*Article*

The International Journal of Robotics Research
2018, Vol. 37(7) 717–742

DOI: 10.1177/0278364918776060
journals.sagepub.com/home/ijr

# Shared autonomy via hindsight optimization for teleoperation and teaming

**Shervin Javdani[1], Henny Admoni[1], Stefania Pellegrinelli[2], Siddhartha S. Srinivasa[1] and J. Andrew Bagnell[1]**

**Abstract**

*In shared autonomy, a user and autonomous system work together to achieve shared goals. To collaborate effectively, the autonomous system must know the user's goal. As such, most prior works follow a predict-then-act model, first predicting the user's goal with high confidence, then assisting given that goal. Unfortunately, confidently predicting the user's goal may not be possible until they have nearly achieved it, causing predict-then-act methods to provide little assistance. However, the system can often provide useful assistance even when confidence for any single goal is low (e.g. move towards multiple goals). In this work, we formalize this insight by modeling shared autonomy as a partially observable Markov decision process (POMDP), providing assistance that minimizes the expected cost-to-go with an unknown goal. As solving this POMDP optimally is intractable, we use hindsight optimization to approximate. We apply our framework to both shared-control teleoperation and human–robot teaming. Compared with predict-then-act methods, our method achieves goals faster, requires less user input, decreases user idling time, and results in fewer user–robot collisions.*

**Keywords**

Physical human–robot interaction, telerobotics, rehabilitation robotics, personal robots, human performance augmentation

## 1. Introduction

Human–robot collaboration studies interactions between humans and robots sharing a workspace. One instance of collaboration arises in shared autonomy, where both the user and robotic system act simultaneously to achieve shared goals. For example, in shared control teleoperation (Aigner and McCarragher, 1997; Debus et al., 2000; Dragan and Srinivasa, 2013b; Goertz, 1963; Rosenberg, 1993), both the user and system control a single entity, the robot, to achieve the user's goal. In human–robot teaming, the user and system act independently to achieve a set of related goals (Arai et al., 2010; Dragan and Srinivasa, 2013a; Gombolay et al., 2014; Hoffman and Breazeal, 2007; Koppula and Saxena, 2013; Mainprice and Berenson, 2013; Nikolaidis et al., 2017a).

Although each instance of shared autonomy has many unique requirements, they share a key common challenge: for the autonomous system to be an effective collaborator, it needs to know the user's goal. For example, feeding with shared control teleoperation, an important task for assistive robotics (Chung et al., 2013), requires knowing what the user wants to eat (Figure 1a). Wrapping gifts with a human–robot team requires knowing which gift the user will wrap to avoid getting in their way and hogging shared resources (Figure 1b).

In general, the system does not know the user's goal a priori. We could alleviate this issue by requiring users to explicitly specify their goals (e.g. through voice commands). However, there are often a continuum of goals to choose from (e.g. location to place an object, size to cut a bite of food), making it impossible for users to precisely specify their goals. Furthermore, prior works suggest requiring explicit communication leads to ineffective collaboration (Goodrich and Olsen, 2003; Green et al., 2007; Vanhooydonck et al., 2003). Instead, implicit information should be used to make collaboration seamless. In shared autonomy, this suggests utilizing sensing of the environment and user actions to infer the user's goal. This idea has been successfully applied for shared control teleoperation (Aarno and Kragic, 2008; Carlson and Demiris, 2012; Dragan and Srinivasa, 2013b; Hauser, 2013; Kofman et al., 2005; Kragic et al., 2005; Li and Okamura, 2003; Muelling

[1]The Robotics Institute, Carnegie Mellon University, USA
[2]ITIA-CNR, Institute of Industrial Technologies and Automation, National Research Council, Italy

**Corresponding author:**
Shervin Javdani, Carnegie Mellon University, Robotics Institute, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA.
Email: sjavdani@cmu.edu

(a) (b)

**Fig. 1.** We can provide useful assistance even when we do not know the user's goal. (a) Our feeding experiment, where the user wants to eat one of the bites of food on the plate. With an unknown goal, our method autonomously orients the fork and moves towards all bites. In contrast, predict-then-act methods only helped position the fork at the end of execution. Users commented that the initial assistance orienting the fork and getting close to all bites was the most helpful, as this was the most time-consuming portion of the task. (b) Our teaming experiment, where the user wraps a box, and the robot must stamp a different box. Here, the user's motion so far suggests their goal is likely either the green or white box. Though we cannot confidently predict their single goal, our method starts making progress for the other boxes.

et al., 2015; Yu et al., 2005) and human-robot teaming (Hoffman and Breazeal, 2007; Koppula and Saxena, 2013; Lasota and Shah, 2015; Macindoe et al., 2012; Mainprice and Berenson, 2013; Nguyen et al., 2011).

As providing effective assistance requires knowing the user's goal, most shared autonomy methods do not assist when the goal is unknown. These works split shared autonomy into two parts: (1) predict the user's goal with high probability; and (2) assist for that single goal, potentially using prediction confidence to regulate assistance. We refer to this approach as predict-then-act. Although this has been effective in simple scenarios with few goals (Carlson and Demiris, 2012; Dragan and Srinivasa, 2013b; Kofman et al., 2005; Koppula and Saxena, 2013; Muelling et al., 2015; Yu et al., 2005), it is often impossible to predict the user's goal until the end of execution (e.g. cluttered scenes), causing these methods to provide little assistance. Addressing this lack of assistance is of great practical importance: with uncertainty over only goals in our feeding experiment, a predict-then-act method provided assistance for only 31% of the time on average, taking 29.4 seconds on average before the confidence threshold was initially reached.

In this work, we present a general framework for goal-directed shared autonomy that does not rely on predicting a single user goal (Figure 2). We assume the user's goal is fixed (e.g. they want a particular bite of food), and the autonomous system should adapt to the user goal.[1] Our key insight is that there are useful assistance actions for distributions over goals, even when confidence for a particular goal is low (e.g. move towards multiple goals; Figure 1). We formalize this notion by modeling our problem as a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998), treating the user's goal as hidden state. When the system is uncertain of the user goal, our framework naturally optimizes for an assistance action that is helpful for many goals. When the system confidently predicts a single user goal, our framework focuses assistance given that goal (Figure 3).

As our state and action spaces are both continuous, solving for the optimal action in our POMDP is intractable. Instead, we approximate using QMDP (Littman et al., 1995), also referred to as hindsight optimization (Chong et al., 2000; Yoon et al., 2008). This approximation has many properties suitable for shared autonomy: it is computationally efficient, works well when information is gathered easily (Koval et al., 2014), and will not oppose the user to gather information. The result is a system that minimizes the expected cost-to-go to assist for any distribution over goals.

We apply our framework in user study evaluations for both shared control teleoperation and human–robot teaming. For shared control teleoperation, users performed two tasks: a simpler object-grasping task (Section 4.1); and a more difficult feeding task (Section 4.2). In both cases, we find that our POMDP-based method enabled users to achieve goals faster and with less joystick input than a state-of-the-art predict-then-act method (Dragan and Srinivasa, 2013b). Subjective user preference differed by task, with no statistical difference for the simpler object-grasping task, and users preferring our POMDP method for the more difficult feeding task.

For human–robot teaming (Section 5.1), the user and robot performed a collaborative gift-wrapping task, where both agents had to manipulate the same set of objects while avoiding collisions. We found that users spent less time idling and less time in collision while collaborating with a robot using our method. However, results for total task completion time are mixed, as predict-then-act methods are able to take advantage of more optimized motion planners, enabling faster execution once the user goal is predicted confidently.

## 2. Related works

### 2.1. Shared control teleoperation

Shared control teleoperation has been used to assist disabled users using robotic arms (Katyal et al., 2014; Kim et al., 2012, 2006; McMullen et al., 2014; Muelling et al., 2015; Schröer et al., 2015) or wheelchairs (Argall, 2014; Carlson and Demiris, 2012), operate robots remotely (Leeper et al., 2012; Shen et al., 2004; You and Hauser, 2011), decrease operator fatigue in surgical settings (Aarno et al., 2005; Kragic et al., 2005; Li et al., 2007; Marayong et al., 2003; Park et al., 2001), and many other applications. As such, there are a great many methods catering to the specific needs of each domain.

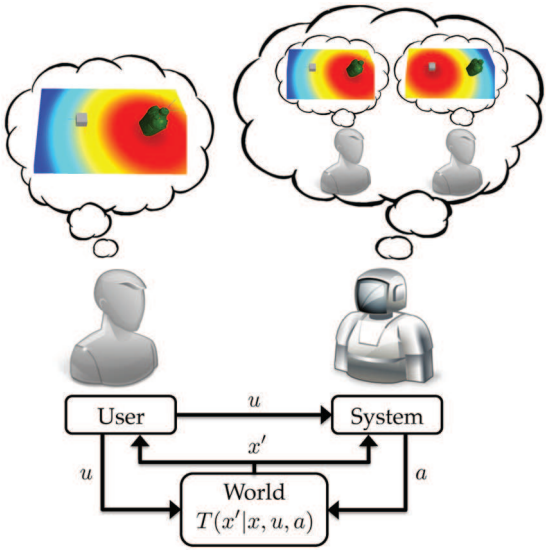



**Fig. 2.** Our shared autonomy framework. We assume the user executes a policy for their single goal, depicted as a heatmap plotting the value function at each position. Our shared autonomy system models all possible user goals and their corresponding policies. From user actions $u$, a distribution over goals is inferred. Using this distribution and the value functions for each goal, the system selects an action $a$. The world transitions from $x$ to $x'$. The user and shared autonomy system both observe this state, and repeat action selection.

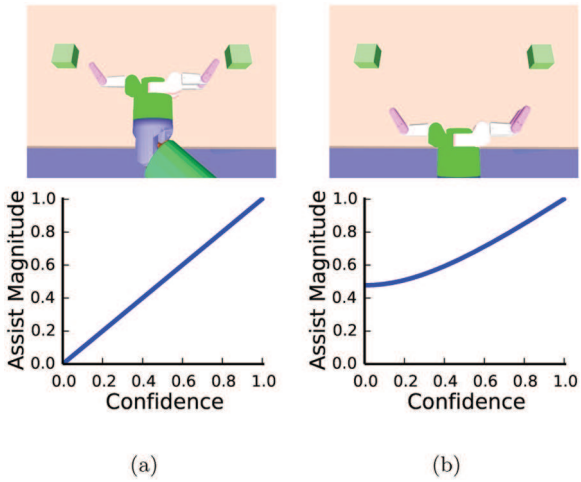



**Fig. 3.** Arbitration as a function of confidence with two goals. Confidence $= \max_g p(g) - \min_g p(g)$, which ranges from 0 (equal probability) to 1 (all probability on one goal). (a) The hand is directly between the two goals, where no action assists for both goals. As confidence for one goal increases, assistance increases. (b) From here, going forward assists progress towards both goals, enabling the assistance policy to make progress even with 0 confidence.

One common paradigm launches a fully autonomous takeover when some trigger is activated, such as a user command (Bien et al., 2004; Kim et al., 2012; Shen et al., 2004; Simpson, 2005), or when a goal predictor exceeds some confidence threshold (Fagg et al., 2004; Katyal et al., 2014; Kofman et al., 2005; McMullen et al., 2014). Others have utilized similar triggers to initiate a subtask in a sequence (Jain et al., 2015; Schröer et al., 2015). Although these systems are effective at accomplishing the task, studies have shown that users often prefer having more control (Kim et al., 2012).

Another line of work utilizes high-level user commands, and relies on autonomy to generate robot motions. Systems have been developed to enable users to specify an end-effector path in two dimensions, which the robot follows with full configuration space plans (Hauser, 2013; You and Hauser, 2011). Point-and-click interfaces have been used for object grasping with varying levels of autonomy (Leeper et al., 2012). Eye gaze has been utilized to select a target object and grasp position (Bien et al., 2004).

Another paradigm augments user inputs minimally to maintain some desired property, e.g. collision avoidance, without necessarily knowing exactly what goal the user wants to achieve. Sensing and complaint controllers have been used increase safety during teleoperation (Kim et al., 2006; Vogel et al., 2014). Potential field methods have been employed to push users away from obstacles (Crandall and Goodrich, 2002) and towards goals (Aigner and McCarragher, 1997). For assistive robotics using modal control, where users control subsets of the degrees of freedom (DOFs) of the robot in discrete modes (Figure 4), Herlant et al. (2016) demonstrated a method for automatic time-optimal mode switching.

Similarly, methods have been developed to augment user inputs to follow some constraint. Virtual fixtures, commonly used in surgical robotics settings, are employed to project user commands onto path constraints (e.g. straight lines only) (Aarno et al., 2005; Kragic et al., 2005; Li et al., 2007; Li and Okamura, 2003; Marayong et al., 2003; Park et al., 2001). Mehr et al. (2016) learned constraints online during execution, and apply constraints softly by combining constraint satisfaction with user commands. Although these methods benefit from not needing to predict the user's goal, they generally rely on a high-DOF input, making their use limited for assistive robotics, where disabled users can only operate a few DOF at a time and thus rely on modal control (Herlant et al., 2016).

Blending methods (Dragan and Srinivasa, 2013b) attempt to bridge the gap between highly assistive methods with little user control and minimal assistance with higher user burden. User actions and full autonomy are treated as two independent sources, which are combined by some arbitration function that determines the relative contribution of each (Figure 5). Dragan and Srinivasa (2013b) show that many methods of shared control teleoperation (e.g. autonomous takeover, potential field methods, virtual fixtures) can be generalized as blending with a particular arbitration function.

Blending is one of the most widely used shared control teleoperation paradigms due to computational

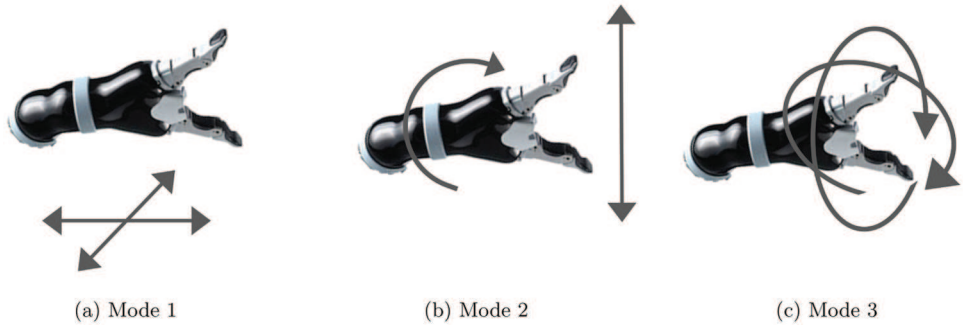



**Fig. 4.** Modal control used in our feeding experiment on the Kinova MICO, with three control modes and a two-DOF input device. Fewer input DOFs means more modes are required to control the robot.

efficiency, simplicity, and empirical effectiveness (Carlson and Demiris, 2012; Dragan and Srinivasa, 2013b; Gopinath et al., 2016; Li et al., 2011; Muelling et al., 2015). However, blending has two key drawbacks. First, as two independent decisions are being combined without evaluating the action that will be executed, catastrophic failure can result even when each independent decision would succeed (Trautman, 2015). Second, these systems rely on a predict-then-act framework, predicting the single goal the user is trying to achieve before providing any assistance. Often, assistance will not be provided for large portions of execution while the system has low confidence in its prediction, as we found in our feeding experiment (Section 4.2).

Recently, Hauser (2013) presented a system that provides assistance for a distribution over goals. Like our method, this policy-based method minimizes an expected cost-to-go while receiving user inputs (Figure 6). The system iteratively plans trajectories given the current user goal distribution, executes the plan for some time, and updates the distribution given user inputs. In order to efficiently compute the trajectory, it is assumed that the cost function corresponds to squared distance, resulting in the calculation decomposing over goals. Our model generalizes these notions, enabling the use of any cost function for which a value function can be computed.

In this work, we assume the user does not change their goal or actions based on autonomous assistance, putting the burden of goal inference entirely on the system. Nikolaidis et al. (2017c) presented a game-theoretic approach to shared control teleoperation, where the user adapts to the autonomous system. Each user has an adaptability, modeling how likely the user is to change goals based on autonomous assistance. They used a POMDP to learn this adaptability during execution. Although more general, this model is computationally intractable for continuous state and actions.

