Modeling User Expertise for Choosing Levels of Shared Autonomy

Lauren Milliken and Geoffrey A. Hollinger

Abstract— **In shared autonomy, a robot and human user both have some level of control in order to achieve a shared goal. Choosing the balance of control given to the user and the robot can be a challenging problem since different users have different preferences and vary in skill levels when operating a robot. We propose using a novel formulation of Partially Observable Markov Decision Process (POMDP) to represent a model of the user's expertise in controlling the robot. The POMDP uses observations from the user's actions and from the environment to update the belief of the user's skill and chooses a level of control between the robot and the user. The level of control given between the user and the robot is encapsulated in macro-action controllers. A user study was run to test the performance of our formulation. Users drive a simulated robot through an obstacle-filled map while the POMDP model chooses appropriate macro-action controllers based on the belief state of the user's skill level. The results of the user study show that our model can encapsulate user skill. The results also show that using the controller with greater robot autonomy helped users of low skill avoid obstacles more than it helped users of high skill.**

I. Introduction

When deploying robots in the field, there are many advantages to using shared autonomy, where both the human operator and the robot have some level of control. Direct teleoperation may be tedious or difficult, and assistance from the robot may be able to take away a great deal of burden from the human user. However, different users may have varying needs when it comes to the level of autonomy the robot should be given. A novice user may need a great deal of assistance in performing even basic tasks in order to complete the goal safely. On the other hand, a user with more experience may be able to accomplish these tasks easily without as much assistance from the robot, and may even dislike the lack of control in certain situations. Different users also have different learning curves. Some may quickly become familiar with the system, while others might need assistance for longer. In this paper, we present a method of modeling a human user's expertise in systems using shared autonomy.

We propose a novel method using Partially Observable Markov Decision Processes (POMDPs) to model user expertise and choose the optimal level of shared autonomy. Fig. 1 shows a visualization of how our model uses the human operator's actions to make a prediction of what state of expertise they are at. This paper is a continuation of our prior workshop paper [1]. The shared control of the system is encompassed by macro-action controllers. The macro-actions are designed to model the robot controls for varying levels of autonomy. The actions encompassed by the controllers are designed for users of varying expertise. Rather than solving the POMDP with many low-level actions, the macro-action controllers encapsulate the shared autonomy control that maximizes the performance based on the user's expertise.

This research is funded in part by grant NSF IIS-1317815 and support from PCC Structurals, Inc. and Oregon Metals Initiative. User studies were performed under Oregon State University IRB 6656.

L. Milliken and G. Hollinger are with the Robotics Program of the School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR. (e-mail: {millikel, geoff.hollinger}@oregonstate.edu)

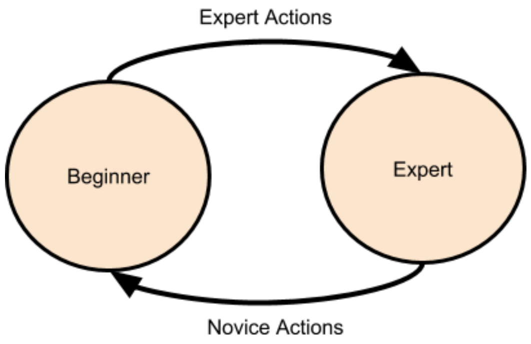


Fig. 1: As the user performs more actions expected from an expert, the likelihood the user is classified as an expert increases. If the user performs more actions expected from a novice, they will be more likely to be classified as a beginner.

Our POMDP model can be used to predict the level of a user's expertise and assign a controller that will best assist them. A system created for expert users may be difficult or even dangerous for beginners to use. On the other hand, an expert may find a system created for beginners to be too simplistic or they may be skilled enough to control the robot in ways that would be otherwise unsafe for someone not familiar with the system. By modeling user expertise, we can create a flexible interface for the robot that can work for users of many skill levels. The novelty of our method lies in the ability to find the skill level of users and assign them macro-controllers based on that expertise. Past work in shared autonomy has shown that different users have different preferences and requirements for the levels of robot autonomy used [2]–[4]. Our method is able to assign controllers for users of different skill levels without the need to solve a POMDP online, which is computationally intractable. Our model is able to adapt to user's skill level over time, such as a beginner improving or an expert becoming fatigued. A user study testing the performance of our formulation in a robot driving simulator confirmed the benefits of our approach.

The rest of the paper is organized as follows. Section II gives an overview of related works. Section III describes the implementation of the model and the domain we designed to test it. Section IV describes the user trial, Section V shows the results, and Section VI discusses the results and future work.

