Coordinated Multi-Robot Shared Autonomy Based on Scheduling and Demonstrations

Michael Hagenow,[1] Emmanuel Senft,[2] Nitzan Orr,[3] Robert Radwin,[4]
Michael Gleicher,[3] Bilge Mutlu,[3] Dylan P. Losey,[5] Michael Zinn[1]

Abstract—Shared autonomy methods, where a human operator and a robot arm work together, have enabled robots to complete a range of complex and highly variable tasks. Existing work primarily focuses on one human sharing autonomy with a *single* robot. By contrast, in this paper we present an approach for *multi-robot shared autonomy* that enables one operator to provide real-time corrections across two coordinated robots completing the same task in parallel. Sharing autonomy with multiple robots presents fundamental challenges. The human can only correct one robot at a time, and without coordination, the human may be left idle for long periods of time. Accordingly, we develop an approach that aligns the robot's learned motions to best utilize the human's expertise. Our key idea is to leverage *Learning from Demonstration (LfD)* and *time warping* to schedule the motions of the robots based on when they may require assistance. Our method uses variability in operator demonstrations to identify the types of corrections an operator might apply during shared autonomy, leverages flexibility in how quickly the task was performed in demonstrations to aid in scheduling, and iteratively estimates the likelihood of when corrections may be needed to ensure that only one robot is completing an action requiring assistance. Through a preliminary simulated study, we show that our method can decrease the overall time spent sanding by iteratively estimating the times when each robot could need assistance and generating an optimized schedule that allows the operator to provide corrections to each robot during these times. See video of our experiments here: https://youtu.be/6ayZwsJNz_A

I. INTRODUCTION

Human-robot teaming is a promising alternative for tasks where robust automation is infeasible due to high complexity and variability. In many cases, human-robot teaming is achieved through *shared autonomy*, where a human operator and a robot policy share command over the physical robot platform and leverage their respective strengths [16, 13]. For example, the expert operator can offload a task's physical

Fig. 1. Our method enables *multi-robot shared autonomy* by leveraging expert demonstrations and coordinating robots around times where they may need assistance from the operator. *Top:* An operator (task expert) provides a set of demonstrations to inform the robot task model, task variability, and flexibility in scheduling. *Bottom:* The operator provides corrections to two robots completing the same task. The robots are scheduled such that only one robot (*e.g.*, the robot with the red indicator) could require corrections at any given time (*i.e.*, the other robot has high confidence in its action). The confidence is estimated from demonstration variability and empirical correction data.

burden to the robot and use their own task knowledge and superior sensing to make corrections to the robot's behavior. However, many tasks do not require input from the operator during the entire execution but only during *regions of task variability*. For example, during a fastener insertion task, the robot may only need help from the human to fix alignment error when inserting fasteners but not while fetching or prepping the fasteners. When the regions of variability make up a small part of the overall task the worker is poorly utilized. While the worker may perform secondary tasks during idle time, there are advantages to the operator working with multiple robots engaging in the same task, such as reducing context switching. In this work, we propose a method for multi-robot shared

[1]Michael Hagenow and Michael Zinn are with the Department of Mechanical Engineering, University of Wisconsin–Madison, Madison 53706, USA [mhagenow|mzinn]@wisc.edu
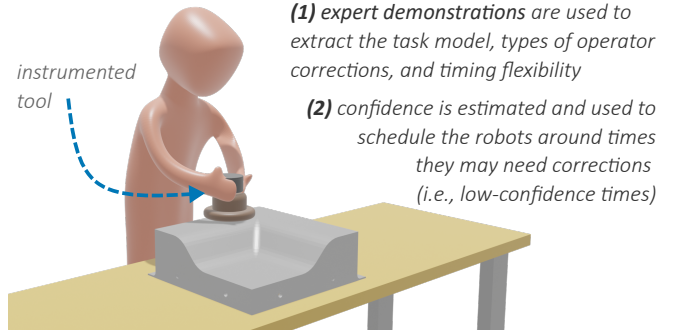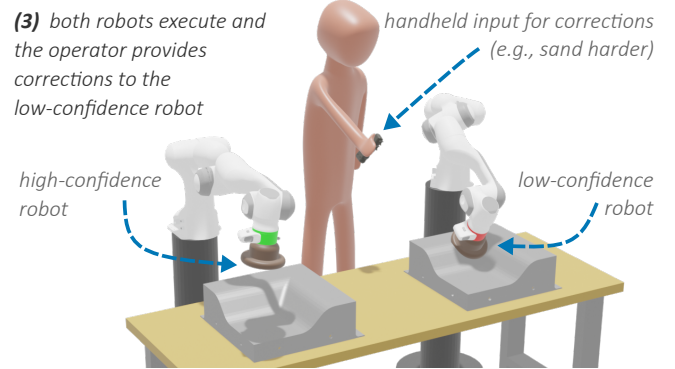
[2]Emmanuel Senft is a Research Scientist at the Idiap Research Institute, Martigny, Switzerland esenft@idiap.ch