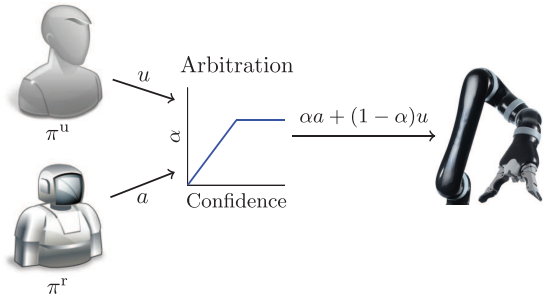



**Fig. 5.** Blend method for shared control teleoperation. The user and robot are both modeled as separate policies $\pi^{\mathrm{u}}$ and $\pi^{\mathrm{r}}$, each independently providing actions $u$ and $a$ for a single goal. These actions are combined through a specified arbitration function, which generally uses some confidence measure to augment the magnitude of assistance. This combined action is executed on the robot.

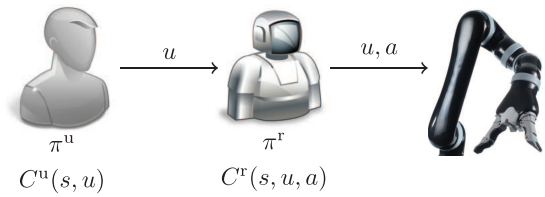



**Fig. 6.** Policy method for shared control teleoperation. The user is modeled as a policy $\pi^{\mathrm{u}}$, which selects user input $u$ to minimizes the expected sum of user costs $C^{\mathrm{u}}(x,u)$. The user input $u$ is provided to the system policy $\pi^{\mathrm{r}}$, which then selects action $a$ to minimize its expected sum of costs $C^{\mathrm{r}}(s,u,a)$. Both actions are passed to the robot for execution. Unlike the blend method, the user and robot actions are not treated separately, which can lead to catastrophic failure (Trautman, 2015). Instead, the robot action $a$ is optimized given the user action $u$.

## 2.2. Human–robot teaming

In human–robot teaming, robot action selection that models and optimizes for the human teammate leads to better collaboration. Hoffman and Breazeal (2007) showed that using predictions of a human collaborator during action selection led to more efficient task completion and more favorable perception of robot contribution to team success. Lasota and Shah (2015) showed that planning to avoid portions of the workspace the user will occupy led to faster task completion, less user and robot idling time, greater user satisfaction, and greater perceived safety and comfort. Arai

et al. (2010) showed that users feel high mental strain when a robot collaborator moves too close or too quickly.

Motion planners have been augmented to include user models and collaboration constraints. For static users, researchers have incorporated collaboration constraints such as safety and social acceptability (Sisbot et al., 2007), and task constraints such as user visibility and reachability (Mainprice et al., 2011; Pandey and Alami, 2010; Sisbot et al., 2010). For moving users, Mainprice and Berenson (2013) used a Gaussian mixture model to predict user motion, and select a robot goal that avoids the predicted user locations.

Similar ideas have been used to avoid moving pedestrians. Ziebart et al. (2009) learned a predictor of pedestrian motion, and use this to predict the probability a location will be occupied at each time step. They built a time-varying cost map, penalizing locations likely to be occupied, and optimized trajectories for this cost. Chung and Huang (2011) used A* search to predict pedestrian motions, including a model of uncertainty, and planned paths using these predictions. Bandyopadhyay et al. (2012) used fixed models for pedestrian motions, and focused on utilizing a POMDP framework with SARSOP (Kurniawati et al., 2008) for selecting good actions. Like our approach, this enables them to reason over the entire distribution of potential goals. They showed that this outperforms utilizing only the maximum likelihood estimate of goal prediction for avoidance.

Others developed methods for how the human–robot team should be structured. Gombolay et al. (2014) studied the effects of having the user and robot assign goals to each other. They found that users were willing to cede decision making to the robot if it resulted in greater team fluency (Gombolay et al., 2014). However, Gombolay et al. (2017) later showed that having the autonomous entity assign goals led to less situational awareness. Inspired by training schemes for human–human teams, Nikolaidis and Shah (2013) presented a human–robot cross-training method, where the user and robot iteratively switch roles to learn a shared plan. Their model leads to greater team fluency, more concurrent motions, greater perceived robot performance, and greater user trust. Koppula and Saxena (2013) used conditional random fields (CRFs) to predict the user goal (e.g. grasp cup), and have a robot achieve a complementary goal (e.g. pour water into cup).

Others have studied how robot motions can influence the belief of users. Sisbot et al. (2010) fixed the gaze of the robot on its goal to communicate intent. Dragan and Srinivasa (2013a) incorporated legibility into the motion planner for a robotic arm, causing the robot to exaggerate its motion to communicate intent. They showed this leads to more quickly and accurately predicting the robot intent (Dragan et al., 2013). Rezvani et al. (2016) studied the effects of conveying a robot's state (e.g. confidence in action selection, anomaly in detection) directly on a user interface for autonomous driving.

Recent works have gone one step further, selecting robot actions that not only change the beliefs of users, but also user actions. Nikolaidis et al. (2017a) modeled how likely users are to adopt the robot's policy based on robot actions. They utilized a POMDP to simultaneously learn this user adaptability while steering users to more optimal goals to achieve greater reward. Nikolaidis et al. (2017b) presented a more general game-theoretic approach where users changed their actions based on robot actions, while not completely adopting the robot's policy. Similarly, Sadigh et al. (2016b) generated motions for an autonomous car using predictions of how other drivers will respond, enabling them to change the behavior of other users, and infer the internal user state (Sadigh et al., 2016a).

Teaming with an autonomous agent has also been studied outside of robotics. Fern and Tadepalli (2010) have studied Markov decision processes (MDPs) and POMDPs for interactive assistants that suggest actions to users, who then accept or reject each action. They showed that optimal action selection even in this simplified model is PSPACE-complete. However, a simple greedy policy has bounded regret. Nguyen et al. (2011) and Macindoe et al. (2012) applied POMDPs to cooperative games, where autonomous agents simultaneously infer human intentions and take assistance actions. Like our approach, they modeled users as stochastically optimizing an MDP, and solved for assistance actions with a POMDP. In contrast to these works, our state and action spaces are continuous.

### 2.3. User prediction

A variety of models and methods have been used for intent prediction. Hidden Markov model (HMM)-based methods (Aarno et al., 2005; Aarno and Kragic, 2008; Kragic et al., 2005; Li and Okamura, 2003) predict subtasks or intent during execution, treating the intent as a latent state. Schrempf et al. (2007) used a Bayesian network constructed with expert knowledge. Koppula and Saxena (2013) extended CRFs with object affordance to predict potential human motions. Wang et al. (2013) learned a generative predictor by extending Gaussian process dynamical models (GPDMs) with a latent variable for intention. Hauser (2013) utilized a Gaussian mixture model over task types (e.g. reach, grasp) and predicted both the task type and continuous parameters for that type (e.g. movements) using Gaussian mixture autoregression.

Many successful works in shared autonomy utilize maximum entropy inverse optimal control (MaxEnt IOC) (Ziebart et al., 2008) for user goal prediction. Briefly, the user is modeled as a stochastic policy approximately optimizing some cost function. By minimizing the worst-case predictive loss, Ziebart et al. (2008) derived a model where trajectory probability decreases exponentially with cost. They then derive a method for inferring a distribution over goals from user inputs, where probabilities correspond to how efficiently the inputs achieve each goal

(Ziebart et al., 2009). A key advantage of this framework for shared autonomy is that the we can directly optimize for the cost function used to model the user.

Exact, global inference over these distributions is computationally infeasible in continuous state and action spaces. Instead, Levine and Koltun (2012) provided a method that considers the expert demonstrations as only locally optimal, and utilized Laplace's method about the expert demonstration to estimate the log likelihood during learning. Similarly, Dragan and Srinivasa (2013b) used Laplace's method about the optimal trajectory between any two points to approximate the distribution over goals during shared control teleoperation. Finn et al. (2016) simultaneously learned a cost function and policy consistent with user demonstrations using deep neural networks, utilizing importance sampling to approximate inference with few samples. Inspired by generative adversarial nets (Goodfellow et al., 2014), Ho and Ermon (2016) directly learned a policy to mimic the user through generative adversarial imitation learning.

We utilize a MaxEnt IOC-based approach (Ziebart et al., 2008) with a Laplace approximation (Dragan and Srinivasa, 2013b) in our framework due to empirical evidence of effectiveness in shared autonomy systems (Dragan and Srinivasa, 2013b; Muelling et al., 2015). We present our formulation in Section 3.4.

## 3. Framework

We present our framework for minimizing a cost function for shared autonomy with an unknown user goal. We assume the user's goal is fixed, and they take actions to achieve that goal without considering autonomous assistance. These actions are used to predict the user's goal based on how optimal the action is for each goal (Section 3.4). Our system uses this distribution to minimize the expected cost-to-go (Section 3.2). As solving for the optimal action is infeasible, we use hindsight optimization to approximate a solution (Section 3.3). For reference, see Table 1 for variable definitions.

### 3.1. Cost minimization with a known goal

We first formulate the problem for a known user goal, which we will use in our solution with an unknown goal. We model this problem as an MDP.

Formally, let $x \in X$ be the environment state (e.g. human and robot pose). Let $u \in U$ be the user actions, and $a \in A$ the robot actions. Both agents can affect the environment state: if the user takes action $u$ and the robot takes action $a$ while in state $x$, the environment stochastically transitions to a new state $x'$ through $T(x' \mid x, u, a)$.[2]

We assume the user has an intended goal $g \in G$ that does not change during execution. We augment the environment state with this goal, defined by $s = (x, g) \in X \times G$. We overload our transition function to model the transition in environment state without changing the goal, $T((x', g) \mid (x, g), u, a) = T(x' \mid x, u, a)$.

We assume access to a user policy for each goal $\pi^{\mathrm{u}}(u \mid s) = \pi^{\mathrm{u}}_g(u \mid x) = p(u \mid x, g)$. We model this policy using the MaxEnt IOC framework of Ziebart et al. (2008), where the policy corresponds to stochastically optimizing a cost function $C^{\mathrm{u}}(s, u) = C^{\mathrm{u}}_g(x, u)$. We assume the user selects actions based only on $s$, the current environment state and their intended goal, and does not model any actions that the robot might take. Details are given in Section 3.4.

The robot selects actions to minimize a cost function dependent on the user goal and action $C^{\mathrm{r}}(s, u, a) = C^{\mathrm{r}}_g(x, u, a)$. At each time step, we assume the user first selects an action, which the robot observes before selecting $a$. The robot selects actions based on the state and user inputs through a policy $\pi^{\mathrm{r}}(a \mid s, u) = p(a \mid s, u)$. We define the value function for a robot policy $V^{\pi^{\mathrm{r}}}$ as the expected cost-to-go from a particular state, assuming some user policy $\pi^{\mathrm{u}}$:

$$
\begin{aligned}
V^{\pi^{\mathrm{r}}}(s) &= \mathbb{E}\left[\sum_t C^{\mathrm{r}}(s_t, u_t, a_t) \mid s_0 = s\right] \\
u_t &\sim \pi^{\mathrm{u}}(\cdot \mid s_t) \\
a_t &\sim \pi^{\mathrm{r}}(\cdot \mid s_t, u_t) \\
s_{t+1} &\sim T(\cdot \mid s_t, u_t, a_t)
\end{aligned}
$$

The optimal value function $V^*$ is the cost-to-go for the best robot policy:

$$V^*(s) = \min_{\pi^{\mathrm{r}}} V^{\pi^{\mathrm{r}}}(s)$$

The action-value function $Q^*$ computes the immediate cost of taking action $a$ after observing $u$, and following the optimal policy thereafter:

$$Q^*(s, u, a) = C^{\mathrm{r}}(s, u, a) + \mathbb{E}\left[V^*(s')\right]$$

where $s' \sim T(\cdot \mid s, u, a)$. The optimal robot action is given by $\arg\min_a Q^*(s, u, a)$.

To make explicit the dependence on the user goal, we often write these quantities as

$$
\begin{aligned}
V_g(x) &= V^*(s) \\
Q_g(x, u, a) &= Q^*(s, u, a)
\end{aligned}
$$

Computing the optimal policy and corresponding action-value function is a common objective in reinforcement learning. We assume access to this function in our framework, and describe our particular implementation in the experiments.

### 3.2. Cost minimization with an unknown goal

We formulate the problem of minimizing a cost function with an unknown user goal as a POMDP. A POMDP maps a distribution over states, known as the belief $b$, to actions.

**Table 1.** Variable definitions.

| Symbol | Description |
|---|---|
| $x \in X$ | Environment state, e.g. robot and human pose |
| $g \in G$ | User goal |
| $s \in S$ | $s = (x, g)$. State and user goal |
| $u \in U$ | User action |
| $a \in A$ | Robot action |
| $C^{\mathrm{u}}(s,u) = C^{\mathrm{u}}_g(x,u)$ | Cost function for each user goal |
| $C^{\mathrm{r}}(s,u,a) = C^{\mathrm{r}}_g(x,u,a)$ | Robot cost function for each goal |
| $T(x' \mid x,u,a)$ | Transition function of environment state |
| $T((x',g) \mid (x,g),u,a) = T(x' \mid x,u,a)$ | User goal does not change with transition |
| $T^{\mathrm{u}}(x' \mid x,u) = T(x' \mid x,u,0)$ | User transition function assumes the user is in full control |
| $V_g(x) = V^*(s)$ | The value function for a user goal and environment state |
| $Q_g(x,u,a) = Q^*(s,u,a)$ | The action-value function for a user goal and environment state |
| $b$ | Belief, or distribution over states in our POMDP. |
| $\tau(b' \mid b,u,a)$ | Transition function of belief state |
| $V^{\pi^{\mathrm{r}}}(b)$ | Value function for following policy $\pi^{\mathrm{r}}$ given belief $b$ |
| $Q^{\pi^{\mathrm{r}}}(b,u,a)$ | Action-value for taking actions $u$ and $a$ and following $\pi^{\mathrm{r}}$ thereafter |
| $V^{\mathrm{HS}}(b)$ | Value given by hindsight optimization approximation |
| $Q^{\mathrm{HS}}(b,u,a)$ | Action-value given by hindsight optimization approximation |

We assume that all uncertainty is over the user's goal, and the environment state is known. This subclass of POMDPs, where uncertainty is constant, has been studied as a hidden goal MDP (Fern and Tadepalli, 2010) and as a POMDP-lite (Chen et al., 2016).

In this framework, we infer a distribution of the user's goal by observing the user actions $u$. Similar to the known-goal setting (Section 3.1), we define the value function of a belief as

$$\begin{aligned} V^{\pi^{\mathrm{r}}}(b) &= \mathbb{E}\left[\sum_t C^{\mathrm{r}}(s_t,u_t,a_t) \mid b_0 = b\right] \\ s_t &\sim b_t \\ u_t &\sim \pi^{\mathrm{u}}(\cdot \mid s_t) \\ a_t &\sim \pi^{\mathrm{r}}(\cdot \mid s_t,u_t) \\ b_{t+1} &\sim \tau(\cdot \mid b_t,u_t,a_t) \end{aligned}$$

where the belief transition $\tau$ corresponds to transitioning the known environment state $x$ according to $T$, and updating our belief over the user's goal as described in Section 3.4. We can define quantities similar to the above over beliefs:

$$\begin{aligned} V^*(b) &= \min_{\pi^{\mathrm{r}}} V^{\pi^{\mathrm{r}}}(b) \\ Q^*(b,u,a) &= \mathbb{E}\left[C^{\mathrm{r}}(b,u,a) + \mathbb{E}_{b'}\left[V^*(b')\right]\right] \end{aligned} \tag{1}$$

## 3.3. Hindsight optimization

Computing the optimal solution for a POMDP with continuous states and actions is generally intractable. Instead, we approximate this quantity through hindsight optimization (Chong et al., 2000; Yoon et al., 2008) or QMDP (Littman et al., 1995). This approximation estimates the value function by switching the order of the min and expectation in Equation (1):

$$\begin{aligned} V^{\mathrm{HS}}(b) &= \mathbb{E}_b\left[\min_{\pi^{\mathrm{r}}} V^{\pi^{\mathrm{r}}}(s)\right] \\ &= \mathbb{E}_g\left[V_g(x)\right] \\ Q^{\mathrm{HS}}(b,u,a) &= \mathbb{E}_b\left[C^{\mathrm{r}}(s,u,a) + \mathbb{E}_{s'}\left[V^{\mathrm{HS}}(s')\right]\right] \\ &= \mathbb{E}_g\left[Q_g(x,u,a)\right] \end{aligned}$$

where we explicitly take the expectation over $g \in G$, as we assume that is the only uncertain part of the state.

