2024 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
July 15-19, 2024. Boston, MA, USA

Assistive Control of Robot Arms via Adaptive Shared Autonomy

Umur Atan, Varun R. Bharadwaj and Chao Jiang

***Abstract*— Shared autonomy is a robot control approach that assists human users to achieve their intended goal while leveraging the precision and efficiency of robot autonomy. In shared autonomy, user input and autonomous assistance are combined to effectively control robots without requiring users to provide direct and precise control inputs. A persistent question in shared autonomy is how to determine the arbitration between user input and autonomous algorithm. Due to variability in users' desired amount of assistance, it is imperative to develop user-centric algorithms that provide customized and adaptive assistance by considering users' preference, physical capability, and expertise. In this paper, we propose a shared autonomy method that factors in both users' task performance and level of expertise to adaptively adjust the amount of assistance at runtime. We validated our method in an assistive control problem where human users teleoperate a robotic arm to perform object reaching and grasping tasks in a simulated environment. The results show that our method assisted the users to achieve a higher efficiency in accomplishing the object reaching and grasping tasks compared to direct teleoperation and two baseline arbitration methods that only consider task-related metrics.**

I. Introduction

Robotic teleoperation plays a crucial role in various domains such as assistive robots, industrial manufacturing, surgical and rescue robots [1], [2]. Teleoperated robot arms in industrial settings enable humans to remotely control machinery, thereby mitigating the potential for harm in hazardous conditions. Surgeons may control medical instruments from a distance with enhanced dexterity and stability to achieve improved surgical outcomes. However, direct teleoperation for robots is a demanding task, which requires users to have comprehensive understanding of the robots' capabilities and motion. Even with proper training and familiarity, users may struggle to effectively control robots due to the limitations of control interface [3]. Inexperienced operators, in particular, may have difficulties in correctly aligning robot arms and synchronizing exact motions, and potentially produce unsafe control inputs [4]. Therefore, it is crucial to address these problems in order to improve user experience and enable effortless interaction with robot arms for users with different levels of expertise.

Shared autonomy [3] has been proposed as an appealing approach to address the challenges of direct teleoperation by augmenting users' input with autonomous assistance [5]. Shared autonomy allows users to effectively teleoperate robots without providing direct and precise control input. This way, users can focus on high-level control towards their intended task objective while the robots compute and execute low-level actions to collaboratively perform the task. Nevertheless, how to arbitrate user and autonomous algorithm inputs to achieve such human-robot collaboration continues to be a point of discussion. Traditional methods compute arbitration primarily based on task-related performance metrics such as efficiency and completion time. However, users' desired amount of assistance typically varies based on user-related factors such as preference, physical capability, and level of expertise [6]. Moreover, previous studies found that users prefer to retain as much control over autonomous algorithms as possible [4]. Therefore, for users to reap the advantages of autonomous assistance in excessively demanding tasks, it is imperative to develop user-centric algorithms that provide customized and adaptive assistance by considering user-related factors.

In this paper, we study an assistive control problem where human users teleoperate a robotic arm to perform object reaching and grasping tasks. We propose a novel approach to shared autonomy that modifies control sharing archetypes to provide customized and adaptive assistance at runtime. Our approach takes into account both users' dynamic performance with respect to task-related metrics and level of expertise to adaptively adjust the amount of assistance. We validated the proposed method in simulation experiments and the results show that our method achieved a higher task efficiency compared to direct teleoperation and two baseline arbitration methods that only consider task-related metrics.

The rest of the paper is organized as follows. Section II provides an overview of shared autonomy approaches. Section III describes our problem formulation. Section IV presents the assistive robotic arm control system and our proposed adaptive control blending method. Experiment results and discussion are provided in Section V. Finally, we conclude our work in Section VI.

II. Related Work

A. Shared Autonomy

Shared autonomy has been extensively studied in the context of robotic teleoperation and assistive robotics [7]. The key idea of shared autonomy is to combine user input with autonomous assistance to achieve users' intended goals. To provide autonomous assistance that aligns with the user's intent, early shared autonomy approaches either assume that the user's goals are known or infer the user's goals from their actions by leveraging prior knowledge of environmental dynamics, goal representations, and user decision-making

*This work was supported in part by the National Science Foundation under grant no. 2024813.