[3]Nitzan Orr, Michael Gleicher, and Bilge Mutlu are with the Department of Computer Sciences, University of Wisconsin–Madison, Madison 53706, USA [nitzan|gleicher|bilge]@cs.wisc.edu

[4]Robert Radwin is with the Department of Industrial and Systems Engineering, University of Wisconsin–Madison, Madison 53706, USA rradwin@wisc.edu

[5]Dylan P. Losey is with the Department of Mechanical Engineering, Virginia Tech, Blacksburg 24061, USA losey@vt.edu

autonomy that sequences the execution of two robots based on regions where they may require assistance.

Previous work includes investigations of elements of multi-robot shared autonomy, such as interfaces for operator attention management [6], supervisor allocation across a fleet of agents (e.g., mobile robots) [3, 15, 4, 17, 11, 12], and scheduling of agent subtasks and supervision [19, 7, 1, 20]. However, the allocation methods focus on enabling an operator to temporarily teleoperate an agent needing assistance and in scheduling, little to no work focuses on coordinating agents around operator intervention. As illustrated in Figure 1, we are interested in *coordinated shared autonomy* where the robots operate at a high level of autonomy. In this paradigm, the robots sequence their behaviors such that only one robot could require assistance at any point, and the operator provides targeted corrections to the low-confidence robot if its action is incorrect. The major challenges in coordinated shared autonomy are determining *what* interventions the operator may want to provide and *when* they may occur during the task. Addressing these challenges requires models of the task, the types of corrections the operator may desire, and when corrections might be needed. Scheduling coordinated shared autonomy also introduces new challenges addressed in this work, such as how to compensate for changes to the timing of the robot's execution that result from operator corrections.

In this paper, we propose a method for multi-robot shared autonomy based on operator corrections [9], multi-agent scheduling, and Learning from Demonstration (LfD) [14]. Our method schedules two robots such that one operator can provide real-time corrections at times when they are needed by both robots. The task model, corrections an operator can make, and sequencing of agents are inferred from expert demonstrations of the task and whether an operator provided corrections in past shared autonomy executions. The main contributions of this paper include (1) an optimization-based method to schedule corrections-based shared autonomy of multiple agents by leveraging variability in demonstrations; (2) a technique for inferring when corrections are needed based on task variability and Bayesian inference; and (3) a real-time adaptation strategy to update the timing of robots when operator corrections cause deviations from the schedule.

II. OPTIMIZING FOR MULTI-ROBOT CORRECTIONS

Our method is formulated as an optimization problem and enables one human operator to provide real-time corrections to two robots completing different instances of the same task.

A. Problem Setting and Assumptions

We explore settings where there are variable aspects of the task that the robots *cannot learn* due to sensor limitations. For example, an expert sander may at times use subtle cues (e.g., watching material build up on the side of the tool, changing views of the surface) to choose an appropriate action. We consider settings with two robots and one skilled operator completing a high-volume (i.e., repetitive) task where the types of variability encountered do not change over time. There are

many industrial tasks that meet this requirement. For example, in many sanding applications, there are small regions of the workpiece that will have defects from upstream manufacturing processes which will cause each piece to have different sanding needs. To encode the robot behaviors, we assume access to a set, $\mathcal{D}$, of n_d varying-length expert demonstrations.