II. Related Works

Many tasks benefit from collaboration between robots and humans, but often this collaboration comes with trade-offs. Shared autonomy can reduce the workload for a human operator in tasks such as search and rescue [5], stable control of marine and aerial vehicles [6], [7], or driving semi-autonomous cars [8]. The robot can do much of the work that would have otherwise required human teleoperation, but the user still needs to maintain situational awareness and an understanding of what the robot will do. This should not increase the workload for the human [9]. However, it is not always clear what level of shared autonomy is needed in different situations since the amount of control between the human and the robot can be very broad [10]. Our method is used to provide different levels of shared autonomy based on an adaptive model of the user's expertise.

Previous work using shared autonomy has looked at the arbitration between the user's input and the robot's assistance [2], [11]. Such work has found that even when assistance from the robot decreases the task completion time, some users still preferred feeling in control of the robot. By encompassing the user's actions, both desirable and undesirable, the system can choose behaviors that both optimize reaching the goal and perform actions preferred by the user. A user study done in [12] found that using assisted teleoperation resulted in fewer obstacles hit, but an increase in time to complete a map. The study also found that users with video game experience found the task more enjoyable and less physically demanding.

Much of the work done in improving shared autonomy looks at being able to predict the user's intentions. Work from [4] showed that in human-robot collaboration, the robot needs the ability to predict the human user's intent and must also act in an intent-expressive way that enables the human to predict what the robot will do. Work done in [13] uses Gaussian mixture models to predict user intention in free-form tasks and a cooperative motion planner is used to generate trajectories based on the user's desired task. Predicting the user's intent has also been shown in assistive technology such as robotic wheelchairs [14], [15] in order to assist users with limited physical capabilities control their wheelchairs. In our work, rather than continuously predicting the user's actions, we use the prediction of their skill level in order to make a generalization of what actions the user will most likely perform. For instance, a user classified as a beginner is more likely to choose poor actions, so the model is used to choose an appropriate controller that provides greater assistance to the user.

Previous work has designed robots to adapt to different users based on their past actions [3]. These methods work to model the what the user prefers the robot to do. Robot autonomy often can decrease the workload and improve efficiency, but some users are unable or unwilling to cooperate with the robot. If the user is trying to work against the robot's actions, workload and frustration in the human will naturally increase. Work done in [3] models the human's willingness to adapt to the robot's actions to improve the effectiveness of the team while retaining human trust in the robot. A method proposed in [16] is used to find how the user reacts to assistive actions and how to choose the most assistive robot actions using the user model. Other work has looked at how robots can learn how to work best side-by-side with the human user. The cross-teaming method of [17] is used to teach a robot to adapt to the preferences of the user. This method uses an MDP to encode a mental model that learns the user's preferences in completing a certain task through training with the user. Our work similarly uses a Partially Observable Markov Process to model the human user, but we utilize the Markov process to find user expertise in a way that is general enough to be used on multiple systems.

III. Algorithm Description

A. User Model

We leverage the POMDP framework to learn a human's level of expertise and use this model to determine the level of autonomy to give the robot. The POMDP model is a tuple $<S,A,O,T,\Omega,R,b_o,\gamma>$. The set of states, S, encompasses the user's level of expertise. The set of observations, O, includes the observations from the environment as well as the actions performed by the user, which reflect either expert or novice skills. These observations may include how many times the user came close to hitting or hit an obstacle, how quickly they completed the goal, or if they are operating the robot at constant speeds or rapidly accelerating. The actions, A, are macro-actions controllers. These controllers are developed for users at different skill levels or as transitional controllers when users are moving between skill levels. T, Ω, R, b_o, and γ represent the conditional transition probabilities between states, the conditional observation probabilities, the reward function, the initial belief, and the discount factor, respectively. Currently, the values of the POMDP tuple are hand-tuned, but could potentially be learned from data.

B. Shared Autonomy Policy

With knowledge of the user's skill level, we can use controllers that will best assist the user based on what actions they are most likely to perform. For our experiments, we used the Approximate POMDP Planning Toolkit (APPL) as an offline POMDP solver [18]. APPL implements the SARSOP algorithm to approximately solve the POMDP. Hand-tuned values of the POMDP tuple were input into APPL to generate a policy offline. The policy π is computed to maximize the expected total reward.