[1] Umur Atan, Varun R. Bharadwaj and Chao Jiang are with the Department of Electrical Engineering and Computer Science, University of Wyoming, 1000 E University Ave, Laramie, Wyoming emails: {uatan,vbharadw,cjiang1}@uwyo.edu

policies towards their goals [8], [9]. However, those approaches usually fall short in complex environments and tasks where acquiring prior knowledge of environments and users become excessively challenging. As a result, learning-based approaches [10] have been proposed to learn neural network models that decipher user preference or policy and generate assistance control action given the current observation of system state. The final robot control is determined by blending the user and assistance control inputs based on preselected arbitration strategies (e.g., linear combination). The drawback of the aforementioned methods is the lack of dynamic blending that adaptively adjusts the level of assistance based on user performance or needs.

Various methods [9], [11], [12] for dynamic blending of user and autonomous algorithm inputs have been developed using the notion of autonomy levels [13]. Dynamic blending methods adaptively seek optimal balance between user and robot autonomy during task operation, aiming to achieve efficient human-robot collaboration and successful task fulfillment. Early dynamic blending methods adjust the arbitration between user and robot autonomy primarily based on task-related performance metrics. However, users' desired amount of assistance typically varies due to user-related factors such as preference, physical capability, and level of expertise [6]. Therefore, user-centric methods have been proposed to offer customization of autonomous assistance according to user preferences and needs [4]. Compared to task-centric methods, user-centric design of autonomy arbitration improves user-related metrics such as independence and satisfaction while achieving desired task performance [14]. Most of the dynamic blending methods mentioned above use discretized levels of arbitration, hence may not produce optimal blending of user and robot inputs. In this paper, we adopt the idea of sliding autonomy [15] and propose a method that enables continuous adjustment of robot autonomy level between direct teleoperation and full autonomy.

B. Expertise Factor

Users' level of expertise has been considered as a key factor in tailoring the response of autonomous robots to human users in shared autonomy. In the context of robotic arm teleoperation, level of expertise can be described by user competence identifiers such as accuracy of reaching goal positions, avoiding collisions with objects, and frequency of user input commands. A user with low input accuracy is prone to errors, potentially leading to subpar performance or accidents [16]. In such cases, the expertise factor adjusts to a lower value, indicating a need for elevated level of autonomous assistance. Input frequency, defined as the number of times a user provides control signals per unit of time [17], is a measure of the user's active participation. A heightened input frequency, irrespective of the correctness of the user's input, typically signifies the user's determination to guide the robot in a preferred direction. If the robot fails to comply, the user may resist autonomous assistance and pursue their intended actions in subsequent steps. Inspired by these findings from prior works, we developed a method that quantifies users' expertise from user attributes such as correctness of input [16] and frequency of input [17] to adaptively adjust the level of assistance at runtime.

III. Problem Statement

We consider a scenario where a human user's task is to teleoperate a robotic arm to reach and grasp objects placed on a table. Each object's position is regarded as a goal position denoted as $\boldsymbol{g}^* \in \mathbb{R}^3$. Let $\mathcal{S} \subset \mathbb{R}^3$ be the state space that includes all reachable positions of the robot's end effector and $\mathcal{A} = \{+\boldsymbol{x}, +\boldsymbol{y}, +\boldsymbol{z}, -\boldsymbol{x}, -\boldsymbol{y}, -\boldsymbol{z}\}$ be the action space for moving the robot's end effector. An action from $\mathcal{A}$ may move the end effector in either positive or negative x, y, or z directions. Users can also open/close the gripper on the robot's end effector to grasp the objects. When an action $\boldsymbol{a}_t \in \mathcal{A}$ is taken in state $\boldsymbol{s}_t \in \mathcal{S}$, the end effector transitions to a new position $\boldsymbol{s}_{t+1} \in \mathcal{S}$ according to the state transition function $\mathcal{T} : \boldsymbol{s}_{t+1} \leftarrow f(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The user's and robot's action inputs are defined as $\boldsymbol{a}_H \in \mathcal{A}$ and $\boldsymbol{a}_R \in \mathcal{A}$, respectively. We assume that the users' intended goal positions and the position of the robot's end effector are known to the robot for computing robot action $\boldsymbol{a}_R \in \mathcal{A}$.

At each time step, the user's input $\boldsymbol{a}_H$ and robot's input $\boldsymbol{a}_R$ are blended to create a blended action, denoted as $\boldsymbol{a}_E$, using the following control arbitration strategy [3], [10], [18].

$$\boldsymbol{a}_E = \beta \cdot \boldsymbol{a}_H + (1 - \beta) \cdot \boldsymbol{a}_R \tag{1}$$

where β is the arbitration parameter. The objective in this work is to develop an adaptive control arbitration method that dynamically finds the optimal β for computing the blended action $\boldsymbol{a}_E$ to reach and successfully grasp the object with minimal time steps.