Conceptually, this approximation corresponds to assuming that all uncertainty will be resolved at the next timestep, and computing the optimal cost-to-go. As this is the best-case scenario for our uncertainty, this is a lower bound of the cost-to-go, $V^{\mathrm{HS}}(b) \leq V^*(b)$. Hindsight optimization has demonstrated effectiveness in other domains (Yoon et al., 2007, 2008). However, as it assumes uncertainty will be resolved, it never explicitly gathers information (Littman et al., 1995), and thus performs poorly when this is necessary.

We believe this method is suitable for shared autonomy for many reasons. Conceptually, we assume the user provides inputs at all times, and therefore we gain information without explicit information gathering. Works in other domains with similar properties have shown that this approximation performs comparably to methods that consider explicit information gathering (Koval et al., 2014). Computationally, computing $Q^{\mathrm{HS}}$ can be done with continuous state and action spaces, enabling fast reaction to user inputs.

Computing $Q_g$ for shared autonomy requires utilizing the user policy $\pi^{\mathrm{u}}_g$, which can make computation difficult. This can be alleviated with the following approximations.

*Stochastic user with robot.* Estimate $u$ using $\pi_g^{\mathrm{u}}$ at each time step, e.g. by sampling, and utilize the full cost function $C_g^{\mathrm{r}}(x,u,a)$ and transition function $T(x' \mid x,u,a)$ to compute $Q_g$. This would be the standard QMDP approach for our POMDP.

*Deterministic user with robot.* Estimate $u$ as the most likely $u$ from $\pi_g^{\mathrm{u}}$ at each time step, and utilize the full cost function $C_g^{\mathrm{r}}(x,u,a)$ and transition function $T(x' \mid x,u,a)$ to compute $Q_g$. This uses our policy predictor, as above, but does so deterministically, and is thus more computationally efficient.

*Robot takes over.* Assume the user will stop supplying inputs, and the robot will complete the task. This enables us to use the cost function $C_g^{\mathrm{r}}(x,0,a)$ and transition function $T(x' \mid x,0,a)$ to compute $Q_g$. For many cost functions, we can analytically compute this value, e.g. cost of always moving towards the goal at some velocity. An additional benefit of this method is that it makes no assumptions about the user policy $\pi_g^{\mathrm{u}}$, making it more robust to modeling errors. We use this method in our experiments.

Finally, it is often difficult to compute $\arg\max_a Q^{\mathrm{HS}}(b,u,a)$ directly, as this requires optimizing over a continuous space of actions. Instead, we use a first-order approximation, which leads to following the gradient of $Q^{\mathrm{HS}}(b,u,a)$.

### 3.4. User prediction

To infer the user's goal, we rely on a model $\pi_g^{\mathrm{u}}$ to provide the distribution of user actions at state $x$ for user goal $g$. In principle, we could use any generative predictor for this model (e.g. Koppula and Saxena, 2013; Wang et al., 2013). We choose to use MaxEnt IOC (Ziebart et al., 2008), as it explicitly models a user cost function $C_g^{\mathrm{u}}$. We optimize this directly by defining $C_g^{\mathrm{r}}$ as a function of $C_g^{\mathrm{u}}$.

In this work, we assume the user does not model robot actions. We use this assumption to define an MDP with states $x \in X$ and user actions $u \in U$ as before, transition $T^{\mathrm{u}}(x' \mid x,u) = T(x' \mid x,u,0)$, and cost $C_g^{\mathrm{u}}(x,u)$. MaxEnt IOC computes a stochastically optimal policy for this MDP.

The distribution of actions at a single state are computed based on how optimal that action is for minimizing cost over a horizon $T$. Define a sequence of environment states and user inputs as $\xi = \{x_0, u_0, \ldots, x_T, u_T\}$. Note that sequences are not required to be trajectories, in that $x_{t+1}$ is not necessarily the result of applying $u_t$ in state $x_t$. Define the cost of a sequence as the sum of costs of all state–input pairs, $C_g^{\mathrm{u}}(\xi) = \sum_t C_g^{\mathrm{u}}(x_t,u_t)$. Let $\xi^{0\to t}$ be a sequence from time 0 to $t$, and $\xi_x^{t\to T}$ a sequence of from time $t$ to $T$, starting at $x$.

Ziebart (2010) showed that minimizing the worst-case predictive loss results in a model where the probability of a sequence decreases exponentially with cost, $p(\xi \mid g) \propto \exp(-C_g^{\mathrm{u}}(\xi))$. Importantly, one can efficiently learn a cost function consistent with this model from demonstrations (Ziebart et al., 2008).

Computationally, the difficulty in computing $p(\xi \mid g)$ lies in the normalizing constant $\int_\xi \exp(-C_g^{\mathrm{u}}(\xi))$, known as the partition function. Evaluating this explicitly would require enumerating all sequences and calculating their cost. However, as the cost of a sequence is the sum of costs of all state–action pairs, dynamic programming can be utilized to compute this through soft-minimum value iteration when the state is discrete (Ziebart et al., 2009, 2012):

$$Q_{g,t}^{\approx}(x,u) = C_g^{\mathrm{u}}(x,u) + \mathbb{E}\left[V_{g,t+1}^{\approx}(x')\right]$$
$$V_{g,t}^{\approx}(x) = \operatorname*{softmin}_u Q_{g,t}^{\approx}(x,u)$$

where $\operatorname{softmin}_x f(x) = -\log \int_x \exp(-f(x))\,dx$ and $x' \sim T^{\mathrm{u}}(\cdot \mid x,u)$.

The log partition function is given by the soft value function, $V_{g,t}^{\approx}(x) = -\log \int_{\xi_x^{t\to T}} \exp(-C_g^{\mathrm{u}}(\xi_x^{t\to T}))$, where the integral is over all sequences starting at $x$ and time $t$. Furthermore, the probability of a single input at a given environment state is given by $\pi_t^{\mathrm{u}}(u \mid x,g) = \exp V_{g,t}^{\approx}(x) - Q_{g,t}^{\approx}(x,u))$ (Ziebart et al., 2009).

Many works derive a simplification that enables them to only look at the start and current states, ignoring the inputs in between (Dragan and Srinivasa, 2013b; Ziebart et al., 2012). Key to this assumption is that $\xi$ corresponds to a trajectory, where applying action $u_t$ at $x_t$ results in $x_{t+1}$. However, if the system is providing assistance, this may not be the case. In particular, if the assistance strategy believes the user's goal is $g$, the assistance strategy will select actions to minimize $C_g^{\mathrm{u}}$. Applying these simplifications will result positive feedback, where the robot makes itself more confident about goals it already believes are likely. To avoid this, we ensure that the prediction comes from user inputs only, and not robot actions:

$$p(\xi \mid g) = \prod_t \pi_t^{\mathrm{u}}(u_t \mid x_t, g)$$

To compute the probability of a goal given the partial sequence up to $t$, we apply Bayes' rule:

$$p(g \mid \xi^{0\to t}) = \frac{p(\xi^{0\to t} \mid g)\,p(g)}{\sum_{g'} p(\xi^{0\to t} \mid g')\,p(g')}$$

This corresponds to our POMDP observation model, used to transition our belief over goals through $\tau$.

*3.4.1. Continuous state and action approximation.* Soft-minimum value iteration is able to find the exact partition function when states and actions are discrete. However, it is computationally intractable to apply in continuous state and action spaces. Instead, we follow Dragan and Srinivasa (2013b) and use a second-order approximation about the optimal trajectory. They show that, assuming a constant Hessian, we can replace the difficult to compute soft-min

functions $V_g^{\approx}$ and $Q_g^{\approx}$ with the min value and action-value functions $V_g^{\mathrm{u}}$ and $Q_g^{\mathrm{u}}$:

$$\pi_t^{\mathrm{u}}(u \mid x, g) = \exp(V_g^{\mathrm{u}}(x) - Q_g^{\mathrm{u}}(x, u))$$

Recent works have explored extensions of the MaxEnt IOC model for continuous spaces (Boularias et al., 2011; Finn et al., 2016; Levine and Koltun, 2012). We leave experiments using these methods for learning and prediction as future work.

### 3.5. Multi-target MDP

There are often multiple ways to achieve a goal. We refer to each of these ways as a target. For a single goal (e.g. object to grasp), let the set of targets (e.g. grasp poses) be $\kappa \in K$. We assume each target has a cost function $C_\kappa$, from which we compute the corresponding value and action-value functions $V_\kappa$ and $Q_\kappa$, and soft-value functions $V_\kappa^{\approx}$ and $Q_\kappa^{\approx}$. We derive the quantities for goals, $V_g, Q_g, V_g^{\approx}, Q_g^{\approx}$, as functions of these target functions.

We state the theorems below, and provide proofs in Appendix A.

*3.5.1. Multi-target assistance.* We assign the cost of a state–action pair to be the cost for the target with the minimum cost-to-go after this state:

$$C_g(x, u, a) = C_{\kappa*}(x, u, a) \quad \kappa* = \arg\min_\kappa V_\kappa(x') \tag{2}$$

where $x'$ is the environment state after actions $u$ and $a$ are applied at state $x$. For the following theorem, we require that our user policy be deterministic, which we already assume in our approximations when computing robot actions in Section 3.3.

**Theorem 1.** *Let $V_\kappa$ be the value function for target $\kappa$. Define the cost for the goal as in Equation* (2)*. For an MDP with deterministic transitions, and a deterministic user policy $\pi^u$, the value and action-value functions $V_g$ and $Q_g$ can be computed as*

$$Q_g(x, u, a) = Q_{\kappa*}(x, u, a) \qquad \kappa^* = \arg\min_\kappa V_\kappa(x')$$
$$V_g(x) = \min_\kappa V_\kappa(x)$$

*3.5.2. Multi-target prediction.* Here, we do not assign the goal cost to be the cost of a single target $C_\kappa$, but instead use a distribution over targets.

**Theorem 2.** *Define the probability of a trajectory and target as $p(\xi, \kappa) \propto \exp(-C_\kappa(\xi))$. Let $V_\kappa^{\approx}$ and $Q_\kappa^{\approx}$ be the soft-value functions for target $\kappa$. For an MDP with deterministic transitions, the soft value functions for goal g, $V_g^{\approx}$ and $Q_g^{\approx}$, can be computed as*

$$V_g^{\approx}(x) = \mathit{soft}\min_\kappa V_\kappa^{\approx}(x)$$
$$Q_g^{\approx}(x, u) = \mathit{soft}\min_\kappa Q_\kappa^{\approx}(x, u)$$

## 4. Shared control teleoperation

We apply our shared autonomy framework to two shared control teleoperation tasks: a simpler task of object grasping (Section 4.1) and a more complicated task of feeding (Section 4.2). Formally, the state $x$ corresponds to the end-effector pose of the robot, each goal $g$ an object in the world, and each target $\kappa$ a pose for achieving that goal (e.g. pre-grasp pose). The transition function $T(x' \mid x, u, a)$ deterministically transitions the state by applying both $u$ and $a$ as end-effector velocities. We map user joystick inputs to $u$ as if the user were controlling the robot through direct teleoperation.

For both tasks, we hand-specify a simple user cost function, $C_\kappa^{\mathrm{u}}$, from which everything is derived. Let $d$ be the distance between the robot state $x' = T^{\mathrm{u}}(x, u)$ and target $\kappa$:[3]

$$C_\kappa^{\mathrm{u}}(x, u) = \begin{cases} \alpha & d > \delta \\ \frac{\alpha}{\delta} d & d \leq \delta \end{cases}$$

That is, a linear cost near a target ($d \leq \delta$) and a constant cost otherwise. This is based on our observation that users make fast, constant progress towards their goal when far away, and slow down for alignment when near their goal. This is by no means the best cost function, but it does provide a baseline for performance. We might expect, for example, that incorporating collision avoidance into our cost function may enable better performance (You and Hauser, 2011). We use this cost function, as it enables closed-form value function computation, enabling inference and execution at 50 Hz.

For prediction, when the distance is far away from any target ($d > \delta$), our algorithm shifts probability towards goals relative to how much progress the user action makes towards the target. If the user stays close to a particular target ($d \leq \delta$), probability mass automatically shifts to that goal, as the cost for that goal is less than all others.

We set $C_\kappa^{\mathrm{r}}(x, a, u) = C_\kappa^{\mathrm{u}}(x, a)$, causing the robot to optimize for the user cost function directly,[4] and behave similar to how we observe users behaved. When far away from goals ($d > \delta$), it makes progress towards all goals in proportion to their probability of being the user's goal. When near a target ($d \leq \delta$) that has high probability, our system reduces assistance as it approaches the final target pose, letting users adjust the final pose if they wish.

We believe hindsight optimization is a suitable POMDP approximation for shared control teleoperation. A key requirement for shared control teleoperation is efficient computation, to make the system feel responsive. With hindsight optimization, we can provide assistance at 50 Hz, even with continuous state and action spaces.

The primary drawback of hindsight optimization is the lack of explicit information gathering (Littman et al., 1995): it assumes all information is revealed at the next timestep, negating any benefit to information gathering. As we assume the user provides inputs at all times, we gain

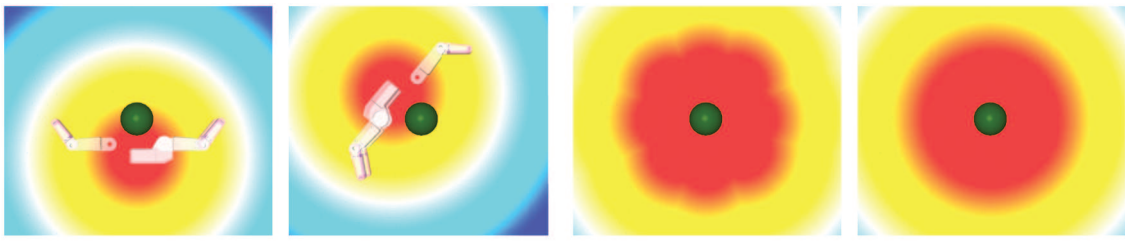



**Fig. 7.** Value function for a goal (grasp the ball) decomposed into value functions of targets (grasp poses). (a, b) Two targets and their corresponding value function $V_\kappa$. In this example, there are 16 targets for the goal. (c) The value function of a goal $V_g$ used for assistance, corresponding to the minimum of all 16 target value functions (d) The soft-min value function $V_g^{\approx}$ used for prediction, corresponding to the soft-min of all 16 target value functions.

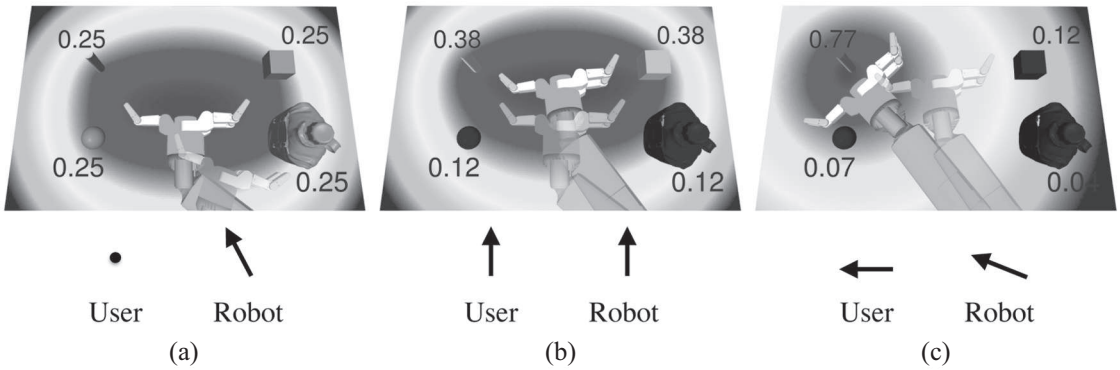



**Fig. 8.** Estimated goal probabilities and value function for object grasping. Top row: the probability of each goal object and a two-dimensional slice of the estimated value function. The transparent end-effector corresponds to the initial state, and the opaque end-effector to the next state. Bottom row: the user input and robot control vectors that caused this motion. (a) Without user input, the robot automatically goes to the position with lowest value, whereas estimated probabilities and value function are unchanged. (b) As the user inputs "forward," the end-effector moves forward, the probability of goals in that direction increase, and the estimated value function shifts in that direction. (c) As the user inputs "left," the goal probabilities and value function shift in that direction. Note that as the probability of one object dominates the others, the system automatically rotates the end-effector for grasping that object.

information automatically when it matters. When the optimal action is the same for multiple goals, we take that action. When the optimal action differs, our model gains information proportional to how suboptimal the user action is for each goal, shifting probability mass towards the user goal, and providing more assistance to that goal.