$$\mathcal{D} = \{\mathbf{X}_1, ... \mathbf{X}_{n_d}\}$$
$$\forall \mathbf{X}_i \in \mathcal{D}, \mathbf{X}_i \in \mathbb{R}^{m \times T_i} \tag{1}$$

where $\mathbf{X}_i$ is the demonstration state data (e.g., motions, forces, tool state) with m state variables and T_i is the length of the demonstration. For brevity, we will use subscripts on the demonstration set to refer to the data of an individual demonstration (i.e., $\mathcal{D}_i \equiv \mathbf{X}_i$) and refer to interpolated values of the demonstration data using function notation (i.e., $\mathbf{x}_i(t)$). To enable time warping and regression across multiple demonstrations, we assume the demonstrations are performed using similar trajectories (e.g., always going left to right when sanding). We also require that the robots must execute the task as shown in the demonstrations to satisfy any sequential constraints of the task. In other words, the execution cannot be broken up or reordered. Finally, we assume the robots must operate within the demonstration execution rates (i.e., velocities). While in some tasks it may be possible for the robots to stop at times when they need help, there are many examples where this is not possible. For example, a robot that needs help with part of a sanding pass cannot stop mid-pass without impacting the continuity of the sanding.

B. Proposed Approach

Our approach centralizes around scheduling the two robot executions. Our high-level method is illustrated in Figure 2. We start with a set of expert demonstrations to expose the task variability, including differences in how quickly the task was executed (i.e., the execution rate), and encode the robot behavior. To schedule the robots around operator corrections, we need to know what types of corrections the operator wishes to provide and when they may be needed. Following previous work [10], we assume that the demonstration variance indicates the corrections that may be necessary (e.g., variance in force indicates potential force corrections) and allow operators to provide these differential corrections by mapping the variance to a low degree-of-freedom input (e.g., joystick).

We determine when the corrections may be needed by estimating the robot confidence throughout the task. Specifically, we identify times in the robot behavior that might require corrections (i.e., low-confidence times) and times where corrections are unlikely (i.e., high-confidence times). The two robot behaviors are scheduled such that the low-confidence actions do not occur at the same time. The robot confidence is estimated iteratively. As the operator uses the system, it collects empirical data of when the operator provides corrections and refines its confidence estimates. Each time the system is used, we update the confidence and re-optimize the schedule based on the new confidence estimates. As the amount of high-confidence times increases, the scheduling method finds more
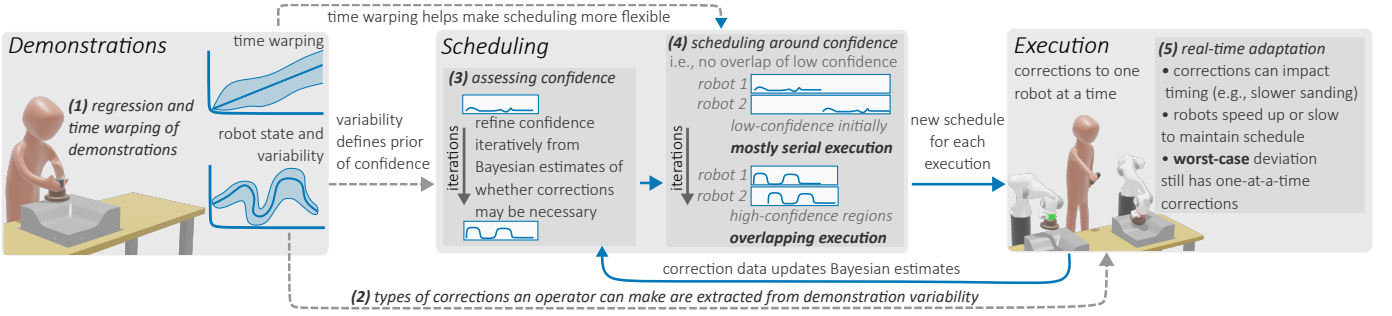
Fig. 2. High-level overview of the proposed method. The numbers correspond to the subsection of §II-B where the technical details are described. Blue arrows indicate the flow of information at every iteration. Gray arrows indicate information that is only processed once.

efficient schedules by shortening and overlapping the robot executions. Our scheduling method leverages flexibility in the execution rate during high-confidence times to aid in finding an optimized schedule. As operator corrections can impact the timing of the robots, we also ensure the scheduling solution is robust to timing deviations from the corrections. The following subsections describe our method's technical details.