IV. Methodology

A. Overview

An overview of our assistive robotic arm control system is shown in Fig. 1. A human user uses a keyboard to provide action input $\boldsymbol{a}_H$ that moves the robot's end effector to the user's intended directions based on their observation of the robot. A deep Q-network (DQN) model is trained for computing action $\boldsymbol{a}_R$ given the state of the robot's end effector. The DQN model acts as an agent that is capable of controlling the robot to complete the task in a fully autonomous manner. Our method uses the simulated annealing (SA) algorithm [19] to search for an intermediate control arbitration parameter, denoted as α, such that $\boldsymbol{a}_H$ and $\boldsymbol{a}_R$ are blended to yield an optimal action towards reaching the goal. Meanwhile, the user's expertise factor e_x is evaluated based on the user's input. The final arbitration parameter β is determined by combining α and e_x and a blended action $\boldsymbol{a}_E$ is computed via Eqn. (1) to control the robot.

B. DQN for Autonomous Control

We train a DQN model that acts as an autonomous agent for robot control using the deep Q-learning algorithm [20] in a simulation environment. The agent aims to control the

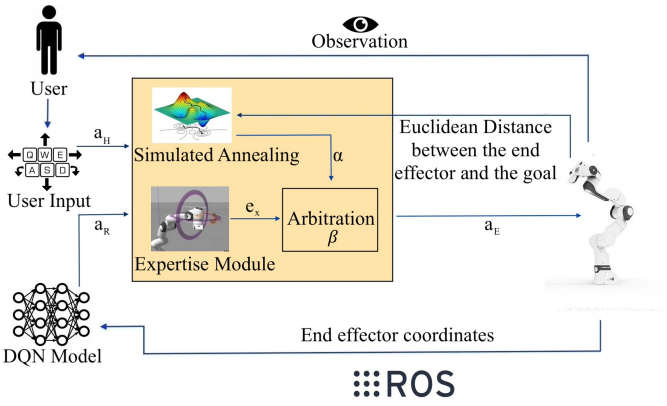

Fig. 1: An overview of our method for assisting human users in robotic arm teleoperation via shared autonomy.

Algorithm 1 Expertise Factor

Start timer τ when the robot transitions to $\boldsymbol{s}_t$
Terminate timer τ when the user input is received
Initialize expertise factor e_x
Get the user action euclidean distance $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\|$ using $\boldsymbol{a}_H$
if $\tau < 0.1$ s and $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| < \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ **then**
    Set $e_x = 1$
else if $\tau > 0.1$ s and $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| < \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ **then**
    Set $e_x = 0.75$
else if $\tau < 0.1$ s and $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| \geq \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ **then**
    Set $e_x = 0.5$
else if $\tau > 0.1$ s and $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| \geq \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ **then**
    Set $e_x = 0.25$
end if

robot to approach a given goal position that is randomized in each training episode. To shape a control policy that takes the shortest path, the reward function is designed as the negative squared error between the end effector's current position and the goal position, i.e., $r_t = -\|\boldsymbol{s}_t - \boldsymbol{g}^*\|^2$. After training, the autonomous agent employs a greedy policy that selects the optimal action with the maximum Q value at each time step until reaching the goal.

C. Action Blending

The action arbitration parameter β in Eqn. (1) plays a pivotal role in determining the combination of user and agent inputs. β is viewed as a dynamic parameter that influences the control authority granted to the user and autonomous agent. Our method computes β as the sum of two contributing factors: 1) an expertise factor e_x that evaluates the user's level of expertise and 2) an intermediate arbitration parameter α that is optimized using the simulated annealing algorithm.

1) Expertise Factor e_x: The expertise factor e_x is determined based on two expertise indicators: the correctness and frequency of user input [16], [17]. The user action $\boldsymbol{a}_{H,t}$ at time step t is considered correct if the Euclidean distance between the end effector and goal decreases by taking the action $\boldsymbol{a}_{H,t}$, i.e., $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| < \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$, where $\hat{\boldsymbol{s}}_{t+1} \leftarrow f(\boldsymbol{s}_t, \boldsymbol{a}_{H,t})$. Otherwise, the action makes no progress towards reaching the goal if $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| \geq \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$. The frequency of user input is evaluated as how quickly a user gives input at each time step. To measure the input frequency, a timer denoted as τ is initiated as soon as the robot transitions to state $\boldsymbol{s}_t$ at each time step t and is terminated when the user input for this time step is received. The threshold is set to 0.1 seconds which enables rapid detection of queued commands and reflects the proficiency of the user.

We consider four different levels of expertise for e_x, namely, high, medium, low, and very low expertise. Our method to determine the expertise factor e_x using the correctness and frequency of user action is summarized in Algorithm 1.