For shared control teleoperation, explicit information gathering would move the user to a location where their actions between goals were maximally different. Prior works suggest that treating users as an oracle is frustrating (Amershi et al., 2014; Guillory and Bilmes, 2011), and this method naturally avoids it.

We evaluated this system in two experiments, comparing our POMDP-based method, referred to as policy, with a conventional predict-then-act approach based on Dragan and Srinivasa (2013b), referred to as blend (Figure 5). In our feeding experiment, we additionally compare to direct teleoperation, referred to as direct, and full autonomy, referred to as autonomy.

The blend baseline of Dragan and Srinivasa (2013b) requires estimating the predictor's confidence of the most probable goals, which controls how user action and autonomous assistance are arbitrated (Figure 5). We use the distance-based measure used in the experiments of Dragan and Srinivasa (2013b), $\text{conf} = \max\left(0, 1 - \frac{d}{D}\right)$, where $d$ is the distance to the nearest target, and $D$ is some threshold past which confidence is zero.

## 4.1. Grasping experiment

Our first shared-control teleoperation user study evaluates two methods, our POMDP framework and a predict-then-act blending method (Dragan and Srinivasa, 2013b), on the task of object grasping. This task appears broadly in teleoperation systems, appearing in nearly all applications of teleoperated robotic arms. In addition, we chose this task for its simplicity, evaluating these methods on tasks where direct teleoperation is relatively easy.

*4.1.1. Metrics.* Our experiment aims to evaluate the efficiency and user satisfaction of each method.

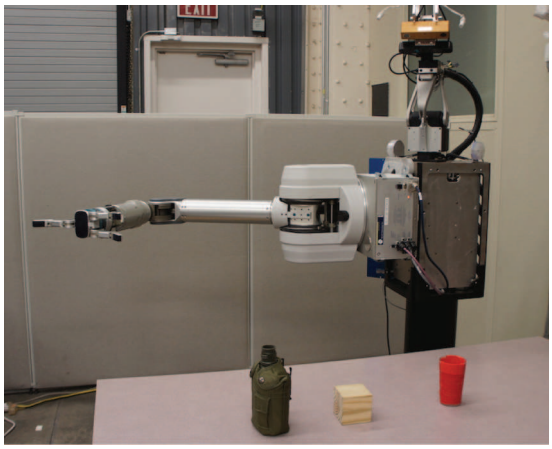


**Fig. 9.** Our experimental setup for object grasping. Three objects, a canteen, a block, and a glass, were placed on the table in front of the robot in a random order. Prior to each trial, the robot moved to the configuration shown. Users picked up each object using each teleoperation system.

**Objective measures.** We measure the objective efficiency of the system in two ways. Total execution time measures how long it took the participant to grasp an object, measuring the effectiveness in achieving the user's goal. Total joystick input measures the magnitude of joystick movement during each trial, measuring the user's effort to achieve their goal.

**Subjective measures.** We also evaluated user satisfaction with the system through a seven-point Likert scale survey. After using each control method, we asked users to rate whether they would like to use the method. After using both methods, we asked users which they preferred.

*4.1.2. Hypotheses.* Prior work suggests that more autonomy leads to greater efficiency for teleoperated robots (Dragan and Srinivasa, 2013b; Hauser, 2013; Javdani et al., 2015; Leeper et al., 2012; You and Hauser, 2011). In addition, prior work indicates that users subjectively prefer more assistance when it leads to more efficient task completion (Dragan and Srinivasa, 2013b; You and Hauser, 2011). Based on this, we formulate the following hypotheses.

**H1a** *Participants using the policy method will grasp objects significantly faster than the blend method*

**H1b** *Participants using the policy method will grasp objects with significantly less control input than the blend method*

**H1c** *Participants will agree more strongly on their preferences for the policy method compared with the blend method*

*4.1.3. Experiment design.* We set up our experiments with three objects on a table: a canteen, a block, and a cup (Figure 9). Users teleoperated a robot arm using two joysticks on a Razer Hydra system. The right joystick mapped to the horizontal plane and the left joystick mapped to the height. A button on the right joystick closed the hand. Each trial consisted of moving from the fixed start pose, shown in Figure 9, to the target object, and ended once the hand was closed.

*4.1.4. Procedure.* We conducted a within-subjects study with one independent variable (control method) that had two conditions (policy, blend). We counteract the effects of novelty and practice by counterbalancing the order of conditions. Each participant grasped each object one time for each condition for a total of six trials.

We recruited 10 participants (nine men, one woman), all with experience in robotics, but none with prior exposure to our system. To counterbalance individual differences of users, we chose a within-subjects design, where each user used both systems.

Users were told they would be using two different teleoperation systems, referred to as "method1" and "method2." Users were not provided any information about the methods. Prior to the recorded trials, users went through a training procedure. First, they teleoperated the robot directly, without any assistance or objects in the scene. Second, they grasped each object one time with each system, repeating if they failed the grasp. Users were then given the option of additional training trials for either system if they wished.

Users then proceeded to the recorded trials. For each system, users picked up each object one time in a random order. Users were told they would complete all trials for one system before the system switched, but were not told the order. However, it was obvious immediately after the first trial started, as the policy method assists from the start pose and blend does not. Upon completing all trials for one system, they were told the system would be switching, and then proceeded to complete all trials for the other system. If users failed at grasping (e.g. they knocked the object over), the data was discarded and they repeated that trial. Execution time and total user input were measured for each trial.

Upon completing all trials, users were given a short survey. For each system, they were asked for their agreement on a 1–7 Likert scale for the following statements:

1. "I felt in *control*;"
2. "The robot did what I *wanted*;"
3. "I was able to accomplish the tasks *quickly*;"
4. "If I was going to teleoperate a robotic arm, I would *like* to use the system."

They were also asked "which system do you *prefer*", where 1 corresponded to blend, 7 to policy, and 4 to neutral. Finally, they were asked to explain their choices and provide any general comments.

*4.1.5. Results.* Users were able to use both systems successfully. There were a total of two failures while using each system: once each because the user attempted to grasp too

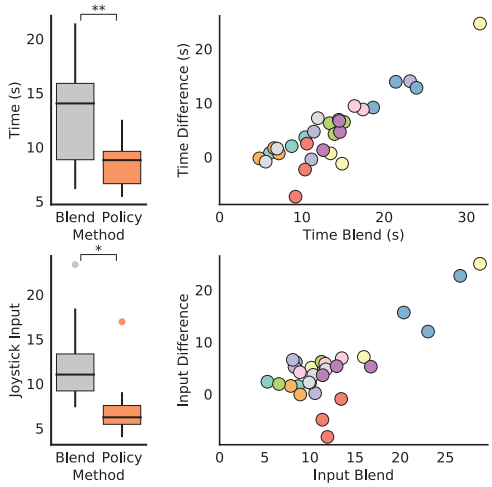



**Fig. 10.** Task completion times and total input for all trials. On the left, box plots for each system. On the right, the time and input of blend minus policy, as a function of the time and total input of blend. Each point corresponds to one trial, and colors correspond to different users. We see that policy was faster ($p < 0.01$) and resulted in less input ($p < 0.05$). In addition, the difference between systems increases with the time/input of blend.

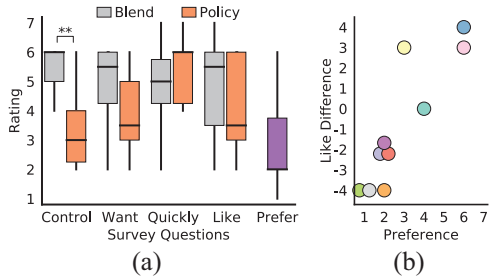



**Fig. 11.** (a) Boxplots of the survey results from our user study. For each system, users were asked whether they felt in *control*, whether the robot did what they *wanted*, whether they were able to accomplish tasks *quickly*, and whether they would *like* to use the system. In addition, they were asked which system they *prefer*, where a rating of 1 corresponds to blend and 7 corresponds to policy. We found that users agreed with feeling in control more when using the blend method compared with the policy method ($p < 0.01$). (b) The *like* rating of policy minus blend, plotted against the *prefer* rating. When multiple users mapped to the same coordinate, we plot multiple dots around that coordinate. Colors correspond to different users, where the same user has the same color in Figure 10.

early, and once each because the user knocked the object over. These experiments were reset and repeated.

We assess our hypotheses using a significance level of $\alpha = 0.05$. For data that violated the assumption of sphericity, we used a Greenhouse–Geisser correction. If a significant main effect was found, a post-hoc analysis was used to identify which conditions were statistically different from each other, with Holm–Bonferroni corrections for multiple comparisons.

Trial times and total control input were assessed using a two-factor repeated measures analysis of variance (ANOVA), using the assistance method and object grasped as factors. Both trial times and total control input had a significant main effect. We found that our policy method resulted in users accomplishing tasks more quickly, supporting **H1a** ($F(1,9) = 12.98, p = 0.006$). Similarly, our policy method resulted in users grasping objects with less input, supporting **H1b** ($F(1,9) = 7.76, p = 0.021$). See Figure 10 for more detailed results.

To assess user preference, we performed a Wilcoxon paired signed-rank test on our survey question asking whether they would *like* to use each system, and a Wilcoxon rank-sum test on the survey question of which system they *prefer* against the null hypothesis of no preference (value of 4). There was no evidence to support **H1c**.

In fact, our data suggests a trend towards the opposite: that users prefer blend over policy. When asked whether they would *like* to use the system, there was a small difference between methods (blend: $M = 4.90$, $SD = 1.58$; policy: $M = 4.10$, $SD = 1.64$). However, when asked which system they *preferred*, users expressed a stronger preference for blend ($M = 2.90$, $SD = 1.76$). Although these results are not statistically significant according to our Wilcoxon tests and $\alpha = 0.05$, it does suggest a trend towards preferring blend. See Figure 11 for results for all survey questions.

We found this surprising, as prior work indicates a strong correlation between task completion time and user satisfaction, even at the cost of control authority, in both shared autonomy (Dragan and Srinivasa, 2013b; Hauser, 2013) and human–robot teaming (Gombolay et al., 2014) settings.[5] Not only were users faster, but they recognized they could accomplish tasks more quickly (see *quickly* in Figure 11). One user specifically commented that "[Policy] took more practice to learn…but once I learned I was able to do things a little faster. However, I still don't like feeling it has a mind of its own."

Users agreed more strongly that they felt in *control* during blend ($Z = -2.687$, $p = 0.007$). Interestingly, when asked whether the robot did what they *wanted*, the difference between methods was less drastic. This suggests that for some users, the robot's autonomous actions were in-line with their desired motions, even though the user did not feel that they were in control.

Users also commented that they had to compensate for policy in their inputs. For example, one user stated that "[policy] did things that I was not expecting and resulted in unplanned motion." This can perhaps be alleviated with user-specific policies, matching the behavior of particular users.

Some users suggested their preferences may change with better understanding. For example, one user stated they "disliked (policy) at first, but began to prefer it slightly after learning its behavior. Perhaps I would prefer it more strongly with more experience." It is possible that with

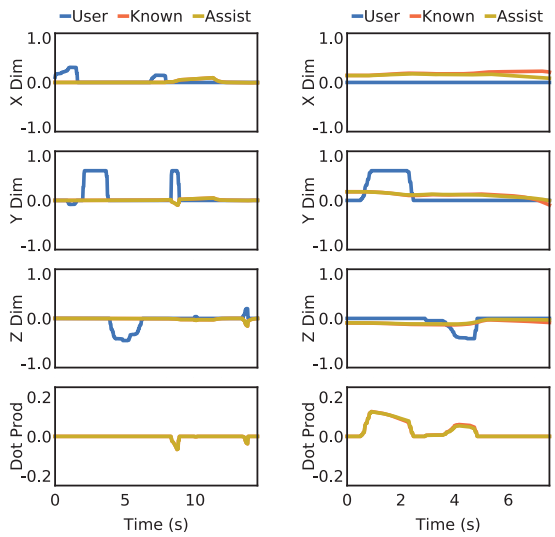



**Fig. 12.** User input and autonomous actions for a user who preferred policy assistance, using (a) blending and (b) policy for grasping the same object. We plot the user input, autonomous assistance with the estimated distribution, and what the autonomous assistance would have been had the predictor known the true goal. We subtract the user input from the assistance when plotting, to show the autonomous action as compared with direct teleoperation. The top three figures show each dimension separately. The bottom shows the dot product between the user input and assistance action. This user changed their strategy during policy assistance, letting the robot do the bulk of the work, and only applying enough input to correct the robot for their goal. Note that this user never applied input in the "$X$" dimension in this or any of their three policy trials, as the assistance always went towards all objects in that dimension.

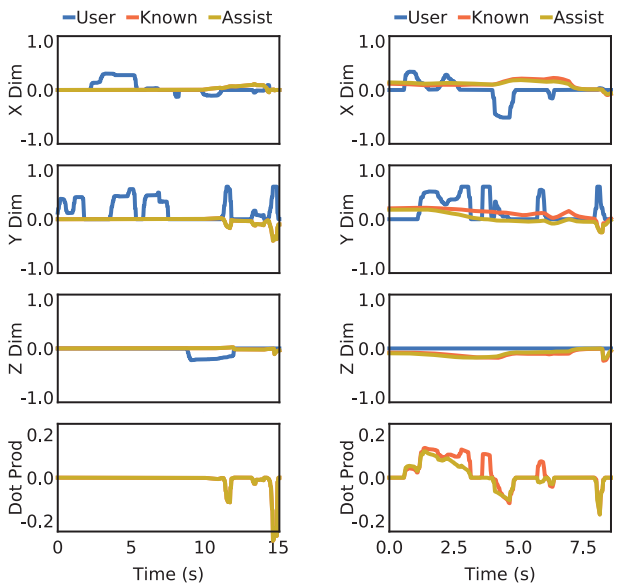



**Fig. 13.** User input and autonomous assistance for a user who preferred blending, with plots as in Figure 12. The user inputs sometimes opposed the autonomous assistance (such as in the "$X$" dimension) for both the estimated distribution and known goal, suggesting the cost function did not accomplish the task in the way the user wanted. However, the user was still able to accomplish the task faster with the autonomous assistance than blending.

more training, or an explanation of how policy works, users would have preferred the policy method. We leave this for future work.

*4.1.6. Examining trajectories.* Users with different preferences had very different strategies for using each system. Some users who preferred the assistance policy changed their strategy to take advantage of the constant assistance towards all goals, applying minimal input to guide the robot to the correct goal (Figure 12). In contrast, users who preferred blending were often opposing the actions of the autonomous policy (Figure 13). This suggests the robot was following a strategy different from their own.

## 4.2. Feeding experiment

Building on the results of the grasping study (Section 4.1), we designed a broader evaluation of our system. In this evaluation, we test our system in an eating task using a Kinova Mico robot manipulator. We chose the Mico robot because it is a commercially available assistive device, and thus provides a realistic testbed for assistive applications. We selected the task of eating for two reasons. First, eating independently is a real need; it has been identified as one of the most important tasks for assistive robotic arms (Chung et al., 2013). Second, eating independently is hard; interviews with current users of assistive arms have found that people generally do not attempt to use their robot arm for eating, as it requires too much effort (Herlant et al., 2016). By evaluating our systems on the desirable but difficult task of eating, we show how shared autonomy can improve over traditional methods for controlling an assistive robot in a real-world domain that has implications for people's quality of life.

We also extended our evaluation by considering two additional control methods: direct teleoperation and full robot autonomy. Direct teleoperation is how assistive robot manipulators such as the Mico are currently operated by users. Full autonomy represents a condition in which the robot is behaving "optimally" for its own goal, but does not take the user's goal into account.

Thus, in this evaluation, we conducted a user study to evaluate four methods of robot control, our POMDP framework, a predict-then-act blending method (Dragan and Srinivasa, 2013b), direct teleoperation, and full autonomy, in an assistive eating task.

*4.2.1. Metrics.* Our experiments aim to evaluate the effectiveness and user satisfaction of each method.

