Minimizing User Cost for Shared Autonomy

Shervin Javdani, J. Andrew Bagnell, Siddhartha S. Srinivasa
Robotics Institute, Carnegie Mellon University
{sjavdani, dbagnell, siddh}@cs.cmu.edu

***Abstract*—In *shared autonomy*, user input and robot autonomy are combined to control a robot to achieve a goal. One often used strategy considers the user and autonomy as *independent* decision makers, with the system blending these decisions. However, this independence leads to suboptimal, and often frustrating, behavior. Instead, we propose a system that explicitly models the interplay between the user and assistance. Our approach centers around the idea of learning how users respond to assistance. We then propose a cost minimization framework for assisting while utilizing this learned model.**

I. Introduction

Robotic teleoperation enables a user to achieve their goal by providing inputs into a robotic system. In *direct teleoperation*, user inputs are mapped directly to robot actions, putting the burden of control entirely on the user. However, input interfaces are noisy, and often have fewer degrees of freedom than the robot they control. This makes operation tedious, and many goals impossible to achieve. *Shared autonomy* seeks to alleviate these issues by combining manual teleoperation with autonomous assistance.

Dragan and Srinivasa [1] show that nearly all prior shared autonomy methods can be thought of as a *blending* between two independent sources - user inputs and an autonomous policy. Methods differ in the autonomous policy used, and how the blending is done. Potential field methods [2] combine direct teleoperation with a policy which pushes the robot away from obstacles and towards goals. Virtual fixtures [3], [4] combine direct teleoperation with an autonomous policy that projects the robot onto path constraints. Autonomous task completing systems [5], [6] use goal achieving policies, and give full control to either the user or autonomy. Linear blending [1] uses goal achieving policies, and blend by utilizing goal prediction confidence to smoothly switch between sources.

While blending is intuitive, Trautman [7] presents many examples where user inputs and the autonomous policy are reasonable in isolation, but their blended combination is not. These examples demonstrate a key shortcoming of the blending approach - the autonomous policy selects assistance actions *independent* of user inputs.

In most existing systems, user inputs have been used to predict the user's goal and select the autonomous policy [1], [4], [8], [10]. However, these systems do not reason about user inputs in the autonomous policy itself. Additionally, these predictors are generally learned from direct teleoperation, without the shared autonomy system in place. Experiments, on the other hand, suggest that users change their strategy when assistance is present [9].

More recent work has proposed minimizing a *user-robot cost function* [8], [9]. These frameworks take a step towards optimizing for assistance actions while incorporating a user model. However, current implementations do not incorporate predictions of future user inputs into the policies.

Our aim is predict how users respond to assistance, and utilize these predictions to select assistance actions. Intuitively, if certain assistance actions cause users to fight the system, we should predict that fight and penalize those actions. If other actions cause users to achieve their goal while incurring lower cost, we should prefer those actions.

We first review the cost minimization framework of Javdani et al. [9] assuming we have a model of user behavior. We next present a method for learning user behavior during assistance. Finally, we discuss some ongoing computational challenges for making this system practical.

II. Assistance Action Selection

We present our user-robot cost minimization framework for a known goal. Note that we can extend this cost minimization to an unknown goal by following Javdani et al. [9] and using the QMDP method [11].

A. User-Robot Cost Minimization

Formally, let $s \in S$ be the robot state (e.g. position, velocity), and $a \in A$ be the actions (e.g. velocity, torque). We model the robot as a dynamical system with transition function $T : S \times A \to S$. The user supplies inputs $u \in U$ via an interface (e.g. joystick, mouse). These user inputs map to robot actions through a known deterministic function $\mathcal{D} : U \to A$, corresponding to the effect of *direct teleoperation*. We define a trajectory ξ_t as a sequence of states, user inputs, and actions, $\xi_t = \{s_0, u_0, a_0, \ldots, u_{t-1}, a_{t-1}, s_t\}$.

We assume a known user-robot cost function $C : S \times U \times A \to \mathcal{R}$. This cost function can be hand-tuned or learned, e.g. through maximum entropy inverse optimal control (MaxEnt IOC) [12]. Together, the tuple $(S, U, T, C^{\mathrm{u}})$ defines a Markov Decision Process (MDP).

Unlike standard MDP formulations, we cannot directly optimize for actions, as we do not decide user inputs. In order to find the optimal robot actions, we assume access to a stochastic user policy $\pi^{\mathrm{u}}(\xi_t) = p(u_t|\xi_t)$, which provides a distribution over user inputs given the entire history, including the previous assistance actions. In Sec. III, we discuss how we can learn this policy.

We similarly define the assistance policy $\pi^{\text{r}}(s) = p(a|s)$. This allows us to define the *value function*, or expected cost-to-go, given a user policy:

$$\begin{aligned} V^{\pi^{\text{u}}}_{\pi^{\text{r}}}(s_0) &= \sum_t \mathbb{E}_{s_t,u_t,a_t}[C(s_t,u_t,a_t)] \\ s_t &\sim T(s_{t-1},a_{t-1}) \\ u_t &\sim \pi^{\text{u}}(\xi_t) \\ a_t &\sim \pi^{\text{r}}(s_t) \end{aligned}$$

We denote the optimal value function as the expected cost-to-go of the best policy, $V^{\pi^{\text{u}}}(s) = \min_{\pi^{\text{r}}} V^{\pi^{\text{u}}}_{\pi^{\text{r}}}(s)$.