- High expertise level ($e_x = 1$) is assigned to users who take correct actions $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| < \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ and respond fast ($\tau < 0.1$ s).
- Medium expertise level ($e_x = 0.75$) is assigned to users who take correct actions $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| < \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ but do not respond fast enough ($\tau > 0.1$ s).
- Low expertise level ($e_x = 0.5$) is assigned to users who fail to take correct actions $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| \geq \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ but respond fast ($\tau < 0.1$ s).
- Very low expertise level ($e_x = 0.25$) is assigned to users who fail to take correct actions $\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{g}^*\| \geq \|\boldsymbol{s}_t - \boldsymbol{g}^*\|$ and do not respond fast enough ($\tau > 0.1$ s).

2) Intermediate Arbitration α via Simulated Annealing: We employ simulated annealing [19] to dynamically search for an intermediate arbitration parameter α, which provides adaptive and seamless adjustment of shared autonomy compared to methods using discretized levels of arbitration [3]. Simulated annealing is an optimization method based on stochastic search. Compared to gradient descent methods, it is less prone to local optima and thus could be more effective in finding (globally) optimal solutions for complex, non-convex problems.

To find the optimal α, the simulated annealing algorithm starts with a random value of α and iteratively optimizes α based on a cost function and a temperature parameter. In this work, the cost function is chosen as $c(\alpha) = \|\boldsymbol{s}_{t+1} - \boldsymbol{g}^*\|$, which is the Euclidean distance between the end effector and goal position. The temperature parameter was used to modulate the acceptance of suboptimal solutions using the Metropolis criterion that probabilistically determines the acceptance of proposed solutions. For a given value of α, a blended action $\boldsymbol{a}_{E,t}$ is calculated using Eqn. (1) and the resulting $\boldsymbol{s}_{t+1}$ is given by $\boldsymbol{s}_{t+1} \leftarrow f(\boldsymbol{s}_t, \boldsymbol{a}_{E,t})$ for evaluating $c(\alpha)$. At each iteration i, a random perturbation is applied to the current α_i to generate a new value α_i' and the change in the cost function, $\Delta c = c(\alpha_i') - c(\alpha_i)$, is calculated. If Δc is negative (indicating an improvement), α_i' is then accepted probabilistically based on the Metropolis criterion and the value of α_i is updated to α_i'. If $\Delta c \geq 0$, α is not updated. In

Algorithm 2 Action blending using simulated annealing and expertise factor for each time step

Input: Human action $\boldsymbol{a}_H$.
Robot action $\boldsymbol{a}_R$.
Output: Blended action $\boldsymbol{a}_E$.
Obtain expertise factor e_x from Algorithm 1
// Simulated annealing
Initialize *temperature*
Initialize a random α
Set desired *stepsize*
for $i <$ max iteration **do**
  $\alpha_i' = \alpha_i + rand(0,1) \cdot stepsize$
  Calculate the change in the cost function: $\Delta c = c(\alpha_i', e_x) - c(\alpha, e_x)$
  Update $temperature$
  $metropolis = \exp(-\Delta c / temperature)$
  if $\Delta c < 0$, **or** $rand(0,1) < metropolis$
    $\alpha_{i+1} = \alpha_i'$
  else
    $\alpha_{i+1} = \alpha_i$
end for
// Action blending
Calculate $\beta = (\alpha + e_x)/2$
Calculate blended action $\boldsymbol{a}_E$ using Eqn. (1)
Execute action $\boldsymbol{a}_E$ on the robot.

our experiments, the simulated annealing algorithm required fewer than 0.03 seconds to compute an optimal output for each time step.

3) Combining e_x and α: The final arbitration parameter β in Eqn. (1) is obtained as the sum of the expertise factor e_x and the intermediate arbitration parameter α. The value of β is normalized using the following equation so that $\beta \in [0, 1]$.

$$\beta = \frac{\alpha + e_x}{2} \tag{2}$$

The expertise factor e_x acts as an offset to the intermediate arbitration parameter α to adaptively adjust the desired amount of assistance tailored for individual users. Our method for determining the control arbitration between user and autonomous agent is summarized in Algorithm 2.

V. Experiment and Results

A. Experiment Setup

The experiments were conducted in a Gazebo simulation environment with a Franka Panda robotic arm as shown in Fig. 2. The simulator receives blended action $\boldsymbol{a}_E$ for the end effector and computes joint motor control to move the end effector in three-dimensional space or open/close the end effector claws. The joint motor controls were computed using an inverse kinematics algorithm provided in the robotic systems toolbox [21]. There were four different cuboids as goal objects, each with a distinct color. Users were asked to reach and grasp the goal objects in a random order using a keyboard to provide end effector movement actions.