**Objective measures.** We measure the objective efficiency of the system in four ways. Success rate identifies the proportion of successfully completed trials, where success is determined by whether the user was able to pick up their intended piece of food. Total execution time measures how long it took the participant to retrieve the food in each trial. Number of mode switches identifies how many times participants had to switch control modes during the trial (Figure 4). Total joystick input measures the magnitude of joystick movement during each trial. The first two measures evaluate how effectively the participant could reach their goal, whereas the last two measures evaluate how much effort it took them to do so.

**Subjective measures.** We also evaluated user satisfaction with the system through subjective measures. After five trials with each control method, we asked users to respond to questions about each system using a seven-point Likert scale. These questions, specified in Section 4.2.4, assessed user preferences, their perceived ability to achieve their goal, and feeling they were in control. In addition, after they saw all of the methods, we asked users to rank order the methods according to their preference.

*4.2.2. Hypotheses.* As in the previous evaluation, we are motivated by prior work that suggests that more autonomy leads to greater efficiency and accuracy for teleoperated robots (Dragan and Srinivasa, 2013b; Hauser, 2013; Javdani et al., 2015; Leeper et al., 2012; You and Hauser, 2011). We formulate the following hypotheses regarding the efficiency of our control methods, measured through objective metrics.

**H2a** *Using methods with more autonomous assistance will lead to more successful task completions*

**H2b** *Using methods with more autonomous assistance will result in faster task completion*

**H2c** *Using methods with more autonomous assistance will lead to fewer mode switches*

**H2d** *Using methods with more autonomous assistance will lead to less joystick input*

Feeding with an assistive arm is difficult (Herlant et al., 2016), and prior work indicates that users subjectively prefer more assistance when the task is difficult even though they have less control (Dragan and Srinivasa, 2013b; You and Hauser, 2011). Based on this, we formulate the following hypotheses regarding user preferences, measured through our subjective metrics:

**H2e** *Participants will more strongly agree on feeling in control for methods with less autonomous assistance*

**H2f** *Participants will more strongly agree preference and usability subjective measures for methods with more autonomous assistance*

**H2g** *Participants will rank methods with more autonomous assistance above methods with less autonomous assistance*

Our hypotheses depend on an ordering of "more" or "less" autonomous assistance. The four control methods in this study naturally fall into the following ordering (from least to most assistance): direct teleoperation, blending, policy, and full autonomy. Between the two shared autonomy methods, policy provides more assistance because it creates assistive robot behavior over the entire duration of the trajectory, whereas blend must wait until the intent prediction confidence exceeds some threshold before it produces an assistive robot motion.

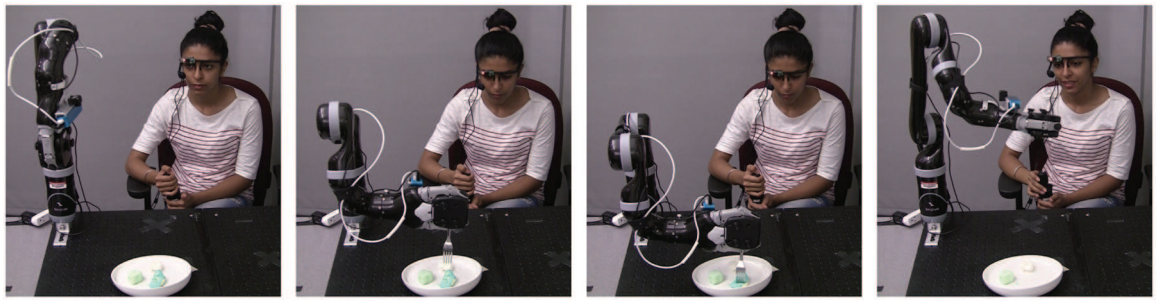
*4.2.3. Experimental design.* To evaluate each robot control algorithm on a realistic assistive task, participants tried to spear bites of food from a plate onto a fork held in the robot's end effector (Figure 14). For each trial, participants controlled the robot through a joystick and attempted to retrieve one of three bites of food on a plate.

Each trial followed a fixed bite retrieval sequence. First, the robot would move to a pose where its wrist-mounted camera could detect bites of food on the plate. This step ensured that the system was robust to bite locations and could operate no matter where on the plate the bites were located. While the camera captured and processed the scene to identify bite locations, we asked users to verbally specify which bite they wanted to retrieve,[6] which allowed us to identify whether people were able to successfully retrieve their target bite.

Next, participants used the joystick to position the robot's end effector so that the fork was directly above their target bite. Six-DOF control was available in three modes of two DOFs each (Figure 4), and participants could switch between modes by pressing a button on the joystick.

Once they had the fork positioned above their target bite, the participant prompted the robot to retrieve the bite by pressing and holding the mode switch button. The robot would then automatically move straight down to the height of the table, spearing the bite on the fork. Finally, the robot automatically served the bite.

*4.2.4. Procedure.* We conducted a within-subjects study with one independent variable (control method) that had four conditions (full teleoperation, blend, policy, and full autonomy). Because each participant saw all control methods, we counteract the effects of novelty and practice by fully counterbalancing the order of conditions. Each participant completed five trials for each condition for a total of 20 trials. The bite retrieval sequence described in Section 4.2.3 was the same in each trial across the four control conditions. The only difference between trials was the control method used for the alignment step, where the fork is positioned above the bite. We measure the metrics discussed in Section 4.2.2 only during this step.



(a) Detect (b) Align (c) Acquire (d) Serve

**Fig. 14.** Our eating study. A plate with three bites of food was placed in front of users. (a) The robot start by detecting the pose of all bites of food. (b) The user then uses one of the four methods to align the fork with their desired bite. When the user indicates they are aligned, the robot automatically (c) acquires and (d) serves the bite.

We recruited 23 able-bodied participants from the local community (11 male, 12 female, ages 19–59). After obtaining written consent, participants were given a brief overview of the feeding task, and told the robot may provide help or take over completely. Users then received instruction for teleoperating the system with modal control, and were given 5 minutes to practice using the robot under direct teleoperation. An eye-tracking system was then placed on users for future data analysis, but participant gaze had no effect on the assistance provided by the robot.

As described in Section 4.2.3, participants used a joystick to spear a piece of food from a plate on a fork held in the robot's end effector. The different control methods were never explained or identified to users, and were simply referred to by their order of presentation (e.g. "method 1," "method 2," etc.). After using each method, users were given a short questionnaire pertaining to that specific method. The questions were:

1. "I felt in *control*;"
2. "The robot did what I *wanted*;"
3. "I was able to accomplish the tasks *quickly*;"
4. "My *goals* were perceived accurately;"
5. "If I were going to teleoperate a robotic arm, I would *like* to use the system."

These questions are identical to those asked in the previous evaluation (Section 4.1), with the addition of question 4, which focuses specifically on the user's goals. Participants were also provided space to write additional comments. After completing all 20 trials, participants were asked to *rank* all four methods in order of preference and provide final comments.

*4.2.5. Results.* One participant was unable to complete the tasks due to lack of comprehension of instructions, and was excluded from the analysis. One participant did not use the blend method because the robot's finger broke during a previous trial. This user's blend condition and final ranking data were excluded from the analysis, but all other data (which were completed before the finger breakage) were used. Two other participants missed one trial each due to technical issues.

Our metrics are detailed in Section 4.2.1. For each participant, we computed the task success rate for each method. For metrics measured per trial (execution time, number of mode switches, and total joystick input), we averaged the data across all five trials in each condition, enabling us to treat each user as one independent datapoint in our analyses. Differences in our metrics across conditions were analyzed using a repeated measures ANOVA with a significance threshold of $\alpha = 0.05$. For data that violated the assumption of sphericity, we used a Greenhouse–Geisser correction. If a significant main effect was found, a post-hoc analysis was used to identify which conditions were statistically different from each other, with Holm–Bonferroni corrections for multiple comparisons.

**Success Rate** differed significantly between control methods ($F(2.33, 49.00) = 4.57$, $p = 0.011$). Post-hoc analysis revealed that more autonomy resulted in significant differences of task completion between policy and direct ($p = 0.021$), and a significant difference between policy and blend ($p = 0.0498$). All other comparisons were not significant. Surprisingly, we found that policy actually had a higher average task completion ratio than autonomy, though not significantly so. Thus, we found support **H2a** (Figure 15a).

**Total execution time** differed significantly between methods ($F(1.89, 39.73) = 43.55$, $p < 0.001$). Post-hoc analysis revealed that more autonomy resulted in faster task completion: autonomy condition completion times were faster than policy ($p = 0.001$), blend ($p < 0.001$), and direct ($p < 0.001$). There were also significant differences between policy and blend ($p < 0.001$), and policy and direct ($p < 0.001$). The only pair of methods which did not have a significant difference was blend and direct. Thus, we found support for **H2b** (Figure 15b).

**Number of mode switches** differed significantly between methods ($F(2.30, 48.39) = 65.16$, $p < 0.001$). Post-hoc analysis revealed that more autonomy resulted fewer mode

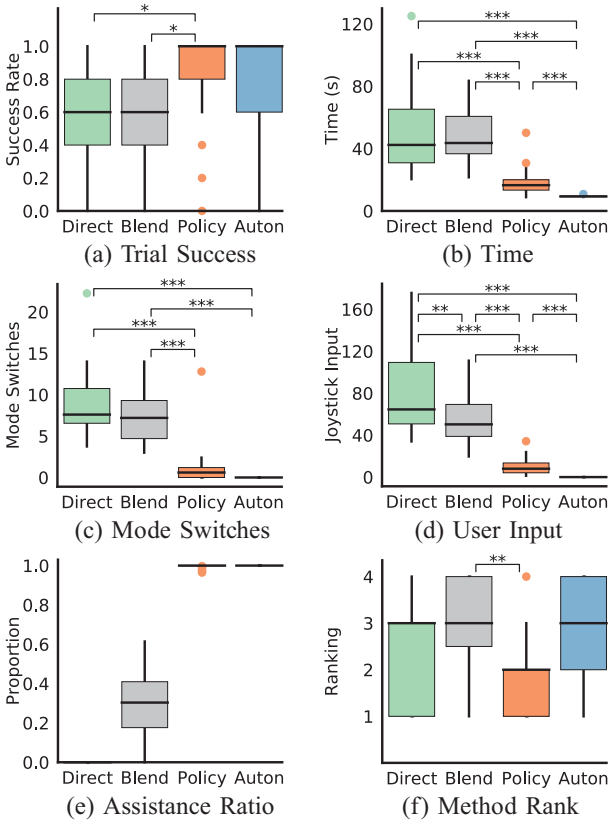



**Fig. 15.** Boxplots for each algorithm across all users of the (a) task completion ratio, (b) total execution time, (c) number of mode switches, (d) total joystick input, (e) the ratio of time that robotic assistance was provided, and (f) the ranking as provided by each user, where 1 corresponds to the most preferred algorithm. Pairs that were found significant during post-analysis are plotted, where $*$ indicates $p < 0.05$, $**$ that $p < 0.01$, and $***$ that $p < 0.001$.

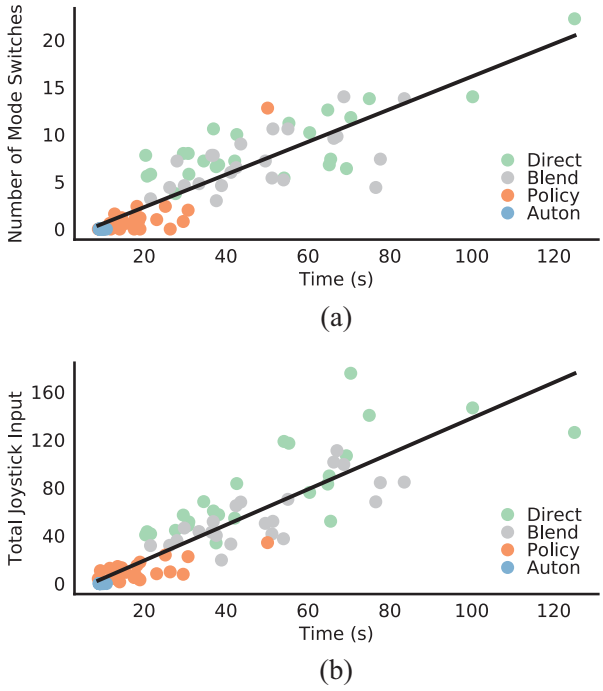



**Fig. 16.** User input versus time for both the number of mode switches (a) and joystick input (b). Each point corresponds to the average for one user for each method. We see a general trend that trials with more time corresponded to more user input. We also fit a line so all points for all methods. Note that the direct teleoperation methods are generally above the line, indicating that shared and full autonomy usually results in less user input even for similar task completion time.

switches between autonomy and blend ($p < 0.001$), autonomy and direct ($p < 0.001$), policy and blend ($p < 0.001$), and policy and direct ($p < 0.001$). Interestingly, there was not a significant difference in the number of mode switches between full autonomy and policy, even though users cannot mode switch when using full autonomy at all. Thus, we found support for **H2c** (Figure 15c).

**Total joystick input** differed significantly between methods ($F(1.67, 35.14) = 65.35$, $p < 0.001$). Post-hoc analysis revealed that more autonomy resulted in less total joystick input between all pairs of methods: autonomy and policy ($p < 0.001$), autonomy and blend ($p < 0.001$), autonomy and direct ($p < 0.001$), policy and blend ($p < 0.001$), policy and direct ($p < 0.001$), and blend and direct ($p = 0.026$). Thus, we found support for **H2d** (Figure 15d).

User-reported subjective measures for the survey questions are assessed using a Friedman's test and a significance threshold of $p = 0.05$. If significance was found, a post-hoc analysis was performed, comparing all pairs with Holm–Bonferroni corrections.

User agreement on **control** differed significantly between methods, $\xi^2(3) = 15.44$, $p < 0.001$, with more autonomy leading to less feeling of control. Post-hoc analysis revealed that all pairs were significant, where autonomy resulting in less feeling of control compared with policy ($p < 0.001$), blend ($p = 0.001$), and direct ($p < 0.001$). Policy resulted in less feeling of control compared with blend ($p < 0.001$) and direct ($p = 0.008$). Blend resulted in less feeling of control compared to direct ($p = 0.002$). Thus, we found support for **H2e**.

User agreement on preference and usability subjective measures sometimes differed significantly between methods. User agreement on **liking** differed significantly between methods, $\xi^2(3) = 8.74$, $p = 0.033$. Post-hoc analysis revealed that between the two shared autonomy methods (policy and blend), users liked the more autonomous method more ($p = 0.012$). User ability for achieving goals **quickly** also differed significantly between methods, $\xi^2 3 = 11.90, p = 0.008$. Post-hoc analysis revealed that users felt they could achieve their goals more quickly with policy than with blend ($p = 0.010$) and direct ($p = 0.043$). We found no significant differences for our other measures. Thus, we find partial support for **H2f** (Figure 18).

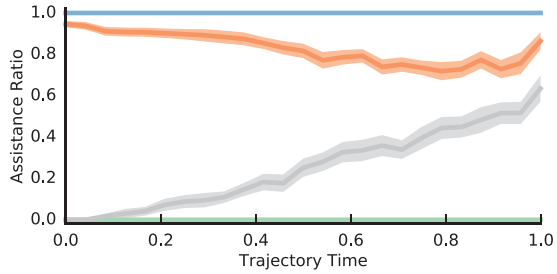



**Fig. 17.** Ratio of the magnitude of the assistance to user input as a function of time. The line shows the mean of the assistance ratio as a function of the proportion of the trajectory. Shaded array plots the standard error over users. We see that blend initially provides no assistance, as the predictor is not confident in the user goal. In contrast, policy provides assistance throughout the trajectory. We also see that policy decreases in assistance ratio over time, as many users provided little input until the system moved and oriented the fork near all objects, at which time they provided input to express their preference and align the fork.

**Ranking** differed significantly between methods, $\xi^2(3) = 10.31, p = 0.016$. Again, post-hoc analysis revealed that between the two shared autonomy methods (policy and blend), users ranked the more autonomous one higher ($p = 0.006$). Thus, we find support for **H2g**. As for the like rating, we also found that on average, users ranked direct teleoperation higher than both blend and full autonomy, though not significantly so (Figure 15f).

*4.2.6. Discussion.* The robot in this study was controlled through a two-DOF joystick and a single button, which is comparable with the assistive robot arms in use today.