We use the POMDP and the policy generated offline to update the belief of the user's level of expertise as they operate the robot. At each time step, the policy π is used to select the macro-action controller $a \in A$ based on the current belief state b. The controller provides some level of shared autonomy to the user. When the time step completes, the observation $o \in O$ is received based on the user's actions,

Algorithm 1 User Expertise Prediction and Controller Selection

```
1: procedure CHOOSECONTROLLER($\pi$,$b$)
2:     $a \leftarrow \pi(b)$
3:     while not at update condition do
4:         Use macro-action controller, $a$
5:         Record observation, $o$
6:     $b' = \tau(b, a, o)$
7:     $b \leftarrow b'$
   return $b$
```

TABLE I: POMDP for modeling user expertise

S	Beginner, Expert
A	LeastHelp, SomeHelp, MoreHelp, MostHelp
O	No Obstacles Hit-Easy, Few Obstacles Hit-Easy, Many Obstacles Hit-Easy, No Obstacles Hit-Hard, Few Obstacles Hit-Hard, Many Obstacles Hit-Hard

the actions taken by the controller, and the observation of the environment. The belief state is updated based on the current belief, the controller used, and the observation given by $b' = \tau(b, a, o)$. The process then continues with the new belief state. The complete procedure is described in Algorithm 1.

C. Domain

For our experiments, a robotic driving simulator was created using the pygame library of Python [19]. A picture of the simulator is shown in Fig. 2. The user drives the robot using the WASD keys on the computer keyboard with the objective of reaching the goal as fast as they can.

The states of the world represent the user's expertise and the difficulty of the map. Easy maps contained fewer, more spread out, obstacles while hard maps contained a greater number of obstacles which require finer maneuvering to avoid. The observations in the model represented the perceived difficulty of the map and the amount the user collided with an obstacle. Four controllers were created that provided different levels of assistance in avoiding obstacles. These controllers are described in more detail below. Table I shows the states, actions, and observations chosen for our experiments. The values of the POMDP were set so that the policy assigns user's with a high probability of being at the *Expert* state the "LeastHelp" controller. Those with a high probability of being at the *Beginner* state are assigned the "MostHelp" controller. When the probability is distributed more evenly between *Beginner* and *Expert*, the "SomeHelp" or "MoreHelp" controllers are assigned.

D. Robot Controller

The macro-action controllers can be set up for different user levels and environmental states. By selecting macro-action controllers rather than low-level motions, the number of actions the POMDP must be solved for can be greatly reduced [20]. In a human-robot system, some of the burden of modeling the state of the world is taken over by the human rather than relying on the POMDP alone. Our macro-action controllers combine the actions of the user and the robot so that the POMDP requires a fewer number of states and observations to navigate through the environment. By encapsulating the user skill level in our model, we can predict a range of probabilities that certain actions will be performed by the user depending on their skill and the difficulty of the task.

The macro-action controllers were created with the possible differences between beginners and experts in mind. The principal difference being that beginners would be more likely to hit obstacles. We note that our model is general enough to incorporate many different controllers and observations across domains, and future work will incorporate more complex observations of novice and expert actions.

In order to assist the user in avoiding obstacles, the controllers use a potential field method similar to the method presented in [21]. When obstacles are within range of the robot, the controller attempts to slow the robot down and steer it away from the incoming obstacle. To balance the control between the user and the robot, a variable $U = [0, 1]$ defines the amount of user influence. As the user presses the control buttons on the keyboard, U increases. Once the button is released, U decreases. The user influence affects both the velocity and the heading of the system with U_v representing the user's influence over velocity (when pressing the W or S keys) and with U_r representing the user's influence over rotation (when pressing the A or D keys). If they hold the key long enough, the user can override the robot's actions when it tries to slow or rotate. The variable v is the current speed of the robot, α_1 is the factor for the amount slowed when approaching an obstacle from any side, and α_2 is the variable for the amount slowed when approaching an obstacle from the front of the robot. The speed of the robot is calculated as shown below.

$$v_{new} = v - (1 - U_v)(\alpha_1 + \alpha_2)sign(v) \tag{1}$$

The angle of rotation from the user's control is defined as θ_U, the repelling angle from the potential field of the obstacles is defined as θ_R, and β is a weighting factor for the amount of control given to the robot. Below is the final

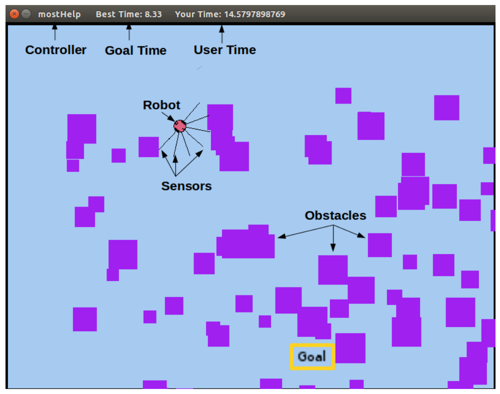


Fig. 2: Simulated driving environment used to test macro-action POMDP.

