pp. 22–31).
- Carlson, T., & Millan, Jd. R. (2013). Brain-controlled wheelchairs: A robotic architecture. *IEEE Robotics & Automation Magazine*, 20(1), 65–73.
- Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., & Thomas, P. (2019). Learning action representations for reinforcement learning. In *International conference on machine learning* (pp. 941–950).
- Chapelle, O., Scholkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.
- Ciocarlie, M. T., & Allen, P. K. (2009). Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28(7), 851–867.
- Co-Reyes, J., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., & Levine, S. (2018). Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International conference on machine learning* (pp. 1009–1018).
- Doersch, C. (2016). Tutorial on variational autoencoders. arXiv preprint [arXiv:1606.05908](https://arxiv.org/abs/1606.05908).
- Dragan, A. D., & Srinivasa, S. S. (2013). A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7), 790–805.
- Edwards, A., Sahni, H., Schroecker, Y., & Isbell, C. (2019). Imitating latent policies from observation. In *International conference on machine learning* (pp. 1755–1763).
- Feng, R., Kim, Y., Lee, G., Gordon, E. K., Schmittle, M., Kumar, S., Bhattacharjee, T., & Srinivasa, S. S. (2019). Robot-assisted feeding: Generalizing skewering strategies across food items on a realistic plate. In *International symposium on robotics research*.
- Gallenberger, D., Bhattacharjee, T., Kim, Y., & Srinivasa, S. S. (2019). Transfer depends on acquisition: Analyzing manipulation strategies for robotic feeding. In *ACM/IEEE international conference on human-robot interaction* (pp. 267–276).
- Gopinath, D., Jain, S., & Argall, B. D. (2016). Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters*, 2(1), 247–254.
- Gordon, E. K., Meng, X., Bhattacharjee, T., Barnes, M., & Srinivasa, S. S. (2020). Adaptive robot-assisted feeding: An online learning framework for acquiring previously unseen food items. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 9659–9666).
- Herlant, L. V., Holladay, R. M., & Srinivasa, S. S. (2016). Assistive teleoperation of robot arms via automatic time-optimal mode switching. In *ACM/IEEE international conference on human-robot interaction* (pp. 35–42).
- Jacobsson, C., Axelsson, K., Österlind, P. O., & Norberg, A. (2000). How people with stroke and healthy older people experience the eating process. *Journal of Clinical Nursing*, 9(2), 255–264.
- Jain, S., & Argall, B. (2019). Probabilistic human intent recognition for shared autonomy in assistive robotics. *ACM Transactions on Human-Robot Interaction*, 9(1), 1–23.
- Javdani, S., Admoni, H., Pellegrinelli, S., Srinivasa, S. S., & Bag nell, J. A. (2018). Shared autonomy via hindsight optimization for teleoperation and teaming. *The International Journal of Robotics Research*, 37(7), 717–742.
- Jeon, H. J., Losey, D. P., & Sadigh, D. (2020). *Shared autonomy with learned latent actions*. In *Robotics: Science and systems*.
- Jonschkowski, R., & Brock, O. (2014). State representation learning in robotics: Using prior knowledge about physical interaction. In *Robotics: Science and systems*.
- Karamcheti, S., Zhai, A. J., Losey, D. P., & Sadigh, D. (2021). Learning visually guided latent actions for assistive teleoperation. In *Learning for dynamics & control*.
- Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *International conference on learning (representations)*.
- Kinova Software Development Kit User Guide (2019) Accessed May 24, 2021. https://www.kinovarobotics.com/sites/default/files/UG-008_KINOVA_Software_development_kit-User_guide_EN_R02%20%281%29.pdf.
- Li, M., Losey, D. P., Bohg, J., & Sadigh, D. (2020). Learning user-preferred mappings for intuitive robot control. In *IEEE international conference on intelligent robots and systems*.
- Losey, D. P., Srinivasan, K., Mandlekar, A., Garg, A., & Sadigh, D. (2020). Controlling assistive robots with learned latent actions. In *IEEE international conference on robotics and automation* (pp. 378–384).
- Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., & Sermanet, P. (2019). Learning latent plans from play. In *Conference on robot learning* (pp. 1113–1132).
- Matrone, G. C., Cipriani, C., Carrozza, M. C., & Magenes, G. (2012). Real-time myoelectric control of a multi-fingered hand prosthesis using principal components analysis. *Journal of Neuroengineering and Rehabilitation*, 9(1), 40.
- Mitzner, T. L., Sanford, J. A., & Rogers, W. A. (2018). Closing the capacity-ability gap: Using technology to support aging with disability. *Innovation in Aging*, 2(1), igy008.
- Muellung, K., Venkatraman, A., Valois, J. S., Downey, J. E., Weiss, J., Javdani, S., et al. (2017). Autonomy infused teleoperation with application to brain computer interface controlled manipulation. *Autonomous Robots*, 41(6), 1401–1422.
- Naotunna, I., Perera, C. J., Sandaruwan, C., Gopura, R., & Lalitharatne, T. D. (2015). Meal assistance robots: A review on current status, challenges and future directions. In *IEEE/SICE international symposium on system integration* (pp. 211–216).
- Newman, B. A., Aronson, R. M., Srinivasa, S. S., Kitani, K., & Admoni, H. (2018) HARMONIC: A multimodal dataset of assistive human-robot collaboration. arXiv preprint [arXiv:1807.11154](https://arxiv.org/abs/1807.11154).
- Nikolaidis, S., Zhu, Y. X., Hsu, D., & Srinivasa, S. (2017). Human-robot mutual adaptation in shared autonomy. In *ACM/IEEE international conference on human-robot interaction* (pp. 294–302).
- Noseworthy, M., Paul, R., Roy, S., Park, D., & Roy, N. (2020). Task-conditioned variational autoencoders for learning movement primitives. In *Conference on robot learning* (pp. 933–944).
- Pacelli, V., & Majumdar, A. (2020). Learning task-driven control policies via information bottlenecks. In *Robotics: Science and systems*.
- Park, D., Hoshi, Y., Mahajan, H. P., Kim, H. K., Erickson, Z., Rogers, W. A., & Kemp, C. C. (2020). Active robot-assisted feeding with a general-purpose mobile manipulator: Design, evaluation, and lessons learned. *Robotics and Autonomous Systems*, 124, 103344.
- Pertsch, K., Lee, Y., & Lim, J. J. (2020). Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*.