As expected, we saw a general trend in which more autonomy resulted in better performance across all objective measures (task completion ratio, execution time, number of mode switches, and total joystick input), supporting **H2a**–**H2d**. We also saw evidence that autonomy decreased feelings of control, supporting **H2e**. However, it improved people's subjective evaluations of usability and preference, particularly between the shared autonomy methods (policy and blend), supporting **H2f** and **H2g**. Most objective measures (particularly total execution time, number of mode switches, and total joystick input) showed significant differences between all or nearly all pairs of methods, whereas the subjective results were less certain, with significant differences between fewer pairs of methods.

We can draw several insights from these findings. First, autonomy improves peoples' performance on a realistic assistive task by requiring less physical effort to control the robot. People use fewer mode switches (which require button presses) and move the joystick less in the more autonomous conditions, but still perform the task more quickly and effectively. For example, in the policy method, 8 of our 22 users did not use any mode switches for any trial, but this method yielded the highest completion ratio and low execution times. Clearly, some robot autonomy can benefit people's experience by reducing the amount of work they have to do.

Interestingly, full autonomy is not always as effective as allowing the user to retain some control. For example, the policy method had a slightly (though not significantly) higher average completion ratio than the full autonomy method. This appears to be the result of users fine-tuning the robot's end-effector position to compensate for small visual or motor inaccuracies in the automatic bite localization process. Because the task of spearing relatively small bites of food requires precise end-effector localization, users' ability to fine-tune the final fork alignment seems to benefit the overall success rate. Though some users were able to achieve it, our policy method is not designed to allow this kind of fine-tuning, and will continually move the robot's end effector back to the erroneous location against the user's control. Detecting when this may be occurring and decreasing assistance would likely enhance people's ability to fine-tune alignment, and improve their task completion rate even further.

Given the success of blending in previous studies (Li et al., 2011; Carlson and Demiris, 2012; Dragan and Srinivasa, 2013b; Muelling et al., 2015; Gopinath et al., 2016), we were surprised by the poor performance of blend in our study. We found no significant difference for blending over direct teleoperation for success rate, task completion time, or number of mode switches. We also saw that it performed the worst among all methods for both user liking and ranking. One possible explanation is that blend spent relatively little time assisting users (Figure 15e). For this task, the goal predictor was unable to confidently predict the user's goal for 69% of execution time, limiting the amount of assistance (Figure 17). Furthermore, the difficult portion of the task, rotating the fork tip to face downward, occurred at the beginning of execution. Thus, as one user put it "While the robot would eventually line up the arm over the plate, most of the hard work was done by me." In contrast, user comments for shared autonomy indicated that "having help earlier with fork orientation was best." This suggests that the *magnitude* of assistance was less important than assisting at a time that would have been helpful. In fact, assisting only during the portion where the user could do well themselves resulted in additional frustration.

Although worse by all objective metrics, participants tended to prefer direct teleoperation over autonomy. This is not entirely surprising, given prior work where users expressed preference for more control (Kim et al., 2012). However, for difficult tasks such as this, users in prior works tend to favor more assistance (Dragan and Srinivasa, 2013b; You and Hauser, 2011). Many users commented that they disliked autonomy due to the lack of item selection, for example, "While [autonomy] was fastest and easiest, it did not account for the marshmallow I wanted." Another user mentioned that autonomy "made me feel inadequate."

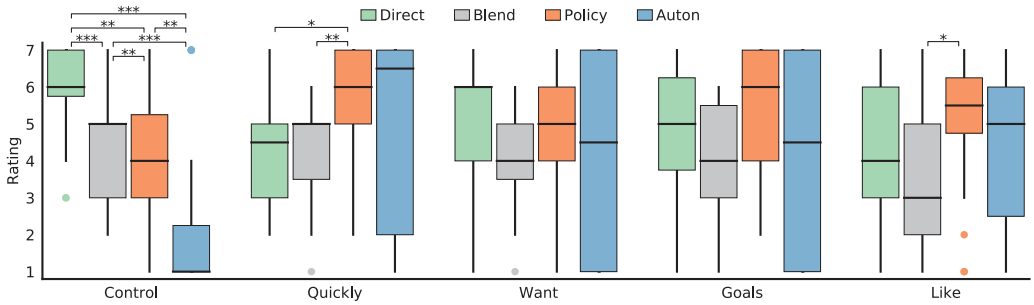



**Fig. 18.** Boxplots for user responses to all survey questions. See Section 4.2.4 for specific questions. Pairs that were found significant during post-hoc analysis are plotted, where $*$ indicates $p < 0.05$, $**$ that $p < 0.01$, and $***$ that $p < 0.001$. We note that policy was perceived as quick, even though autonomy actually had lower task completion (Figure 15b). In addition, autonomy had a very high variance in user responses for many questions, with users very mixed on whether it did what they wanted, and achieved their goal. On average, we see that policy did better than other methods for most user responses.

We also found that users responded to failures by blaming the system, even when using direct teleoperation. Of the eight users who failed to successfully spear a bite during an autonomous trial, five users commented on the failure of the algorithm. In contrast, of the 19 users who had one or more failure during teleoperation, only two commented on their own performance. Instead, users made comments about the system itself, such as how the system "seemed off for some reason" or "did not do what I intended." One user blamed their viewpoint for causing difficulty for the alignment and another the joystick. This suggests that people are more likely to penalize autonomy for its shortcomings than their own control. Interestingly, this was not the case for the shared autonomy methods. We find that when users had some control over the robot's movement, they did not blame the algorithm's failures (for example, mistaken alignments) on the system.

## 5. Human–robot teaming

In human–robot teaming, the user and robot want to achieve a set of related goals. Formally, we assume a set of user goals $g^{\mathrm{u}} \in G^{\mathrm{u}}$ and robot goals $g^{\mathrm{r}} \in G^{\mathrm{r}}$, where both want to achieve all goals. However, there may be constraints on how these goals can be achieved (e.g. user and robot cannot simultaneously use the same object (Hoffman and Breazeal, 2007)). We apply a conservative model for these constraints through a *goal restriction set* $\mathcal{R} = \{(g^{\mathrm{u}}, g^{\mathrm{r}}) : \text{Cannot achieve } g^{\mathrm{u}} \text{ and } g^{\mathrm{r}} \text{ simultaneously}\}$. To efficiently collaborate with the user, our objective is to simultaneously predict the human's intended goal, and achieve a robot goal not in the restricted set. We remove the achieved goals from their corresponding goal sets, and repeat this process until all robot goals are achieved.

The state $x$ corresponds to the state of both the user and robot, where $u$ affects the user portion of state, and $a$ affects the robot portion. The transition function $T(x' \mid x, u, a)$ deterministically transitions the state by applying $u$ and $a$ sequentially.

For prediction, we used the same cost function for $C^{\mathrm{u}}_{\kappa}$ as in our shared teleoperation experiments (Section 4). Let $d$ be the distance between the state[7] $x' = T^{\mathrm{u}}(x, u)$ and target $\kappa$:

$$C^{\mathrm{u}}_{\kappa}(x, u) = \begin{cases} \alpha & d > \delta \\ \frac{\alpha}{\delta} d & d \leq \delta \end{cases}$$

which behaves identically to our shared control teleoperation setting: when the distance is far away from any target ($d > \delta$), probability shifts towards goals relative to how much progress the user makes towards them. When the user stays close to a particular target ($d \leq \delta$), probability mass shifts to that goal, as the cost for that goal is less than all others.

Unlike our shared control teleoperation setting, our robot cost function does not aim to achieve the same goal as the user, but rather any goal not in the restricted set. As in our shared autonomy framework, let $g$ be the user's goal. The cost function for a particular user goal is

$$C^{\mathrm{r}}_{g}(x, u, a) = \min_{g^{\mathrm{r}} \text{ s.t. } (g, g^{\mathrm{r}}) \notin \mathcal{R}} C^{\mathrm{u}}_{g^{\mathrm{r}}}(x, a)$$

where $C^{\mathrm{u}}_{g}$ uses the cost for each target $C^{\mathrm{u}}_{\kappa}$ to compute the cost function as described in Section 3.5. In addition, note that the min over cost functions looks identical to the min over targets to compute the cost for a goal. Thus, for deterministic transition functions, we can use the same proof for computing the value function of a goal given the value function for all targets (Section 3.5.1) to compute the value function for a robot goal given the value function for all user goals:

$$V^{\mathrm{r}}_{g}(x) = \min_{g^{\mathrm{r}} \text{ s.t. } (g, g^{\mathrm{r}}) \notin \mathcal{R}} V^{\mathrm{u}}_{g^{\mathrm{r}}}(x)$$

This simple cost function provides us with a baseline for performance. We might expect better collaboration performance by incorporating costs for collision avoidance with the user (Lasota and Shah, 2015; Mainprice and Berenson, 2013), social acceptability of actions (Sisbot et al., 2007), and user visibility and reachability (Mainprice et al., 2011;

Pandey and Alami, 2010; Sisbot et al., 2010). We use this cost function to test the viability of our framework as it enables closed-form computation of the value function.

This cost and value function causes the robot to go to any goal currently in its goal set $g^r \in G^r$ that is not in the restriction set of the user goal $g$. Under this model, the robot makes progress towards goals that are unlikely to be in the restricted set and have low cost-to-go. As the form of the cost function is identical to that which we used in shared control teleoperation, the robot behaves similarly: making constant progress when far away ( $d > \delta$) and slowing down for alignment when near ( $d \leq \delta$). The robot terminates and completes the task once some condition is met (e.g. $d \leq \epsilon$).

*Hindsight optimization for human–robot teaming.* Similar to shared control teleoperation, we believe hindsight optimization is a suitable POMDP approximation for human–robot teaming. The efficient computation enables us to respond quickly to changing user goals, even with continuous state and action spaces. For our formulation of human–robot teaming, explicit information gathering is not possible: as we assume the user and robot affect different parts of state space, robot actions are unable to explicitly gather information about the user's goal. Instead, we gain information freely from user actions.

## 5.1. Human–robot teaming experiment

We apply our shared autonomy framework to a human–robot teaming task of gift-wrapping, where the user and robot must both perform a task on each box to be gift-wrapped.[8] Our goal restriction set enforces that they cannot perform a task on the same box at the same time.

In a user study, we compare three methods: our shared autonomy framework, referred to as *policy*, a standard predict-then-act system, referred to as *plan*, and a non-adaptive system where the robot executes a fixed sequence of motions, referred to as *fixed*.

*5.1.1. Metrics. Task fluency* involves seamless coordination of action. One measure for task fluency is the minimum distance between the human and robot end effectors during a trial. This was measured automatically by a Kinect mounted on the robot's head, operating at 30 Hz. Our second fluency measure is the proportion of trial time spent in collision. Collisions occur when the distance between the robot's end effector and the human's hand goes below a certain threshold. We determined that 8 cm was a reasonable collision threshold based on observations before beginning the study.

*Task efficiency* relates to the speed with which the task is completed. Objective measures for task efficiency were total task duration for robot and for human, the amount of human idle time during the trial, and the proportion of trial time spent idling. Idling is defined as time a participant spends with their hands still (i.e. not completing the task). For example, idling occurs when the human has to wait for the robot to stamp a box before they can tie the ribbon on it. We only considered idling time while the robot was executing its tasks, so idle behaviors that occurred after the robot was finished stamping the boxes, which could not have been caused by the robot's behavior, were not taken into account.

We also measured subjective *human satisfaction* with each method through a seven-point Likert scale survey evaluating perceived safety (four questions) and sense of collaboration (four questions). The questions were:

1. "HERB was a good partner;"
2. "I think HERB and I worked well as a team;"
3. "I'm dissatisfied with how HERB and I worked together;"
4. "I trust HERB;"
5. "I felt that HERB kept a safe distance from me;"
6. "HERB got in my way;"
7. "HERB moved too fast;"
8. "I felt uncomfortable working so close to HERB."

*5.1.2. Hypotheses.* We hypothesize that:

**H3a** *Task fluency will be improved with our policy method compared with the plan and fixed methods*

**H3b** *Task efficiency will be improved with our policy method compared with the plan and fixed methods*

**H3c** *People will subjectively prefer the policy method to the plan or fixed methods*

*5.1.3. Experimental design.* We developed a gift-wrapping task (Figure 19). A row of four boxes was arranged on a table between the human and the robot; each box had a ribbon underneath it. The robot's task was to stamp the top of each box with a marker it held in its hand. The human's task was to tie a bow from the ribbon around each box. By the nature of the task, the goals had to be selected serially, though ordering was unspecified. Though participants were not instructed explicitly to avoid the robot, tying the bow while the robot was stamping the box was challenging because the robot's hand interfered, which provided a natural disincentive toward selecting the same goal simultaneously.

*5.1.4. Implementation.* We implemented the three control methods on HERB (Srinivasa et al., 2012), a bi-manual mobile manipulator with two Barrett WAM arms. A Kinect was used for skeleton tracking and object detection. Motion planning was performed using CHOMP, except for our policy method in which motion planning works according to Section 3.

The stamping marker was pre-loaded in HERB's hand. A stamping action began at a home position, the robot extended its arm toward a box, stamped the box with the marker, and retracted its arm back to the home position.

To implement the fixed method, the system simply calculated a random ordering of the four boxes, then performed

**Fig. 19.** Participants performed a collaborative gift-wrapping task with HERB to evaluate our POMDP-based reactive system against a state-of-the-art predict-then-act method, and a non-adaptive fixed sequence of robot goals.

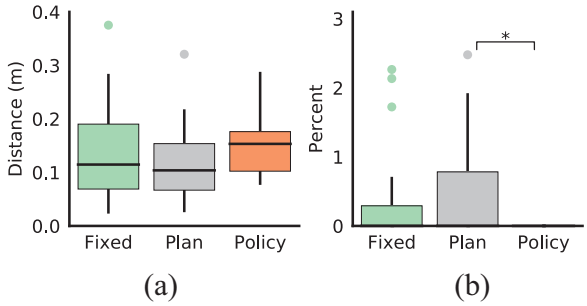



**Fig. 20.** Distance metrics: (a) minimum distance; (b) percent in collision. No difference between methods for minimum distance during interaction, but the policy method yields significantly ($p < 0.05$) less time in collision between human and robot.

a stamping action for each box. To implement the predict-then-act method, the system ran the human goal prediction algorithm from Section 3.4 until a certain confidence was reached (50%), then selected a goal that was not within the restricted set $\mathcal{R}$ and performed a stamping action on that goal. There was no additional human goal monitoring once the goal action was selected. In contrast, our policy implementation performed as described in Section 5, accounting continually for adapting human goals and seamlessly re-planning when the human's goal changed.

*5.1.5. Procedure.* We conducted a within-subjects study with one independent variable (control method) that had three conditions (policy, plan, and fixed). Each performed the gift-wrapping task three times, once with each robot control method. To counteract the effects of novelty and practice, we counterbalanced on the order of conditions.

We recruited 28 participants (14 female, 14 male; mean age 24, SD 6) from the local community. Each participant was compensated US$5 for their time. After providing consent, participants were introduced to the task by a researcher. They then performed the three gift-wrapping trials sequentially. Immediately after each trial, before continuing to the next one, participants completed an eight question Likert-scale survey to evaluate their collaboration with HERB on that trial. At the end of the study, participants provided verbal feedback about the three methods. All trials and feedback were video recorded.

*5.1.6. Results.* Two participants were excluded from all analyses for non-compliance during the study (not following directions). In addition, for the fluency objective measures, five other participants were excluded due to Kinect tracking errors that affected the automatic calculation of minimum distance and time under collision threshold. Other analyses were based on video data and were not affected by Kinect tracking errors.

We assess our hypotheses using a significance level of $\alpha = 0.05$. For data that violated the assumption of sphericity, we used a Greenhouse–Geisser correction. If a significant main effect was found, a post-hoc analysis was used to identify those conditions that were statistically different from each other, with Holm–Bonferroni corrections for multiple comparisons.

To evaluate **H3a** (fluency), we conducted a repeated-measures ANOVA testing the effects of method type (policy, plan, and fixed) on our two measures of human–robot distance: the minimum distance between participant and robot end effectors during each trial, and the proportion of trial time spent with end effector distance below the 8 cm collision threshold (Figure 20). The minimum distance metric was not significant ($F(2,40) = 1.405$, $p = 0.257$). However, proportion of trial time spent in collision was significantly affected by method type ($F(2,40) = 3.639$, $p = 0.035$). Interestingly, the policy method never entered under the collision threshold. Post-hoc pairwise comparisons with a Holm–Bonferroni correction revealed that the policy method yielded significantly ($p = 0.027$) less time in collision than the plan method (policy $M = 0.0\%$, $SD = 0$; plan $M = 0.44\%$, $SD = 0.7$).