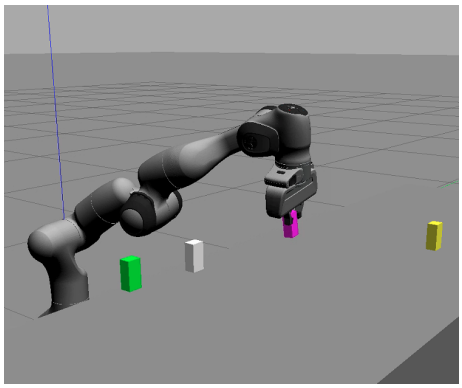

Fig. 2: Simulation environment in Gazebo with a Franka Panda robotic arm and goal objects in different colors.

We invited 6 human participants to perform the experiments. The 6 participants were evenly split into an expert group and a beginner group. The three participants in the expert group were allowed to practice with the simulator to gain proficiency in teleoperating the robot before the experiments. The three participants in the beginner group were asked to perform the experiments without practicing. The two groups thus represent expert level and beginner level of proficiency, respectively, for the teleopration task. Each participant was asked to perform 2 simulation runs for each method with randomized goal positions, resulting in 8 simulation runs for each participant. A total of 48 simulation runs were performed with 6 participants.

B. Evaluation Metrics and Baselines

1) Evaluation Metrics: Our proposed shared autonomy method was empirically validated and evaluated with respect to three performance metrics: total completion time, total number of steps taken by users, and user experience.

- *Total completion time* represents the duration from a user starting the simulation to reaching and grasping the goal object. This metric is used to assess the efficiency of the teleoperation methods.
- *Total number of steps* amounts to the number of robot actions executed from the beginning of a simulation run to reaching and grasping the goal. This metric also captures the trajectory length between the initial and final positions of end effector.
- *User experience* is an objective metric that assesses users' perceived support from the algorithms and comfort while they were teleoprating the robot.

2) Baseline Methods: Our method was compared with three baselines methods: 1) direct teleoperation, 2) simulated annealing, and 3) discretized arbitration.

- *Direct teleoperation:* This baseline represents the traditional teleoperation approach, where a user manually controls the robot using a keyboard without any assistance from autonomous algorithms. The robot simply follows the direct commands provided by the user.
- *Simulated annealing:* This baseline uses the simulated annealing algorithm to calculate the arbitration param-