1) Regression and Time Warping of Demonstrations: To encode the desired robot skill and to uncover time flexibility in the demonstrations, we take advantage of *time warping* [21]. A time warp, ψ, is a nonlinear and monotonically increasing function that maps a time between two sets of data:

$$\psi_{\mathcal{D}_1 \to \mathcal{D}_2} : [0, T_1] \to [0, T_2] \qquad (2)$$

This is accomplished by minimizing a loss between the data of the two time series. We consider a weighted metric that accounts for state variables with varying units. The time warping optimization can be summarized as:

$$\min_{\psi_{\mathcal{D}_1 \to \mathcal{D}_2}} \int_0^{T_1} \mathcal{L}(\mathbf{x}_2(\psi_{\mathcal{D}_1 \to \mathcal{D}_2}(t)) - \mathbf{x}_1(t))dt \qquad (3)$$
$$\mathcal{L}(\mathbf{e}_\psi) = \mathbf{e}_\psi^\mathsf{T} \mathbf{W} \mathbf{e}_\psi$$

where $\mathbf{e}_\psi$ is the error between the vectors of the two states and $\mathbf{W} \in \mathbb{R}^{m \times m}$ is a diagonal weighting matrix with nonnegative weights that is set based on variance or domain knowledge. We solve Equation 3 using a parametric optimizer [5]. We align all demonstrations to the first demonstration as a common reference and thus truncate the warp notation going forward by dropping the reference demonstration (e.g., $\psi_{\mathcal{D}_2} \equiv \psi_{\mathcal{D}_1 \to \mathcal{D}_2}$). Given that we will optimize new warps for each robot execution, the choice of reference is mostly arbitrary. Going forward, we mainly focus on the gradients of the warps. Working with the gradient allows us to assess the rate at which the robot can execute the task at any given time and allows us to easily enforce that warps are monotonically increasing.

Using the time warps, we align the demonstration data to determine the robot behavior. There are many possible methods for cloning the demonstration data. In this work, we use a simple regression that estimates of the mean behavior of the aligned demonstrations, which we denote as $\hat{\mu}(t)$.

2) Process Variability and Operator Corrections: During a robot's execution, the operator can provide real-time corrections to a robot's behavior. We provide a simplified control

to the operator by extracting the likely types of corrections from the variability in the expert demonstrations. We leverage our previous method [10] to determine admissible corrections, including coordinations of robot state variables (e.g., pitch, force, and speed to modulate abrasiveness). The remainder of this subsection provides a brief review of the previous method for completeness.

The differential corrections are provided through an input device that functions like a joystick. Given that the corrections can affect the execution rate, we consider an augmented robot state that includes the warp gradient.

$$\mathbf{x}_i^+(t) = [\mathbf{x}_i^\mathsf{T}(t), \dot{\psi}_{\mathcal{D}_i}(t)]^\mathsf{T} \qquad (4)$$

where the superscript + refers to the augmented state. The augmented mean behavior is also defined by concatenating the mean warp gradient of the demonstrations.

$$\dot{\bar{\psi}}_{\mathcal{D}}(t) = \sum_i^{n_d} \dot{\psi}_{\mathcal{D}_i}(t)/n_d$$
$$\hat{\mu}^+(t) = [\hat{\mu}^\mathsf{T}(t), \dot{\bar{\psi}}_{\mathcal{D}}(t)]^\mathsf{T} \qquad (5)$$

where $\dot{\bar{\psi}}_{\mathcal{D}}(t)$ is the mean warp gradient and $\hat{\mu}^+(t)$ is the augmented mean behavior. The mean-removed data and demonstration variability can subsequently be calculated:

$$\mathbf{e}_i^+(t) = \mathbf{x}_i^+ \left(\psi_{\mathcal{D}_i}(t) \right) - \hat{\mu}^+(t)$$
$$\sigma_{\mathcal{D}}^2(t) = \frac{1}{n_d - 1} \sum_i \left(\mathbf{e}_i^+(t) \right)^\mathsf{T} \mathbf{W}^+ \left(\mathbf{e}_i^+(t) \right) \qquad (6)$$

where $\mathbf{e}_i^+(t)$ is the mean-removed data, $\sigma_{\mathcal{D}}^2(t)$ is the magnitude of the demonstration variance, and $\mathbf{W}^+ \in \mathbb{R}^{(m+1) \times (m+1)}$ is the expanded diagonal weighting matrix that adds a weight for the warp gradient variable. The final operator correction, $\delta \mathbf{x}^+(u, t)$, is based on the principal components at a given time (using principal component analysis) and the user input, u. The maximum magnitude of the correction is scaled such that the corrections do not exceed the permissible warp gradients from the demonstrations. The final robot command is arbitrated as the sum of the mean state and differential operator correction.

$$\mathbf{x}_f^+(u, t) = \hat{\mu}^+(t) + \delta \mathbf{x}^+(u, t) \qquad (7)$$

where $\mathbf{x}_f^+$ is the final robot state. The correction also determines the next robot time based on the warp gradient. The timing of the robots is discussed in detail in a later section.