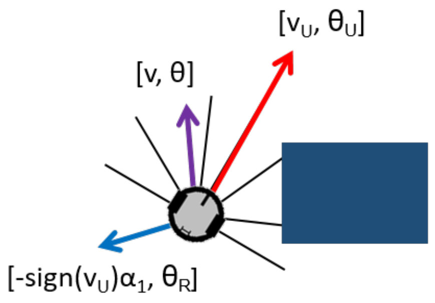


Fig. 3: The effect of the POMDP macro-action controllers. If the user drives towards an obstacle, the force of the robot turns away from the obstacle and slows the robot down. Since the robot is approaching the obstacle from the side, α_1 is activated, but α_2 is not.

angle of rotation for the robot leveraging both the user's and the robot's steering commands:

$$\theta = U_r\theta_U + \beta(1 - U_r)\theta_R \tag{2}$$

The controller's effect on the robot's velocity and steering is shown in Fig 3 While the controllers designed for the beginner states were able to avoid obstacles, they also took away some of the control from the user and would decrease the speed of the robot. This may be preferred by a beginner who may require the robot to assist them, but could slow down an expert user. The differences in the controllers are shown in Table II. These parameters were derived from testing of the simulator in pilot trials.

TABLE II: The parameters set for each of the controllers.

Controller	*Parameters*
MostHelp	$\alpha_1 = 2$, $\alpha_2 = 0.80$, $\beta = 2.0$
MoreHelp	$\alpha_1 = 2$, $\alpha_2 = 0.60$, $\beta = 1.5$
SomeHelp	$\alpha_1 = 1$, $\alpha_2 = 0.45$, $\beta = 1.0$
LeastHelp	$\alpha_1 = 0$, $\alpha_2 = 0.25$, $\beta = 0.0$

IV. User Study

A user study was designed to verify that (1) the POMDP model is able to make accurate predictions of user expertise, (2) users predicted by the POMDP as more likely to be experts perform better than the users predicted more likely to be beginners, and (3) using the different controllers will result in differences in user performance.

Fourteen able-bodied users (6 male, 8 female, mean age 24) were recruited to participate in the user study through emails and fliers distributed through Oregon State University. The participants were compensated $5 USD for the 30-minute experiment. The users were given a brief overview of the robot's level of autonomy and how to drive the robot using the WASD keys.

The participants were given the robot driving simulator described in Section III. The participants are given the view as shown in Fig. 2. At the top of the screen, the user is shown which controller they are currently using, a "best time", and their own time. The controller that is being used by the robot is shown to the user to give the robot's autonomy some transparency. Pilot trials of the simulator showed us that when the users were not shown which controller was being used, they would often become frustrated when a different controller was chosen. Often they would not realize a new controller was chosen and would drive the robot as if it would behave the same as the last map. This often led to mistakes before the user realized the controller was different. The user's time and the goal time are shown to motivate the users to complete the map quickly. The same maps were presented in the same order for every user. This was done so that we could compare every user under the same conditions.

The users first drove the robot through the "training maps", 30 pre-generated maps of varying difficulty. The initial belief state is set at the beginning of the trial with equal probability the user is a *Beginner* or *Expert*. Each user also starts with the "SomeHelp" controller. The belief state is updated once the user completes the map, and the controller to be used in the next map is chosen by checking the policy with the new belief state.

After completing the training maps, the user drove through 5 "test maps". Instead of using the POMDP model to choose the controller, they were given either the "LeastHelp" or the "MostHelp" controller. The users then ran those same 5 maps a second time with the other controller. During the test maps, the controller used first is counterbalanced so that half the users started with the "LeastHelp" controller and half started with the "MostHelp" controller. Therefore, the results will not be skewed from seeing the maps a second time. We used the controllers with the most/least autonomy in the test maps so that we could compare how the amount of autonomy affected the users.

V. Results

A. *Training Maps*

Fig. 4 shows the total number of obstacles hit versus the mean probability of the user being a *Beginner* during the training maps. Fig. 5 shows the total time in seconds to complete the training maps versus the mean probability of the user being a *Beginner*. Though there is variance in the results, the plots show that the users who have a higher probability of being a *Beginner* did perform worse than users who had a lower probability. Users who had a high average probability of being a *Beginner* tend to hit more obstacles and take longer to complete the maps than those with a low average probability. The probability that the model predicts the user is an *Expert* over the training maps is shown for three of the users in Fig. 6. User 1 has a high probability of being an *Expert* through most of the maps. User 2 has a low probability of being an expert for the first 15 maps, but then the probability increases, showing that they have improved. User 3 may have performed well on some maps, but never performed consistently enough for the POMDP to give them a high probability of being an *Expert* by the end of the maps.