Therefore, **H3a** is partially supported: the policy method actually yielded no collisions during the trials, whereas the plan method yielded collisions during 0.4% of the trial time on average. This confirms the intuition behind the differences in the two methods: the policy continually monitors human goals, and thus never collides with the human, whereas the plan method commits to an action once a confidence level has been reached, and is not adaptable to changing human goals.

To evaluate **H3b** (efficiency), we conducted a similar repeated-measures ANOVA for the effect of method type on task durations for robot and human (Figure 21), as well as human time spent idling (Figure 22). Human task duration was highly variable and no significant effect for method was found ($F(2,50) = 2.259$, $p = 0.115$). On the other hand, robot task duration was significantly affected by method

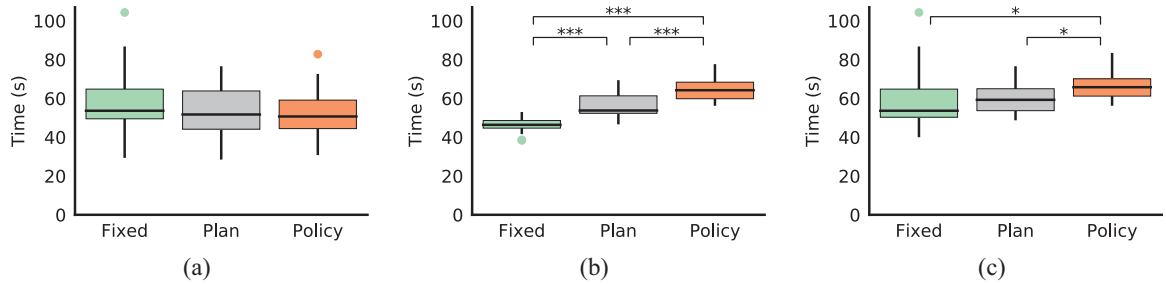



**Fig. 21.** Duration metrics, with pairs that differed significantly during post-analysis are plotted, where $*$ indicates $p < 0.05$ and $***$ that $p < 0.001$: (a) human trial duration; (b) robot trial duration; (c) total trial duration. Human trial time was approximately the same across all methods, but robot time increased with the computational requirements of the method. Total time thus also increased with algorithmic complexity.

condition ($F(2,50) = 79.653$, $p < 0.001$). Post-hoc pairwise comparisons with a Bonferroni correction reveal that differences between all conditions are significant at the $p < 0.001$ level. Unsurprisingly, robot task completion time was shortest in the fixed condition, in which the robot simply executed its actions without monitoring human goals ($M = 46.4$ s, $SD = 3.5$ s). It was significantly longer with the plan method, which had to wait until prediction reached a confidence threshold to begin its action ($M = 56.7$ s, $SD = 6.0$ s). Robot task time was still longer for the policy method, which continually monitored human goals and smoothly replanned motions when required, slowing down the overall trajectory execution ($M = 64.6$ s, $SD = 5.3$ s).

Total task duration (the maximum of human and robot time) also showed a statistically significant difference ($F(2,50) = 4.887$, $p = 0.012$). Post-hoc tests with a Holm–Bonferroni correction show that both fixed ($M = 58.6$ s, $SD = 14.1$ s) and plan ($M = 60.6$ s, $SD = 7.1$ s) performed significantly ($p = 0.026$ and $p = 0.032$, respectively) faster than policy ($M = 65.9$ s, $SD = 6.3$ s). This is due to the slower execution time of the policy method, which dominates the total execution time.

Total idle time was also significantly affected by method type ($F(2,50) = 3.809$, $p = 0.029$). Post-hoc pairwise comparisons with Bonferroni correction reveal that the policy method yielded significantly ($p = 0.048$) less idle time than the fixed condition (policy $M = 0.46$ s, $SD = 0.93$ s, fixed $M = 1.62$ s, $SD = 2.1$ s). Idle time percentage (total idle time divided by human trial completion time) was also significant ($F(2,50) = 3.258$, $p = 0.047$). Post-hoc pairwise tests with Holm–Bonferroni correction finds no significance between pairs. In other words, the policy method performed significantly better than the fixed method for reducing human idling time, while the plan method did not.

Therefore, **H3b** is partially supported: although total human task time was not significantly influenced by method condition, the total robot task time and human idle time were all significantly affected by the method that was running on the robot. The robot task time was slower using the policy method, but human idling was significantly reduced by the policy method.

**Table 2.** Subjective ratings for each method condition, separated by whether a collision occurred during that trial.

| | No collision | | Collision | |
|---|---|---|---|---|
| | *Mean (SD)* | *N* | *Mean (SD)* | *N* |
| Fixed | 5.625 (1.28) | 14 | 4.448 (1.23) | 12 |
| Plan | 5.389 (1.05) | 18 | 4.875 (1.28) | 8 |
| Policy | 5.308 (0.94) | 26 | — | 0 |

To evaluate **H3c** (subjective responses), we first conducted a Chronbach's alpha test to assure that the eight survey questions were internally consistent. The four questions asked in the negative (e.g. "I'm dissatisfied with how HERB and I worked together") were reverse coded so their scales matched the positive questions. The result of the test showed high consistency ($\alpha = 0.849$), so we proceeded with our analysis by averaging together the participant ratings across all eight questions.

During the experiment, participants sometimes saw collisions with the robot. We predict that collisions will be an important covariate on the subjective ratings of the three methods. To account for whether a collision occurred on each trial in our within-subjects design, we cannot conduct a simple repeated measures ANOVA. Instead, we conduct a linear mixed model analysis, with average rating as our dependent variable; method (policy, plan, and fixed), collision (present or absent), and their interaction as fixed factors; and method condition as a repeated measure and participant ID as a covariate to account for the fact that participant ratings were not independent across the three conditions. Table 2 shows details of the scores for each method broken down by whether a collision occurred.

We found that collision had a significant effect on ratings ($F(1,47.933) = 6.055$, $p = 0.018$), but method did not ($F(1,47.933) = 0.312$, $p = 0.733$). No interaction was found. In other words, ratings were significantly affected by whether or not a participant saw a collision, but not by which method they saw independent of that collision.

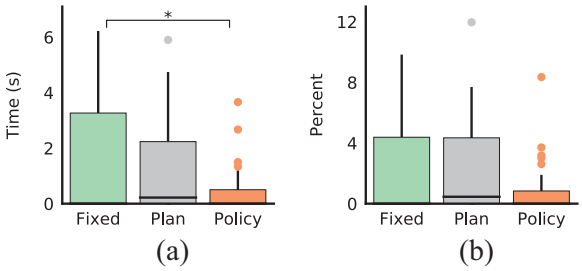



**Fig. 22.** Idle time metrics: (a) human idle time; (b) human idle percentage. Policy yielded significantly ($p < 0.05$) less absolute idle time than the fixed method.

Therefore, **H3c** is not supported directly. However, our analysis shows that collisions lead to poor ratings, and our results above show that the policy method yields fewer collisions. We believe a more efficient implementation of our policy method to enable faster robot task completion, while maintaining fewer collisions, may result in users preferring the policy method.

## 6. Discussion and conclusion

In this work, we have presented a method for shared autonomy that does not rely on predicting a single user goal, but assists for a distribution over goals. Our motivation was a lack of assistance when using predict-then-act methods: in our own experiment (Section 4.2), resulting in no assistance for 69% of execution time. To assist for any distribution over goals, we have formulated shared autonomy as a POMDP with uncertainty over user goals. To provide assistance in real-time over continuous state and action spaces, we used hindsight optimization (Chong et al., 2000; Littman et al., 1995; Yoon et al., 2008) to approximate solutions. We tested our method on two shared-control teleoperation scenarios, and one human–robot teaming scenario. Compared with predict-then-act methods, our method achieves goals faster, requires less user input, decreases user idling time, and results in fewer user–robot collisions.

In our shared control teleoperation experiments, we found user preference differed for each task, even though our method outperformed a predict-then-act method across all objective measures for both tasks. This is not entirely surprising, as prior works have also been mixed on whether users prefer more control authority or better task completion (Dragan and Srinivasa, 2013b; Kim et al., 2012; You and Hauser, 2011). In our studies, user's tended to prefer a predict-then-act approach for the simpler grasping scenario, though not significantly so. For the more-complex eating task, users significantly preferred our shared autonomy method to a predict-then-act method. In fact, our method and blending were the only pair of algorithms that had a significant difference across all objective measures and the subjective measuring of like and rank (Table 3).

However, we believe this difference of rating cannot simply be explained by task difficulty and timing, as the experiments had other important differences. The grasping task required minimal rotation, and relied entirely on assistance to achieve it. Using blending, the user could focus on teleoperating the arm near the object, at which point the predictor would confidently predict the user goal, and assistance would orient the hand. For the feeding task, however, orienting the fork was necessary before moving the arm, at which point the predictor could confidently predict the user goal. For this task, predict-then-act methods usually did not reach their confidence threshold until users completed the most difficult portion of the task: cycling control modes to rotate and orient the fork. These mode switches have been identified as a significant contributor to operator difficulty and time consumption (Herlant et al., 2016). This inability to confidently predict a goal until the fork was oriented caused predict-then-act methods to provide no assistance for the first 29.4 seconds on average, which is greater than the total average time of our method (18.5 s). We believe users were more willing to give up control authority if they did not need to do multiple mode switches and orient the fork, which subjectively felt much more tedious then moving the position.

In all experiments, we used a simple distance-based cost function, for which we could compute value functions in closed form. This enabled us to compute prediction and assistance 50 times a second, making the system feel responsive and reactive. However, this simple cost function could only provide simple assistance, with the objective of minimizing the time to reach a goal. Our new insights into possible differences of user costs for rotation and mode switches as compared with translation can be incorporated into the cost function, with the goal of minimizing user effort.

For human–robot teaming, the total task time was dominated by the robot, with the user generally finishing before the robot. In situations such as this, augmenting the cost function to be more aggressive with robot motion, even at the cost of responsiveness to the user, may be beneficial. In addition, incorporating more optimal robot policies may enable faster robot motions within the current framework.

Finally, though we believe these results show great promise for shared control teleoperation and teaming, we note users varied greatly in their preferences and desires. Prior works in shared control teleoperation have been mixed on whether users prefer control authority or more assistance (Dragan and Srinivasa, 2013b; Kim et al., 2012; You and Hauser, 2011). Our own experiments were also mixed, depending on the task. Even within a task, users had high variance, with users fairly split for grasping (Figure 11), and a high variance for user responses for full autonomy for eating (Figure 18). For teaming, users were similarly mixed in their rating for an algorithm depending on whether or not they collided with the robot (Table 2). This variance

**Table 3.** Post-hoc $p$-value for every pair of algorithms for each hypothesis. For success rate, completion time, mode switches, and total joystick input, results are from a repeated measures ANOVA. For like rating and ranking, results are from a Wilcoxon signed-rank test. All values reported with Holm–Bonferroni corrections.

| Metric | Auton-Policy | Auton-Blend | Auton-Direct | Policy-Blend | Policy-Direct | Blend-Direct |
|---|---|---|---|---|---|---|
| Success Rate | NS | NS | NS | **0.050** | **0.021** | NS |
| Completion Time | **< 0.001** | **< 0.001** | **< 0.001** | **< 0.001** | **< 0.001** | NS |
| Mode Switches | NS | **< 0.001** | **< 0.001** | **< 0.001** | **< 0.001** | NS |
| Control Input | **< 0.001** | **< 0.001** | **< 0.001** | **< 0.001** | **< 0.001** | **0.004** |
| Ranking | NS | NS | NS | **0.006** | NS | NS |
| Like Rating | NS | NS | NS | **0.012** | NS | NS |
| Control Rating | **< 0.001** | **.001** | **< 0.001** | **< 0.001** | **0.008** | **.002** |
| Quickly Rating | NS | NS | NS | **0.010** | **0.043** | NS |

suggests a need for the algorithm to adapt to each individual user, learning their particular preferences. New work by Nikolaidis et al. (2017c) captures these ideas through the user's *adaptability*, but we believe even richer user models and their incorporation into the system action selection would make shared autonomy systems better collaborators.

## Author(s) note

Siddhartha S. Srinivasa is now affiliated to Paul G. Allen School of Computer Science and Engineering, University of Washington.

## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by NSF GRFP (#DGE1252522), NSF NRI (#1227495), NSF CPS (#1544797), DARPA SIMPLEX (through ARO contract #67904LSDRP), the DARPA Autonomous Robotic Manipulation Software Track program, European Project FourByThree (#H2020-FoF-06-2014), the Okawa Foundation, and an Office of Naval Research Young Investigator Award.

## Notes

1. Although we assume the goal is fixed, we do not assume how the user will achieve that goal (e.g. grasp location) is fixed.
2. In Javdani et al. (2015), we presented a framework where only robot actions transition state. The framework presented here also allows user actions to transition state, providing a more general framework. We utilize this generalization for human–robot teaming, where the user affects the environment directly, and in our feeding experiment (Section 4.2), where only the user can change control modes.
3. In practice, we compute the distance of the translation and orientation separately, utilizing the Euclidean distance for translation and the minimum angle between transforms for orientation. We model a separate cost with using the form presented here for each, and add. For brevity, we present here as one cost function
4. In our prior work (Javdani et al., 2015), we used $C^{\mathrm{r}}_{\kappa}(x, a, u) = C^{\mathrm{u}}_{\kappa}(x, a) + (a - u)^2$ in a different framework where only the robot action transitions the state. Both formulations are identical after linearization. Let $a^*$ be the optimal robot action in this framework. The additional term $(a - u)^2$ leads to executing the action $u + a^*$, equivalent to first executing the user action $u$, then $a^*$, as in this framework.
5. In prior works where users preferred greater control authority, task completion times were indistinguishable (Kim et al., 2012).
6. Users verbally specified which bite they wanted for all methods except autonomous, in which the algorithm selects the bite.
7. We sometimes instead observe $x'$ directly (e.g. sensing the pose of the user hand).
8. This study has been previously published in Pellegrinelli et al. (2016). We include results and elaborate here for a unified presentation.

## References

Aarno D, Ekvall S and Kragic D (2005) Adaptive virtual fixtures for machine-assisted teleoperation tasks. In: *IEEE International Conference on Robotics and Automation*.

Aarno D and Kragic D (2008) Motion intention recognition in robot assisted applications. *Robotics and Autonomous Systems* 56: 692–705.

Aigner P and McCarragher BJ (1997) Human integration into robot control utilising potential fields. In: *IEEE International Conference on Robotics and Automation*.

Amershi S, Cakmak M, Knox WB, et al. (2014) Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35: 105–120.

Arai T, Kato R and Fujita M (2010) Assessment of operator stress induced by robot collaboration in assembly. *CIRP Annals - Manufacturing Technology* 59(1): 5–8.

Argall BD (2014) Modular and adaptive wheelchair automation. In: *International Symposium on Experimental Robotics*.

Bandyopadhyay T, Won KS, Frazzoli E, Hsu D, Lee WS and Rus D (2012) Intention-aware motion planning. In: *Workshop on the Algorithmic Foundations of Robotics*.

Bien Z, Chung MJ, Chang PH, et al. (2004) Integration of a rehabilitation robotic system (kares ii) with human-friendly man-machine interaction units. *Autonomous Robots* 16: 165–191.

Boularias A, Kober J and Peters J (2011) Relative entropy inverse reinforcement learning. In: *International Conference on Artificial Intelligence and Statistics*, pp. 182–189.

Carlson T and Demiris Y (2012) Collaborative control for a robotic wheelchair: Evaluation of performance, attention, and

workload. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 42: 876–888.

Chen M, Frazzoli E, Hsu D and Lee WS (2016) POMDP-lite for robust robot planning under uncertainty. In: *IEEE International Conference on Robotics and Automation*.

Chong EKP, Givan RL and Chang HS (2000) A framework for simulation-based network control via hindsight optimization. In: *IEEE Conference on Decision and Control*.

Chung CS, Wang H and Cooper RA (2013) Functional assessment and performance evaluation for assistive robotic manipulators: Literature review. *The Journal of Spinal Cord Medicine* 36: 273–289.

Chung SY and Huang HP (2011) Predictive navigation by understanding human motion patterns. *International Journal of Advanced Robotic Systems* 8(1): 3.

Crandall JW and Goodrich MA (2002) Characterizing efficiency on human robot interaction: A case study of shared–control teleoperation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Debus T, Stoll J, Howe RD and Dupont P (2000) Cooperative human and machine perception in teleoperated assembly. In: *International Symposium on Experimental Robotics*.