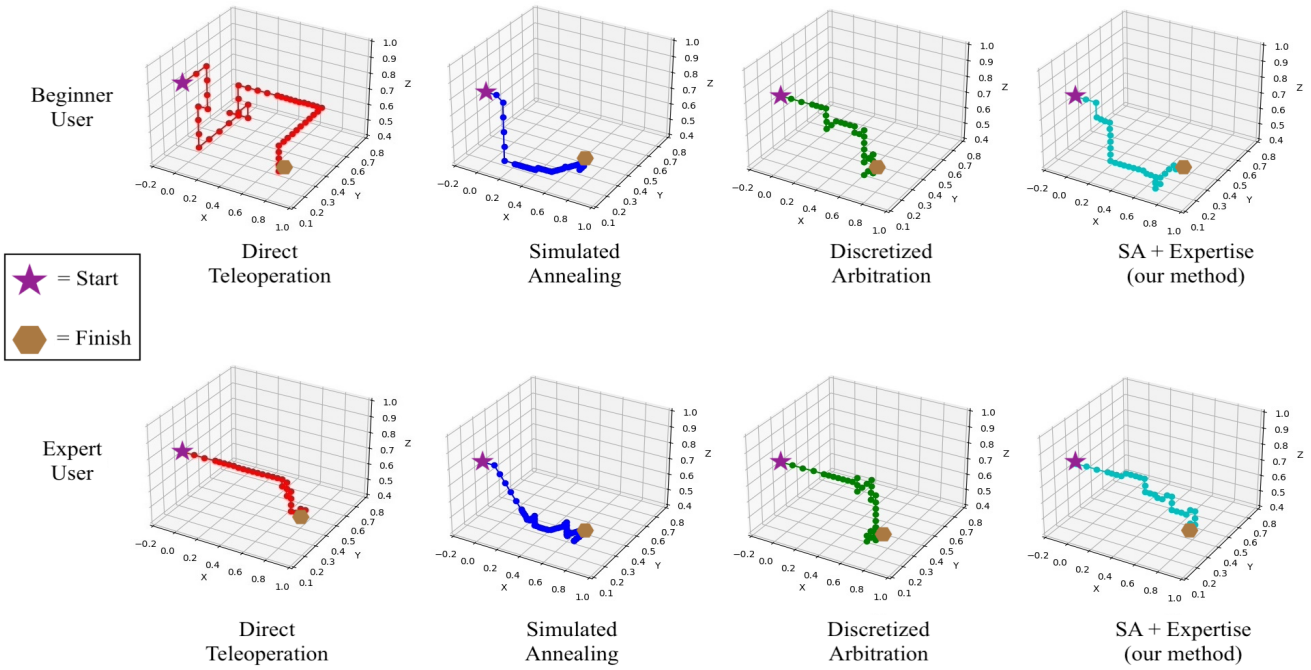

Fig. 3: Trajectories of robot end effector obtained by a beginner user and an expert user using our method and three baseline methods. Top row: trajectories obtained by a beginner user. Bottom row: trajectories obtained by an expert user.

eter β in Eqn. (1). This baseline does not consider the expertise factor e_x as in our proposed method and determines β only based on the correctness of the blended action a_E with respect to the task objective.

- *Discretized arbitration:* This baseline dynamically sets β in Eqn. (1) to 0 or 1 based on the correctness of user input $\boldsymbol{a}_H$ with respect to the task objective. Specifically, $\beta = 1$ if the Euclidean distance between the end effector and goal decreases by executing the user input $\boldsymbol{a}_H$, and $\beta = 0$ otherwise. This dynamic adjustment of arbitration tends to give full control authority to proficient users, while allowing the system to take full control authority when the user inputs are less effective.

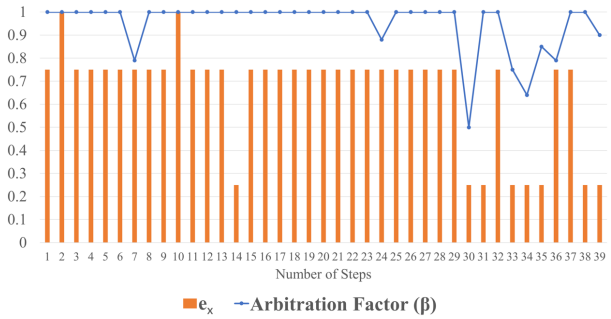

Fig. 4: The change of arbitration factor β and expertise factor e_x over simulation steps.

C. Experiment Results

Fig. 3 shows the trajectories of robot end effector for simulations performed by a beginner user and an expert user using our method and the three baseline methods. One can see that the beginner user took a long trajectory to reach the goal using direct teleoperation, while the three action blending methods assisted the beginner user to obtain significantly improved trajectories. For the expert user, the trajectory for direct teleoperation was shorter compared to the beginner user. However, the action blending methods using simulated annealing and discretized arbitration obtained less optimal trajectories than direct teleoperation. This is because the two methods did not effectively provide desired levels of assistance to the expert user, which caused the user's confusion and resistance against the assistance provided by the autonomous algorithms. In contrast, our method obtained similar trajectory with direct teleoperation for the expert user by accounting for the user's expertise level and adaptively adjusting the level of assistance.

Fig. 4 shows the variation of the arbitration factor β and the expertise factor e_x over steps for a simulation run performed by an expert user. One can see that on average the user exhibited medium or high expertise level from step 1 to step 29, our algorithm produced a high value of β so that the user retained near full control. Starting from step 30, the expertise factor e_x drops frequently as the user took incorrect and slow actions when the robot arm approached the goal. Concurrently, the arbitration factor β decreases, indicating a transition in control paradigm towards the robot autonomy.

1) Completion Time and Total Steps: To compare the efficiency between different methods, we measured the total completion time and total steps taken by the users for each simulation run. Fig. 5 shows the statistical results of total completion time obtained by beginner and expert users across

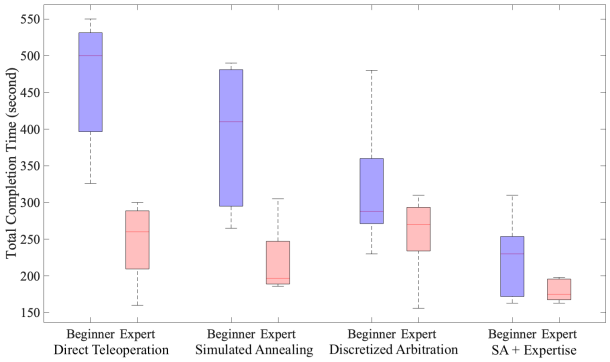

Fig. 5: Statistical results of completion time obtained by beginner and expert users for each method.