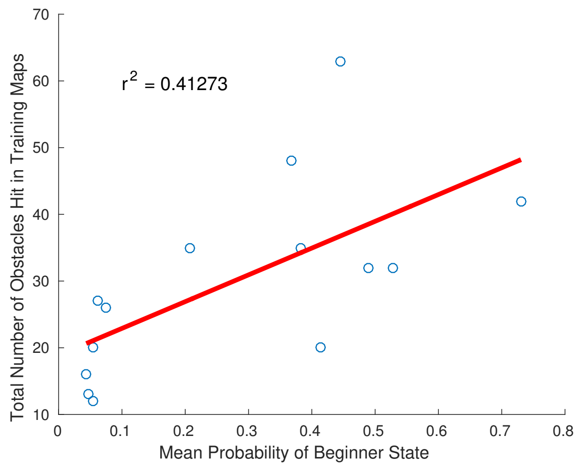


Fig. 4: The total number obstacle hits in the training maps versus the mean probability of being a *Beginner*.

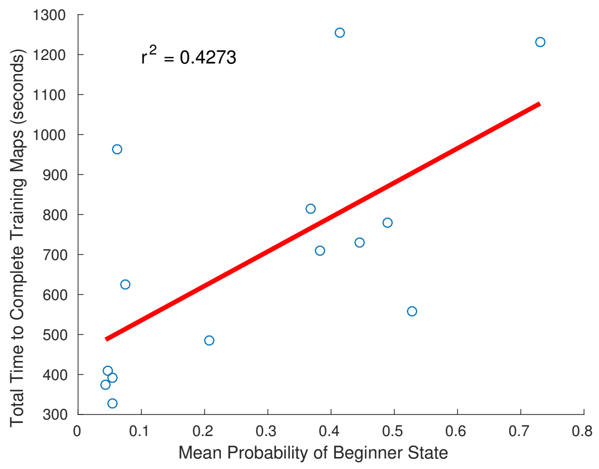


Fig. 5: The total time to complete the training maps versus the mean probability of being a *Beginner*.

B. Test Maps

We pose three hypotheses regarding the validity of using the different controllers for different users. We looked at the number of obstacles hit, the total time to complete the maps, and the total amount of user input, the amount of time the user was pressing one of the WASD keys. The "MostHelp" controller was designed with the intention of helping users avoid obstacles at the cost of slowing the robot more and giving more control to the robot. The "LeastHelp" controller was designed for users more skilled at avoiding obstacles. Since these users need little or no help to avoid obstacles, there is less need to slow them down or give the robot more control. Because the robot's autonomy was presented as the robot "helping", the names Most, More, Some, and LeastHelp were given to show the user how much the robot would be helping them avoid obstacles. However, the robot's increased autonomy also slowed the robot down more and requires more user input to counteract if the robot tries to turn away from obstacles against the user's wishes. While

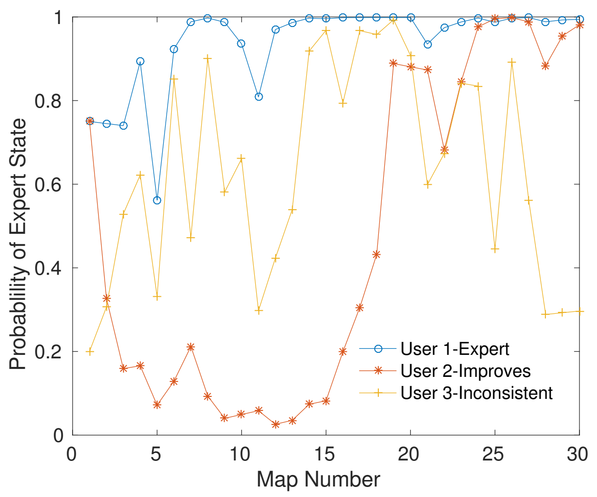


Fig. 6: The POMDP's predicted probability of being at the *Expert* state over the training maps for 3 users.