- Rakita, D., Mutlu, B., & Gleicher, M. (2017). A motion retargeting method for effective mimicry-based teleoperation of robot arms. In *ACM/IEEE international conference on human-robot interaction* (pp. 361–370).
- Rakita, D., Mutlu, B., Gleicher, M., & Hiatt, L. M. (2019). Shared control-based bimanual robot manipulation. *Science Robotics*, 4(30).
- Reddy, S., Dragan, A., & Levine, S. (2018). Where do you think you're going?: Inferring beliefs about dynamics from behavior. *Advances in Neural Information Processing Systems*, 31, 1454–1465.
- Reddy, S., Dragan, A. D., & Levine, S. (2018). Shared autonomy via deep reinforcement learning. In *Robotics: Science and systems*.
- Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*. Wiley.
- Taylor, D. M. (2018). Americans with disabilities: 2014. US Census Bureau
- Watter, M., Springenberg, J., Boedecker, J., & Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems* (pp. 2746–2754).
- Xie, A., Losey, D. P., Tolsma, R., Finn, C., & Sadigh, D. (2020) Learning latent representations to influence multi-agent interaction. In *Conference on robot learning*

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Dylan P. Losey is an Assistant Professor in the Mechanical Engineering Department at Virginia Tech. His research interests lie at the intersection of human-robot interaction, machine learning, and control theory. Specifically, his group develops learning and control algorithms to personalize robots that collaborate with people. Dylan Losey was previously a postdoctoral scholar in the Computer Science Department at Stanford University. He received his doctoral degree in Mechanical Engineering from Rice University in 2018, and his bachelor's degree in Mechanical Engineering from Vanderbilt University in 2014.



Hong Jun Jeon is a Ph.D. candidate in the Computer Science Department at Stanford University. He received his bachelor's degree in Computer Science from the University of California, Berkeley in 2018.



Mengxi Li is a Ph.D. candidate in the Electrical Engineering Department at Stanford University. She is broadly interested in the intersection of artificial intelligence, human robot interaction, and multi-agent systems. Before coming to Stanford, Mengxi Li received her B.S. degree from Tsinghua University in 2018.



Krishnan Srinivasan is a Ph.D. candidate in the Computer Science Department at Stanford University. He received his bachelor's degree in Computer Science from Yale University in 2018.



Ajay Mandlekar is a Ph.D. candidate in the Computer Science Department at Stanford University. He works on applying Reinforcement Learning and Imitation Learning to Robotics. Specifically, he is interested in building systems and algorithms to allow robots to leverage human insight for manipulation. Ajay Mandlekar received his M.S. from Stanford University in 2018, and his B.S. from the California Institute of Technology in 2016.



Animesh Garg is an Assistant Professor of Computer Science at University of Toronto (UofT) and a Faculty Member at the Vector Institute. He directs the UofT People, AI and Robotics group. He is affiliated with Mechanical and Industrial Engineering (courtesy) and UofT Robotics Institute. He is also a Sr. Research Scientist at Nvidia. Animesh Garg earned M.S. in Computer Science and Ph.D. in Operations Research from the University of California, Berkeley. He was later a postdoc in the Computer Science Department at Stanford University.



Jeannette Bohg is an Assistant Professor of Computer Science at Stanford University. She was a group leader at the Autonomous Motion Department (AMD) of the MPI for Intelligent Systems until September 2017. Before joining AMD in January 2012, Jeannette Bohg was a PhD student at the Division of Robotics, Perception and Learning (RPL) at KTH in Stockholm. In her thesis, she proposed novel methods towards multimodal scene understanding for robotic grasping. She also studied

at Chalmers in Gothenburg and at the Technical University in Dresden where she received her Master in Art and Technology and her Diploma in Computer Science, respectively. Her research focuses on perception and learning for autonomous robotic manipulation and grasping. She is specifically interested in developing methods that are goal-directed, real-time and multi-modal such that they can provide meaningful feedback for execution and learning. Jeannette Bohg has received several awards, most notably the 2019 IEEE International Conference on Robotics and Automation (ICRA) Best Paper Award, the 2019 IEEE Robotics and Automation Society Early Career Award and the 2017 IEEE Robotics and Automation Letters (RA-L) Best Paper Award.



Dorsa Sadigh is an assistant professor in Computer Science and Electrical Engineering at Stanford University. Her research interests lie in the intersection of robotics, learning, and control theory. Specifically, she is interested in developing algorithms for safe and adaptive human-robot interaction. Dorsa has received her doctoral degree in Electrical Engineering and Computer Sciences (EECS) from UC Berkeley in 2017, and has received her bachelor's degree in EECS from UC Berkeley in 2012. She

is awarded the NSF CAREER award, the AFOSR Young Investigator award, the IEEE TCCPS early career award, the Google Faculty Award, and the Amazon Faculty Research Award.