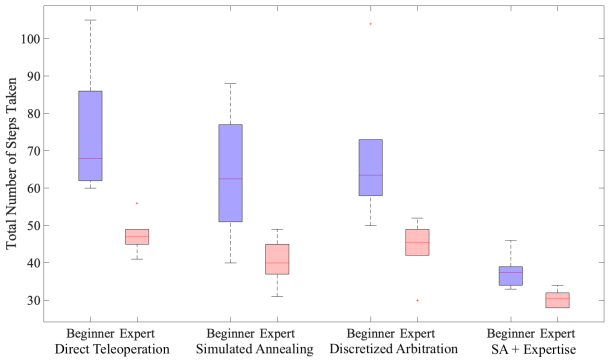

Fig. 6: Statistical results of total steps taken by beginner and expert users for each method.

all methods, where blue boxes represent beginner user group and red boxes represent expert user group. For each method, there were 6 simulation runs for each user group as each user performed 2 simulation runs.

We can see that there was a significant difference between beginner and expert users using direct teleoperation. With simulated annealing, the completion time was improved for both beginner and expert groups while the improvement for the beginner group was more significant than that for the expert group. Discretized arbitration improved the performance of the beginner group even more, compared to simulated annealing. This is attributed to the fact that discretized arbitration dynamically grants full control authority to the autonomous agent if the user provides a nonoptimal action. While this could be an effective assistive method for beginner users, expert users may find it intrusive and confusing and thus tend to resist against the autonomous assistance. As a result, the performance of expert users deteriorates when discretized arbitration was used, which can been seen from Fig. 5. Our method that combines simulated annealing and expert factor provided most effective assistance for both beginner and expert users: beginner users were able to complete the task with a median completion time of 240 s, in contrast to a median of 500 s using direct teleoperation; for expert users, the median completion time was improved from 260 s with direct teleoperation to 170 s with our method, along with a reduced variance. Fig. 6 shows the statistical results of total number of steps taken by beginner and expert users across all methods. Similar to the observations from Fig. 5, our method achieved best performance compared to other three baseline methods.

Overall, the results of performance comparison suggest that the our method provides the most effective assistance for both beginner and expert users, and it could substantially reduce the amount of time and steps for the users to complete the task.

2) Adaptivity to Users' Level of Expertise: To evaluate the adaptivity of the the three action blending methods to users' level of expertise, we recorded the number of time steps for which the algorithms provided assistance (i.e., $\beta \neq 1$) in each simulation run. Then, for each simulation run, an

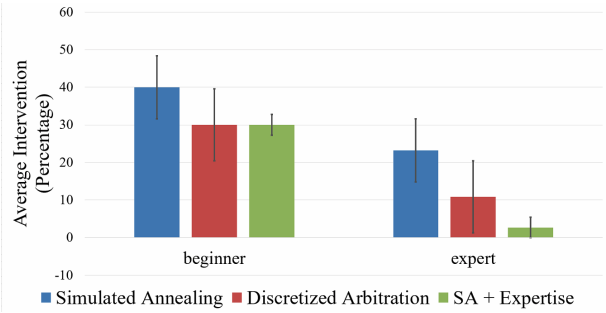

Fig. 7: Average percentage of intervention by different shared autonomy methods for beginner and expert users.

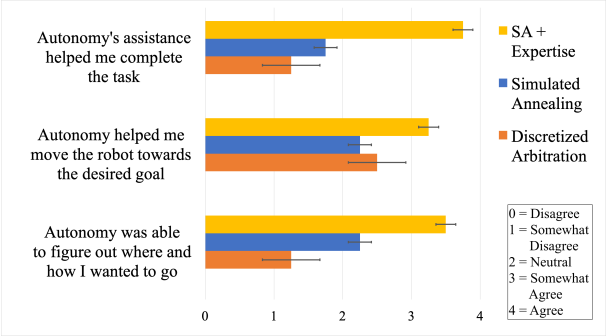

Fig. 8: Users experience questionnaire and feedback.

average intervention was calculated as the number of time steps for which $\beta \neq 1$ over all time steps the user took to complete the task.

Fig. 7 shows the statistical results of average intervention for beginner and expert users using simulated annealing, discretized arbitration, and our method. It can be seen that all three action blending methods provided different amounts of assistance between beginner users and expert users as beginner users typically benefited from more assistance for completing the task. However, compared to the other two methods, our method delivered the least amount of assistance for expert users, hence avoided unnecessary interventions and allowed skilled users to leverage their abilities to efficiently complete the task. The amount of assistance was adaptively adjusted using our method, which ensures that expert control

remains unhindered while necessary tweaks of control are provided.