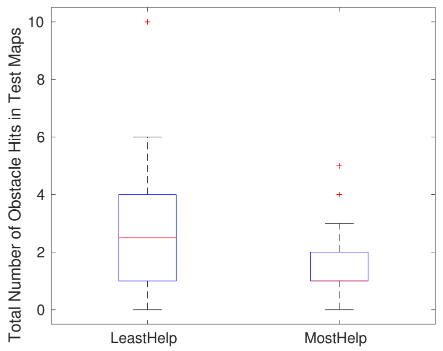
this does help to avoid obstacles, it can also increase the time and amount of user input needed to complete the maps. Three hypothesis were proposed.

Hypothesis 1a *Using the "MostHelp" controller will result in fewer obstacles hit than using the "LeastHelp" controller.*

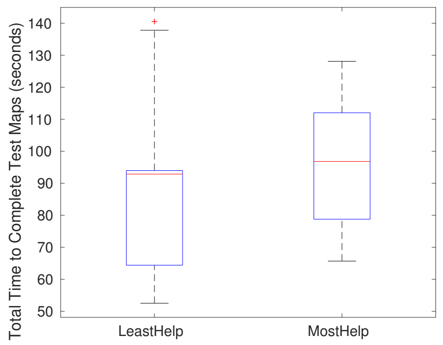
Hypothesis 1b *Using the "LeastHelp" controller will result in shorter time to complete the maps than using the "MostHelp" controller.*

Hypothesis 1c *Using the "LeastHelp" controller will result in less user input to complete the maps than using the "MostHelp" controller.*

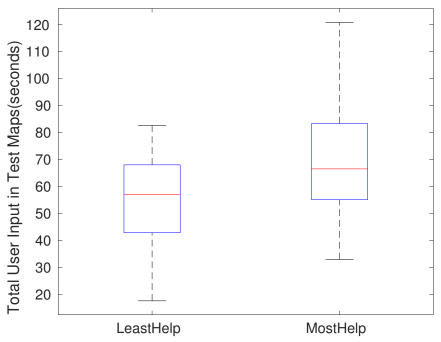
A paired t-test was conducted to compare the users' results from the test maps while using the "LeastHelp" controller and the "MostHelp" controller. We use a p-value<0.05 as a threshold for statistical significance, and a p-value<0.1 as trending towards statistical significance. The results are shown in Fig. 7. There was a statistically significant difference in the number of obstacles hits for "LeastHelp" (M = 3.14, SD = 2.60) and "MostHelp" (M = 1.64,SD = 1.50) conditions; $t(13)$=2.39, p=.033. There was also a statistically significant difference in the total user input for "LeastHelp" (M = 55.68, SD = 18.10) and "MostHelp" (M = 71.97, SD = 25.88) conditions; $t(13)$ = -3.16, p = .008. There was not a statistically significant difference in the total time for "LeastHelp" (M = 90.13, SD = 29.21) and "MostHelp" (M = 97.89, SD = 1.50) conditions; $t(13)$ = -1.80, p = .095. Hypothesis 1a and 1c are supported with statistical significance, and while Hypothesis 1b is not statistically significant, it shows a trend towards significance. These results suggest that while using the "MostHelp" controller, a user will hit less obstacles; however, when using the "LeastHelp" controller the amount of time and user input needed to complete the map decreases. For users who are able to avoid obstacles well on their own, the "LeastHelp" controller is the more fitting option.



(a) Number of Hits (p = .033)

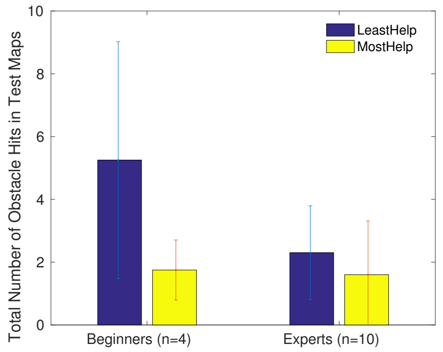


(b) Total Time (p = .095)

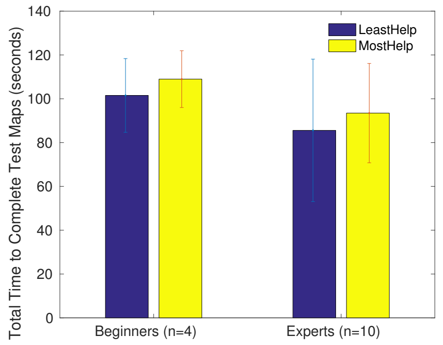


(c) User Input (p = .008)

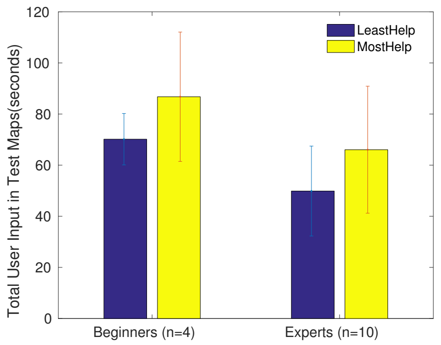
Fig. 7: The performance using the "LeastHelp" and "MostHelp" controllers. There is a statistically significant difference between the number of obstacle hits (7a) and the total user input (7c). The difference in total time is trending towards significance (7b). We can take away from these results is that using the "MostHelp" controller reduces the number of obstacles hit, but using the "LeastHelp" controller reduces the time and the amount of user input needed to complete a map.