3) Assessing Robot Confidence: Intuitively, we want the robots to overlap at times when at least one robot is confident and *not* to overlap when they are both unsure. This scheduling requires determining the times of the execution where the robot may require corrections (i.e., the robot has low confidence in its action). Let $p_{c|t}$ be the probability of the expert giving a correction to a robot at a given time. We desire to estimate $p_{c|t}$ and then threshold the value to determine whether a robot is low or high confidence. While variability in the demonstrations serves as an indicator of what corrections an operator may provide, this estimate is too conservative for estimating $p_{c|t}$ given the multiple sources of demonstration variability [18]. In some cases, the variability is from task uncertainty (e.g., different forces depending on paint when sanding). However, there are also cases when demonstration variability is non-critical. For example, an operator's tool path when moving between sanding passes is non-critical and underconstrained, and thus may be variable.

Our solution leverages the demonstration variability and empirical correction data from past executions of the system to iteratively estimate the probability of corrections at every time step. Initially, the system will only be high-confidence for times where there was negligible variability in the human demonstrations. However, as the system collects empirical data from the n_e previous executions of whether corrections are provided, the system will infer additional high-confidence times and can schedule shorter, more parallel solutions to reduce the overall time on task. The empirical data, $\mathbf{z}_t \in \mathbb{B}^{n_e}$, is a set for each time step (defined in the reference demonstration) of whether corrections were given during each robot execution. With this empirical data, we use Bayesian inference to estimate the probability distribution that a correction is needed.

$$P(p_{c|t} \mid \mathbf{z}_t) = \frac{P(\mathbf{z}_t \mid p_{c|t})P(p_{c|t})}{P(\mathbf{z}_t)} \qquad (8)$$

Given the empirical data is Boolean and assuming the events (i.e., executions) are independent, this inference problem can be modeled using a Beta distribution [8]. The Bayesian posterior can be inferred based on a prior and the empirical data:

$$P(p_{c|t} \mid \mathbf{z}_t) = beta(\alpha_{0,t} + \sum \mathbf{z}_t, \beta_{0,t} + n_e - \sum \mathbf{z}_t) \qquad (9)$$

where *beta* is the Beta distribution and $\alpha_{0,t}$ and $\beta_{0,t}$ define the Beta distribution prior. We here model the prior in terms of the mean (μ_t) and variance (σ_t^2) of the distribution:

$$\alpha_{0,t} = \mu_t \left(\frac{\mu_t(1 - \mu_t)}{\sigma_t^2} - 1 \right)$$
$$\beta_{0,t} = (1 - \mu_t) \left(\frac{\mu_t(1 - \mu_t)}{\sigma_t^2} - 1 \right) \qquad (10)$$

Our Beta prior is a heuristic that reflects the intuition that the confidence should be high for times with near-zero variability (i.e., no corrections) and uncertain otherwise (i.e., $p_{c|t} \approx \frac{1}{2}$):

$$\mu_t = \frac{1}{2} \left(1 - e^{-\gamma_p(\sigma_{\mathcal{D}}^2(t)+\epsilon)} \right)$$
$$\sigma_t^2 = \sigma_{\text{MAX}}^2 \left(1 - e^{-\gamma_p(\sigma_{\mathcal{D}}^2(t)+\epsilon)} \right) \qquad (11)$$

where γ_p and σ_{MAX}^2 are scaling parameters for the prior and ϵ is a small constant to make the prior well posed. For scheduling, we determine whether the robot is confident, $\text{conf}(t) \in \mathbb{B}$ (i.e., the robot is low or high confidence), from the Beta cumulative distribution function (CDF):

$$\text{conf}(t) = p(p_{c|t} < \mu_c \mid \mathbf{z}_t, t) > \gamma_c \qquad (12)$$

where μ_c is the probability of corrections where it is acceptable for the robot to execute without supervision and γ_c is the corresponding confidence interval of the distribution.