Dragan A, Lee K and Srinivasa S (2013) Legibility and predictability of robot motion. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Dragan A and Srinivasa S (2013a) Generating legible motion. In: *Robotics: Science and Systems*.

Dragan A and Srinivasa S (2013b) A policy blending formalism for shared control. *The International Journal of Robotics Research* 32: 790–805.

Fagg AH, Rosenstein M, Platt R and Grupen RA (2004) Extracting user intent in mixed initiative teleoperator control. In: *AIAA*.

Fern A and Tadepalli P (2010) A computational decision theory for interactive assistants. In: *Neural Information Processing Systems*.

Finn C, Levine S and Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. In: *International Conference on Machine Learning*, pp. 49–58.

Goertz RC (1963) Manipulators used for handling radioactive materials. In: Bennett EM, Degan J, and Spiegel J (eds.) *Human Factors in Technology*. New York: McGraw-Hill, pp. 425–443.

Gombolay M, Bair A, Huang C, et al. (2017) Computational design of mixed-initiative human-robot teaming that considers human factors Situational awareness, workload, and workflow preferences. *The International Journal of Robotics Research* 36: 597–617.

Gombolay M, Gutierrez R, Sturla G and Shah J (2014) Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams. In: *Robotics: Science and Systems*.

Goodfellow I, Pouget-Abadie J, Mirza M, et al. (2014) Generative adversarial nets. In: *Neural Information Processing Systems.*

Goodrich MA and Olsen DR (2003) Seven principles of efficient human robot interaction. *IEEE Transactions on Systems, Man, and Cybernetics* 4: 3942–3948.

Gopinath D, Jain S and Argall BD (2016) Human-in-the-loop optimization of shared autonomy in assistive robotics. In: *Conference on Automation Science and Engineering*.

Green S, Billinghurst M, Chen X and Chase JG (2007) Human-robot collaboration: A literature review and augmented reality approach in design. *International Journal of Advanced Robotic Systems* 5: 1.

Guillory A and Bilmes J (2011) Simultaneous learning and covering with adversarial noise. In: *International Conference on Machine Learning*.

Hauser KK (2013) Recognition, prediction, and planning for assisted teleoperation of freeform tasks. *Autonomous Robots* 35: 241–254.

Herlant L, Holladay R and Srinivasa S (2016) Assistive teleoperation of robot arms via automatic time-optimal mode switching. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Ho J and Ermon S (2016) Generative adversarial imitation learning. In: *Neural Information Processing Systems*.

Hoffman G and Breazeal C (2007) Effects of anticipatory action on human–robot teamwork: Efficiency, fluency, and perception of team. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Jain S, Farshchiansadegh A, Broad A, Abdollahi F, Mussa-Ivaldi F and Argall B (2015) Assistive robotic manipulation through shared autonomy and a body–machine interface. In: *IEEE/RAS-EMBS International Conference on Rehabilitation Robotics*.

Javdani S, Srinivasa S and Bagnell JAD (2015) Shared autonomy via hindsight optimization. In: *Robotics: Science and Systems*.

Kaelbling LP, Littman ML and Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101: 99–134.

Katyal KD, Johannes MS, Kellis S, et al. (2014) A collaborative BCI approach to autonomous control of a prosthetic limb system. *IEEE Transactions on Systems, Man, and Cybernetics* 1479–1482.

Kim DJ, Hazlett-Knudsen R, Culver-Godfrey H, et al. (2012) How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 42: 2–14.

Kim HK, Biggs SJ, Schloerb DW, et al. (2006) Continuous shared control for stabilizing reaching and grasping with brain-machine interfaces. *IEEE Transactions on Biomedical Engineering* 53: 1164–1173.

Kofman J, Wu X, Luu TJ, et al. (2005) Teleoperation of a robot manipulator using a vision- based human-robot interface. *IEEE Transactions on Industrial Electronics* 52: 1206–1219.

Koppula H and Saxena A (2013) Anticipating human activities using object affordances for reactive robotic response. In: *Robotics: Science and Systems*.

Koval M, Pollard N and Srinivasa S (2014) Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In: *Robotics: Science and Systems*.

Kragic D, Marayong P, Li M, et al. (2005) Human-machine collaborative systems for microsurgical applications. *The International Journal of Robotics Research* 24: 731–741.

Kurniawati H, Hsu D and Lee WS (2008) Sarsop: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Robotics: Science and Systems*.

Lasota PA and Shah JA (2015) Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. *Human Factors* 57(1): 21–33.

Leeper A, Hsiao K, Ciocarlie M, Takayama L and Gossow D (2012) Strategies for human-in-the-loop robotic grasping. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Levine S and Koltun V (2012) Continuous inverse optimal control with locally optimal examples. In: *International Conference on Machine Learning*.

Li M, Ishii M and Taylor RH (2007) Spatial motion constraints using virtual fixtures generated by anatomy. *IEEE Transactions on Robotics* 23: 4–19.

Li M and Okamura AM (2003) Recognition of operator motions for real-time assistance using virtual fixtures. In: *International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*.

Li Q, Chen W and Wang J (2011) Dynamic shared control for human–wheelchair cooperation. In: *IEEE International Conference on Robotics and Automation*.

Littman ML, Cassandra AR and Kaelbling LP (1995) Learning policies for partially observable environments: Scaling up. In: *International Conference on Machine Learning*.

Macindoe O, Kaelbling LP and Lozano-Pérez T (2012) POMCOP: Belief space planning for sidekicks in cooperative games. In: *Artificial Intelligence and Interactive Digital Entertainment Conference*.

Mainprice J and Berenson D (2013) Human–robot collaborative manipulation planning using early prediction of human motion. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 299–306.

Mainprice J, Sisbot EA, Jaillet L, Cortés J, Alami R and Siméon T (2011) Planning human-aware motions using a sampling-based costmap planner. In: *IEEE International Conference on Robotics and Automation*, pp. 5012–5017.

Marayong P, Li M, Okamura AM and Hager GD (2003) Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures. In: *IEEE International Conference on Robotics and Automation*.

McMullen DP, Hotson G, Katyal KD, et al. (2014) Demonstration of a semi-autonomous hybrid brain-machine interface using human intracranial EEG, eye tracking, and computer vision to control a robotic upper limb prosthetic. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 22: 784–796.

Mehr N, Horowitz R and Dragan AD (2016) Inferring and assisting with constraints in shared autonomy. In: *IEEE Conference on Decision and Control*.

Muelling K, Venkatraman A, Valois J, et al. (2015) Autonomy infused teleoperation with application to BCI manipulation. In: *Robotics: Science and Systems*.

Nguyen THD, Hsu D, Lee WS, et al. (2011) CAPIR: Collaborative action planning with intention recognition. In: *Artificial Intelligence and Interactive Digital Entertainment Conference*.

Nikolaidis S, Hsu D and Srinivasa S (2017a) Human- robot mutual adaptation in collaborative tasks: Models and experiments. *The International Journal of Robotics Research* 36: 618–634.

Nikolaidis S, Nath S, Procaccia A and Srinivasa S (2017b) Game-theoretic modeling of human adaptation in human–robot collaboration. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Nikolaidis S and Shah J (2013) Human-robot cross-training: Computational formulation, modeling and evaluation of a human team training strategy. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Nikolaidis S, Zhu YX, Hsu D and Srinivasa S (2017c) Human–robot mutual adaptation in shared autonomy. In: *ACM/IEEE International Conference on Human–Robot Interaction*.

Pandey AK and Alami R (2010) Mightability maps: A perceptual level decisional framework for co-operative and competitive human–robot interaction. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5842–5848.

Park S, Howe RD and Torchiana DF (2001) Virtual fixtures for robotic cardiac surgery. In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*.

Pellegrinelli S, Admoni H, Javdani S and Srinivasa S (2016) Human–robot shared workspace collaboration via hindsight optimization. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Rezvani T, Driggs-Campbell K, Sadigh D, Sastry SS and Bajcsy R (2016) Towards trustworthy automation: User interfaces that convey internal and external awareness. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*.

Rosenberg LB (1993) Virtual fixtures: Perceptual tools for telerobotic manipulation. In: *IEEE Virtual Reality Annual International Symposium*.

Sadigh D, Sastry SS, Seshia S and Dragan A (2016a) Information gathering actions over human internal state. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Sadigh D, Sastry SS, Seshia SA and Dragan AD (2016b) Planning for autonomous cars that leverage effects on human actions. In: *Proceedings of Robotics: Science and Systems*.

Schrempf OC, Albrecht D and Hanebeck UD (2007) Tractable probabilistic models for intention recognition based on expert knowledge. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Schröer S, Killmann I, Frank B, et al. (2015) An autonomous robotic assistant for drinking. In: *IEEE International Conference on Robotics and Automation*.

Shen J, Ibanez-Guzman J, Ng TC and Chew BS (2004) A collaborative-shared control system with safe obstacle avoidance capability. In: *IEEE International Conference on Robotics, Automation, and Mechatronics*.

Simpson RC (2005) Smart wheelchairs: A literature review. *Journal of Rehabilitation Research and Development* 42: 423–436.

Sisbot EA, Marin-Urias LF, Alami R and Siméon T (2007) A human aware mobile robot motion planner. *IEEE Transactions on Robotics* 23(5): 874–883.

Sisbot EA, Marin-Urias LF, Broquère X, Sidobre D and Alami R (2010) Synthesizing robot motions adapted to human presence. *International Journal of Social Robots* 2(3): 329–343.

Srinivasa S, Berenson D, Cakmak M, et al. (2012) Herb 2.0: Lessons learned from developing a mobile manipulator for the home. *Proceedings of the IEEE* 100(8): 1–19.

Trautman P (2015) Assistive planning in complex, dynamic environments: a probabilistic approach. In: *HRI Workshop on Human Machine Teaming*.

Vanhooydonck D, Demeester E, Nuttin M and Brussel HV (2003) Shared control for intelligent wheelchairs: an implicit estimation of the user intention. In: *Proceedings of the ASER International Workshop on Advances in Service Robotics*.

Vogel J, Haddadin S, Simeral JD, et al. (2014) Continuous control of the DLR Light-weight Robot III by a human with tetraplegia

using the Braingate2 neural interface system. In: *International Symposium on Experimental Robotics*, volume 79.

Wang Z, Mülling K, Deisenroth MP, et al. (2013) Probabilistic movement modeling for intention inference in human-robot interaction. *The International Journal of Robotics Research* 32: 841–858.

Yoon S, Fern A and Givan R (2007) FF-REPLAN: A baseline for probabilistic planning. In: *International Conference on Automated Planning and Scheduling*.

Yoon SW, Fern A, Givan R and Kambhampati S (2008) Probabilistic planning via determinization in hindsight. In: *AAAI Conference on Artificial Intelligence*.

You E and Hauser K (2011) Assisted teleoperation strategies for aggressively controlling a robot arm with 2D input. In: *Robotics: Science and Systems*.

Yu W, Alqasemi R, Dubey RV and Pernalete N (2005) Telemanipulation assistance based on motion intention recognition. In: *IEEE International Conference on Robotics and Automation*.

Ziebart BD (2010) *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD Thesis, Machine Learning Department, Carnegie Mellon University.

Ziebart BD, Dey A and Bagnell JAD (2012) Probabilistic pointing target prediction via inverse optimal control. In: *International Conference on Intelligence User Interfaces*.

Ziebart BD, Maas A, Bagnell JAD and Dey A (2008) Maximum entropy inverse reinforcement learning. In: *AAAI Conference on Artificial Intelligence*.

Ziebart BD, Ratliff N, Gallagher G, et al. (2009) Planning-based prediction for pedestrians. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

## Appendix A Multi-target MDPs proofs

In the following, we provide the proofs for decomposing the value functions for MDPs with multiple targets, as introduced in Section 3.5.

### *A.1 Theorem 1: decomposing value functions*

Here, we show the proof for our theorem that we can decompose the value functions over that the targets for deterministic MDPs. The proofs here are written for our shared autonomy scenario. However, the same results hold for any deterministic MDP.

*Proof.* Proof of Theorem 1 We show how the standard value iteration algorithm, computing $Q_g$ and $V_g$ backwards, breaks down at each time step. At the final timestep $T$, we obtain

$$\begin{aligned} Q_g^T(x,u,a) &= C_g(x,u,a) \\ &= C_\kappa(x,u,a) \qquad \text{for any } \kappa \\ V_g^T(x) &= \min_a C_g(x,u,a) \qquad u = \pi^{\mathrm{u}}(x) \\ &= \min_a \min_\kappa C_\kappa(x,u,a) \\ &= \min_\kappa V_\kappa^T(x) \end{aligned}$$

Let $\kappa^* = \arg\min V_\kappa(x')$ as before. Now, we show the recursive step:

$$\begin{aligned} Q_g^{t-1}(x,u,a) &= C_g(x,u,a) + V_g^t(x') \\ &= C_{\kappa^*}(x,u,a) + \min_\kappa V_\kappa^t(x') \\ &= C_{\kappa^*}(x,u,a) + V_{\kappa^*}^t(x') \\ &= Q_{\kappa^*}(x,u,a) \end{aligned}$$

$$\begin{aligned} V_g^{t-1}(x) &= \min_a Q_g^{t-1}(x,u,a) \qquad u = \pi^{\mathrm{u}}(x) \\ &= \min_a C_{\kappa^*}(x,u,a) + V_{\kappa^*}^t(x') \\ &\geq \min_a \min_\kappa \left( C_\kappa(x,u,a) + V_\kappa^t(x') \right) \\ &= \min_\kappa V_\kappa^{t-1}(x) \end{aligned}$$

In addition, we know that $V_g(x) \leq \min_\kappa V_\kappa(x)$, since $V_\kappa(x)$ measures the cost-to-go for a specific target, and the total cost-to-go is bounded by this value for a deterministic system. Therefore, $V_g(x) = \min_\kappa V_\kappa(x)$. □

### *A.2 Theorem 2: decomposing soft value functions*

Here, we show the proof for our theorem that we can decompose the soft value functions over that the targets for deterministic MDPs.

*Proof.* Proof of Theorem 2 As the cost is additive along the trajectory, we can expand out $\exp(-C_\kappa(\xi))$ and marginalize over future inputs to obtain the probability of an input now:

$$\pi^{\mathrm{u}}(u_t, \kappa | x_t) = \frac{\exp(-C_\kappa(x_t,u_t)) \int \exp(-C_\kappa(\xi_{x_{t+1}}^{t+1\to T}))}{\sum_{\kappa'} \int \exp(-C_{\kappa'}(\xi_{x_t}^{t\to T}))}$$

where the integrals are over all trajectories. By definition, $\exp(-V_{\kappa,t}^{\approx}(x_t)) = \int \exp(-C_\kappa(\xi_{x_t}^{t\to T}))$:

$$\begin{aligned} &= \frac{\exp(-C_\kappa(x_t,u_t)) \exp(-V_{\kappa,t+1}^{\approx}(x_{t+1}))}{\sum_{\kappa'} \exp(-V_{\kappa',t}^{\approx}(x_t))} \\ &= \frac{\exp(-Q_{\kappa,t}^{\approx}(x_t,u_t))}{\sum_{\kappa'} \exp(-V_{\kappa',t}^{\approx}(x_t))} \end{aligned}$$

Marginalizing out $\kappa$ and simplifying:

$$\begin{aligned} \pi^{\mathrm{u}}(u_t|x_t) &= \frac{\sum_\kappa \exp(-Q_{\kappa,t}^{\approx}(x_t,u_t))}{\sum_\kappa \exp(-V_{\kappa,t}^{\approx}(x_t))} \\ &= \exp\left( \log\left( \frac{\sum_\kappa \exp(-Q_{\kappa,t}^{\approx}(x_t,u_t))}{\sum_\kappa \exp(-V_{\kappa,t}^{\approx}(x_t))} \right) \right) \\ &= \exp\left( \operatorname*{softmin}_\kappa V_{\kappa,t}^{\approx}(x_t) - \operatorname*{softmin}_\kappa Q_\kappa^{\approx} t(x_t,u_t) \right) \end{aligned}$$

As $V_{g,t}^{\approx}$ and $Q_{g,t}^{\approx}$ are defined such that $\pi_t^{\mathrm{u}}(u|x,g) = \exp(V_{g,t}^{\approx}(x) - Q_{g,t}^{\approx}(x,u))$, our proof is complete. □