(a) Number of Hits (p = .037)



(b) Total Time (p = .966)



(c) Total User Input (p = .972)

Fig. 8: The performance from users grouped into Beginners and Experts. The error bars show the standard deviation. The change in performance from the two controllers is compared for Beginners and Experts. There is a statistical difference in the number of obstacles hits (8a). There is not a statical difference in the total time (8b) and total user input (8c). These results suggest that Beginners receive the most amount of assistance in avoiding obstacles when using the "MostHelp" controller, but the reduction of time and user input is not significantly greater for Beginners and Experts.

We compared the results of users at different skill levels to see if the two controllers made significant improvements for users of higher and lower expertise. We grouped the users into "beginners" and "experts" based on the mean probability of user state of maps 20-30 performed in the first part of the experiment. Users who had reached a mean Expert probability above 0.9 were classified as "experts" (10 users), while those below were classified as "beginners" (4 users). Looking at the Fig. 6, those such as User 1 (high probability of Expert through most of the training maps) and User 2 (showed improvement and had a high probability of Expert at end of training maps) were classified as experts. Those such as User 3 (did not improve greatly) were classified as beginners. We posed three more hypothesis to compare the controllers between the different kinds of users.

Hypothesis 2a *The difference in the number of obstacles hit between using the "MostHelp" and the "LeastHelp" controller will be greater for users classified as beginners than users classified as experts.*

Hypothesis 2b *The difference in the time to complete the maps between using the "MostHelp" and the "LeastHelp" controller will be greater for users classified as beginners than users classified as experts.*

Hypothesis 2c *The difference in the user input to complete the maps between using the "MostHelp" and the "LeastHelp" controller will be greater for users classified as beginners than users classified as experts.*

An unpaired t-test was conducted to compare the difference between using the "LeastHelp" controller and the "MostHelp" controller for the users classified as Beginners and Experts. Fig. 8 shows the differences between the user types. There was a statistically significant difference in the difference of obstacles hits for Beginners (M = 3.50, SD = 3.11) and Experts (M = 0.70, SD = 1.49); $t(12)$ = 2.34, p = .037. There was not a statistically significant difference in the difference of total time for Beginners (M = -7.45, SD = 9.49) and Experts (M = -7.88, SD = 18.58); $t(12)$ = 0.043, p = .966. There was not a statistically significant difference in the difference in user input for Beginners (M = -16.61, SD = 29.49) and Experts (M = -16.17, SD = 15.72); $t(12)$ = -0.037, p = .972. Hypothesis 2a is supported with statistical significance suggesting that for the Beginner users, using the

"MostHelp" controller gives more improvement in terms of the number of obstacles hit. Hypothesis 2b and 2c do not show a statistically significant difference between Beginners and Experts for total map completion time and user input time. What these results suggest is that users who are less skilled receive more assistance from the robot autonomy in avoiding obstacles than a more skilled user would receive. Since Experts do not improve as much as Beginners from the increase in the robot's control, a controller that lets them complete the maps more quickly may be the more appropriate controller to use.

VI. Conclusion and Future Directions

The results from the user trial show that our model can distinguish between users of different skill level. Our results have shown that the users that have lower probabilities of being an *Expert* hit more obstacles and take more time to complete the maps. They also showed that our designed macro-action controllers were able to reduce the number of obstacles hit with the robot's assistance, but this assistance also increased the amount of time to complete the maps and the amount of input the user had to give. Though this was the case for all users, we found that the users with a lower probability of being an *Expert* received more help in avoiding the obstacles than those with a higher probability of being an *Expert*. There was no statistically significant difference between the two types of users when looking at the amount of time and user input, but for *Expert* users, a controller that allows them to complete the maps faster should be chosen since there is little difference in the number of obstacles hit. These results support our idea that using a macro-action POMDP to predict user expertise can be useful when leveraging levels of shared autonomy. Our model has shown promising results so far, and we are working to make further improvements.