Note that in prior work [9], this value function was approximated by assuming the user would not supply more inputs:

$$V^{\pi^{\text{u}}}_{[9]}(s_0) = \min_{\pi^{\text{r}}} \sum_t \mathbb{E}_{s_t,a_t}[C(s_t,0,a_t)]$$

This assumption was made for computational purposes - rolling out the user policy while selecting assistance actions is computationally difficult. However, if we wish to incorporate the user model into action selection - for example, to avoid fighting the user - we must relax this assumption. We discuss some potential approximations in Sec. IV.

III. Learning the User Policy

Most shared autonomy works have focused on utilizing predictors to infer a distribution over the user's goal [1], [4], [8], [10]. These methods learn a predictor during *direct teleoperation*, and apply it during shared autonomy [1], [8]–[10]. Implicitly, this assumes that users do not change their behavior when assistance is provided. However, recent studies suggest that users alter their behavior during assistance [9].

A. Incorporating User Adaptation

The way in which users respond to assistance actions provides information about the effectiveness of those actions. Intuitively, if the user fights certain assistance actions, we should actively avoid those assistance actions. On the other hand, if we can predict that some assistance actions will enable users to achieve their goal with fewer inputs, and therefore less cost, we should prefer those actions.

Learning a good model of user behavior during assistance is our ongoing work. Intuitively, we believe that user behavior during assistance will resemble that of direct teleoperation. Let p^{me} be a predictor of direct teleoperation behavior, e.g. learned through maximum entropy inverse optimal control (MaxEnt IOC) [12]. Our current approach employs the principle of *minimum cross-entropy* [13], learning a predictor that matches the observed data while minimizing the Kullback-Leibler divergence (KL) to this prior distribution. This has the additional benefit of leveraging existing work on user predictions.

Let f^{ξ}_u be some features of user input u and trajectory so far ξ. Let $\overline{f}^{\xi}_u$ be the average feature observed in the data:

$$\begin{aligned} &\arg\min_{p^{\text{kl}}} \ \text{KL}(p^{\text{kl}} \| p^{\text{me}}) \\ &\text{s.t.} \sum_{\xi \in \text{Data}} p(\xi) \sum_u p^{\text{kl}}(u|\xi) f^{\xi}_u = \overline{f}^{\xi}_u \end{aligned}$$

That is, the average feature of the data $\overline{f}^{\xi}_u$ should match the expected feature predicted by our learned distribution p^{kl} on the trajectories observed in the data.

IV. Ongoing Work

We presented our current formulation for predicting user actions as users adapt to shared autonomy assistance in Sec. III, and how we might optimally use that prediction for selecting new assistance actions in Sec. II. However, these methods as presented are computationally intractable.

Ziebart et al. [14] have utilized predictions of user inputs to compute optimal actions in an MDP in the discrete setting. They use predictions to create and optimize over time-dependent cost maps. We are currently exploring extensions of this idea for the continuous setting. Another possibility is to approximate by assuming a value function approximation after a bounded number of steps, e.g.:

$$V^{\pi^{\text{u}}}(s_0) \approx \min_{\pi^{\text{r}}} \sum_t^T \mathbb{E}_{s_t,u_t,a_t}[C(s_t,u_t,a_t)] + \widetilde{V}(s_T)$$

Where $\widetilde{V}$ is some estimate of the cost-to-go, e.g. assuming the robot will take over as in Javdani et al. [9].

Extending this framework to continuous action spaces presents many additional challenges. We hope to follow methods recently presented for continuous action POMDPs [15].

References

[1] A. Dragan and S. Srinivasa, "A policy blending formalism for shared control," *IJRR*, 2013.

[2] J. W. Crandall and M. A. Goodrich, "Characterizing efficiency on human robot interaction: a case study of shared–control teleoperation," in *IEEE/RSJ IROS*, 2002.

[3] S. Park, R. D. Howe, and D. F. Torchiana, "Virtual fixtures for robotic cardiac surgery," in *Med. Image. Comput. Comput. Assist. Interv.*, 2001.

[4] M. Li and A. M. Okamura, "Recognition of operator motions for real-time assistance using virtual fixtures," in *HAPTICS*, 2003.

[5] A. H. Fagg, M. Rosenstein, R. Platt, and R. A. Grupen, "Extracting user intent in mixed initiative teleoperator control," in *Proceedings of the American Institute of Aeronautics and Astronautics Intelligent Systems Technical Conference*, 2004.

[6] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transa. Ind. Electron.*, 2005.

[7] P. Trautman, "Assistive planning in complex, dynamic environments: a probabilistic approach," in *HRI Workshop Hum. Rob. Team.*, 2015.

[8] K. K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Auton. Robots*, vol. 35, 2013.

[9] S. Javdani, S. Srinivasa, and J. A. D. Bagnell, "Shared autonomy via hindsight optimization," in *RSS*, 2015.

[10] H. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," in *RSS*, 2013.

[11] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *ICML*, 1995.

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

[13] J. E. Shore and R. W. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy." *IEEE Trans. Info. Theory*, vol. 26, pp. 26–37, 1980.

[14] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. D. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *IEEE/RSJ IROS*, 2009.

[15] K. Seiler, H. Kurniawati, and S. Singh, "An online and approximate solver for pomdps with continuous action space," in *IEEE ICRA*, 2015.