3) User Experience: We evaluated the user experience for different teleoperation methods. Specifically, we collected subjective feedback from all participants regarding their perceived support from the algorithms and how comfortable they felt while teleoperating the robot with different methods. Each participant was given a questionnaire where three questions were designed to quantify their feedback regarding their experience on a scale of 0 to 4.

The user experience evaluation shown in Fig. 8 reveals that the participants expressed significantly higher satisfaction with our method regarding the algorithm's support in completing tasks, achieving goals, and the algorithm's understanding of their intended actions. The positive feedback on our method can be attributed to the user-centric design that accounts for the variability in users' preference, expertise and provides adaptive assistance accordingly.

VI. Conclusion

In this paper, we present an adaptive shared autonomous control archetype that assists human users to perform object reaching and grasping tasks via teleoperation. Our method incorporates simulated annealing and users' level of expertise to dynamically and adaptively optimize the blending between human and robot actions. The experiment results show that our method provided more effective assistance for users to accomplish the teleoperation task with lower completion times and fewer steps compared to three baseline methods including direct teleoperation and two action blending methods that only accounts for task-related performance metrics. User survey signified their preference for our method over the baseline methods.

Our study provides a design of user-centric shared autonomy as well as empirical findings that suggest the benefits of considering human expertise in shared autonomous systems. Our findings posit that incorporating human factors such as preference, physical capability, and expertise in shared autonomy holds immense promise for enhancing the capabilities of contemporary shared autonomous systems to offer seamless and effective collaboration between humans and robots. Our future work will exploit probabilistic inference for unknown user objectives and develop nonlinear action blending methods. We will also validate our method with real robot experiments.

Acknowledgment

The authors would like to thank the participants for helping with the experiments.

References

[1] V. Alonso and P. De La Puente, "System transparency in shared autonomy: A mini review," *Frontiers in Neurorobotics*, vol. 12, p. 83, 2018.

[2] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[3] S. Reddy, A. D. Dragan, and S. Levine, "Shared autonomy via deep reinforcement learning," *Proceedings of Robotics: Science and Systems*, 2018.

[4] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 247–254, 2016.

[5] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, and B. Siciliano, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021.

[6] A. Erdogan and B. D. Argall, "The effect of robotic wheelchair control paradigm and interface on user performance, effort and preference: an experimental assessment," *Robotics and Autonomous Systems*, vol. 94, pp. 282–297, 2017.

[7] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 2018.

[8] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," *Proceedings of Robotics: Science and Systems*, 2015.

[9] S. Nikolaidis, Y. X. Zhu, D. Hsu, and S. Srinivasa, "Human-robot mutual adaptation in shared autonomy," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 294–302.

[10] H. J. Jeon, D. P. Losey, and D. Sadigh, "Shared autonomy with learned latent actions," *Proceedings of Robotics: Science and System*, 2020.

[11] S. Jain and B. Argall, "Recursive bayesian human intent recognition in shared-control robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3905–3912.

[12] A. Jonnavittula and D. P. Losey, "Learning to share autonomy across repeated interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 1851–1858.

[13] J. M. Beer, A. D. Fisk, and W. A. 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.

[14] A. Romay, S. Kohlbrecher, A. Stumpf, O. von Stryk, S. Maniatopoulos, H. Kress-Gazit, P. Schillinger, and D. C. Conner, "Collaborative autonomy between high-level behaviors and human operators for remote manipulation tasks using different humanoid robots," *Journal of Field Robotics*, vol. 34, no. 2, pp. 333–358, 2017.

[15] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated multiagent teams and sliding autonomy for large-scale assembly," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1425–1444, 2006.

[16] T. Fong, C. Thorpe, and C. Baur, "Collaboration, dialogue and human-robot interaction," in *Proceedings of the 10th International Symposium of Robotics Research*, 2001.

[17] M. N. Javaremi and B. D. Argall, "Characterization of assistive robot arm teleoperation: A preliminary study to inform shared control," *arXiv preprint arXiv:2008.00109*, 2020.

[18] D. P. Losey, H. J. Jeon, M. Li, K. Srinivasan, A. Mandlekar, A. Garg, J. Bohg, and D. Sadigh, "Learning latent actions to control assistive robots," *Autonomous Robots*, vol. 46, no. 1, pp. 115–147, 2022.

[19] P. J. Van Laarhoven, E. H. Aarts, P. J. van Laarhoven, and E. H. Aarts, *Simulated annealing*. Springer, 1987.

[20] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3389–3396.

[21] J. Haviland and P. Corke. Robotics toolbox for python. [Online]. Available: https://petercorke.github.io/robotics-toolbox-python/