In this paper, the only user observations considered were how often they would hit an obstacle. Future work using our POMDP model will observe more complex human actions, such as control stability. We also plan to apply our model to robots in the field. While the user controls the robot, expertise will be learned on the fly, and controllers will be chosen for the users as they either improve or fatigue. Another future goal is to be able to automatically synthesize controllers. We plan to use our ability to observe user actions to train simulated-user bots to act as either beginners or experts. Using these bots, we will synthesize controllers that are able to optimize the performance different users in numerous scenarios. For example, one might want to use a controller to train a beginner to become an expert, but in another case, they may want just optimize the goal without worrying about improving the user's ability to operate the robot.

User expertise makes a difference in how a user performs with a robot. Finding ways to tune the levels of shared autonomy between different users is necessary for improving human-robot teaming, and using the user's expertise is one more way to design robots that are better able to suit the needs of a wide variety of users.

References

[1] L. Milliken and G. A. Hollinger, "Modeling user expertise for choosing levels of shared autonomy," in *Proc. Planning for Human-Robot Interaction Shared Autonomy and Collaborative Robotics Workshop, Robotics: Science and Systems Conference*, 2016.

[2] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[3] S. Nikolaidis, A. Kuznetsov, D. Hsu, and S. Srinivasa, "Formalizing human-robot mutual adaptation via a bounded memory based model," in *Proc. ACM/IEEE International Conference on Human-Robot Interaction*, 2016, pp. 75–82.

[4] A. Dragan, K. Lee, and S. Srinivasa, "Teleoperation with intelligent and customizable interfaces," *Journal of Human-Robot Interaction*, vol. 1, no. 3, pp. 33–79, 2013.

[5] M. Gao, J. Oberländer, T. Schamm, and J. Zöllner, "Contextual task-aware shared autonomy for assistive mobile robot teleoperation," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3311–3318.

[6] A. Franchi, C. Secchi, M. Ryll, H. Bulthoff, and P. Giordano, "Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 57–68, 2012.

[7] T. Somers and G. Hollinger, "Human–robot planning and learning for marine data collection," *Autonomous Robots*, vol. 40, no. 7, pp. 1123–1137, 2016.

[8] S. Anderson, S. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2012, pp. 383–388.

[9] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, "Common metrics for human-robot interaction," in *Proc. ACM SIGCHI/SIGART conference on Human-Robot Interaction*, 2006, pp. 33–40.

[10] J. Beer, A. Fisk, and W. Rogers, "Toward a framework for levels of robot autonomy in human-robot interaction," *Journal of Human-Robot Interaction*, vol. 3, no. 2, p. 74, 2014.

[11] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," in *Robotics: Science and Systems*, Rome, Italy, 2015.

[12] L. Takayama, E. Marder-Eppstein, H. Harris, and J. M. Beer, "Assisted driving of a mobile remote presence system: System design and controlled user evaluation," in *Proc. IEEE Conference on Robotics and Automation*, 2011, pp. 1883–1889.

[13] K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Autonomous Robots*, vol. 35, no. 4, pp. 241–254, 2013.

[14] P. Pinheiro, E. Cardozo, and C. Pinheiro, "Anticipative shared control for robotic wheelchairs used by people with disabilities," in *Proc. IEEE International Conference on Autonomous Robot Systems and Competitions*, 2015, pp. 91–96.

[15] E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, H. V. Brussel, and M. Nuttin, "User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving," *Autonomous Robots*, vol. 24, no. 2, pp. 193–211, 2008.

[16] S. Javdani, J. A. Bagnell, and S. Srinivasa, "Minimizing user cost for shared autonomy," in *Proc. ACM/IEEE International Conference on Human-Robot Interaction*, 2016, pp. 621–622.

[17] S. Nikolaidis, P. Lasota, R. Ramakrishnan, and J. Shah, "Improved human–robot team performance through cross-training, an approach inspired by human team training practices," *The International Journal of Robotics Research*, vol. 34, no. 14, pp. 1711–1730, 2015.

[18] "Approximate pomdp planning software," http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl, 2014.

[19] P. Shinners, "Pygame," http://pygame.org/, 2011.

[20] C. Amato, G. Konidaris, A. Anders, G. Cruz, J. P. How, and L. P. Kaelbling, "Policy search for multi-robot coordination under uncertainty," in *Robotics: Science and Systems Conference*, 2015.

[21] J. W. Crandall and M. A. Goodrich, "Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation," in *Proc. IEEE/RSJ International Conference Intelligent Robotics and Systems*, vol. 2, 2002, pp. 1290–1295.