4) Scheduling Around Confidence: The full scheduling problem consists of determining a task execution (i.e., time warp), $\psi_{\mathcal{E}_i}$, for each robot and a start offset for the second robot, τ. Each execution warp, $\psi_{\mathcal{E}_i}$, is defined relative to the reference demonstration (similar to the demonstrations) and scheduled by choosing permissible values of the warp gradient. We desire to schedule the execution warps and offset to avoid overlapping low-confidence actions and to minimize the expected total time, T_{total}, for both robots to finish the task.

$$T_{\text{total}} = \max\{|\psi_{\mathcal{E}_1}|, \tau + |\psi_{\mathcal{E}_2}|\} \qquad (13)$$

where $|\psi_{\mathcal{E}_i}|$ denotes the length of the execution warp (i.e., the duration of the behavior). For each robot, we determine a warped execution from the mean warp gradient, $\dot{\psi}(t)$, and the minimum and maximum demonstration warp gradients.

$$\dot{\psi}^{\text{min}}(t) = \min\{\dot{\psi}_{\mathcal{D}_1}(t), \ldots, \dot{\psi}_{\mathcal{D}_{n_d}}(t)\}$$
$$\dot{\psi}^{\text{max}}(t) = \max\{\dot{\psi}_{\mathcal{D}_1}(t), \ldots, \dot{\psi}_{\mathcal{D}_{n_d}}(t)\} \qquad (14)$$

where $\dot{\psi}^{\text{min}}(t)$ and $\dot{\psi}^{\text{max}}(t)$ are the minimum and maximum warp gradients respectively.

When the robot is executing a low-confidence action with operator corrections, the warps should follow the mean warp gradient and mean behavior. When the robot is executing a high-confidence action, we have the flexibility to warp the timing within the demonstration bounds to increase the likelihood of finding a non-overlapping scheduling solution. While it may be possible in some tasks to warp the executions beyond what was seen in the demonstrations (e.g., stop certain robots, replan between critical regions), we choose to respect the demonstration bounds to limit violating any latent task constraints (e.g., time required to start a tool before beginning a critical region, maximum time before a sealant dries).

To this point, we have discussed timing in the context of the reference demonstration. Our formulation ultimately creates new behavior warps and schedules them in a coordinated global fashion. To simplify the notation in our formulation, we introduce a global scheduling notation to convert between times in the overall execution (i.e., both robots) and the corresponding time in a robot's reference behavior.

$$t_{\mathcal{E}_i}(t) = \begin{cases} 0 & t < \tau_i \\ \psi_{\mathcal{E}_i}^{-1}(t - \tau_i) & \tau_i \leq t \leq \tau_i + |\psi_{\mathcal{E}_i}| \\ \psi_{\mathcal{E}_i}^{-1}(|\psi_{\mathcal{E}_i}|) & else \end{cases} \qquad (15)$$

where $\psi_{\mathcal{E}_i}^{-1}$ is the inverse warp that goes from the execution to the reference demonstration and τ is a vector of offsets with

the first offset set to zero (i.e., $\boldsymbol{\tau} = [0, \tau]$). With this global reference time, it is possible to assess the confidence of each robot at any time (set to one when not running) or both robots.

$$\text{conf}_i(t) = \begin{cases} \text{conf}(t_{\mathcal{E}_i}(t)) & \tau_i \leq t \leq \tau_i + |\psi_{\mathcal{E}_i}| \\ 1 & else \end{cases} \tag{16}$$

$$\mathbf{conf}(t) = [\text{conf}_1(t), \text{conf}_2(t)]$$

5) Real-time Adaptation to Operator Corrections: One of the major challenges with scheduling in the shared autonomy setting is that operator corrections can affect the timing of the robots. Given that we know the bounds of operator corrections and the corresponding impacts on timing, we desire to find a scheduling solution that avoids overlapping low-confidence regions under any permissible operator correction.

The simplest way to ensure robustness would be to show that both robots can instantaneously accommodate any timing changes from the corrections (i.e., if one robot slows down or speeds up, the other robot can also slow down or speed up). However, one robot may be completing an action that cannot accommodate the change in real time (e.g., a low-variability sanding pass where there is no flexibility in the speed). In this case, we must show the robots can *make up the timing change* before the next low-confidence region.

Our strategy uses the execution gradient to determine how much faster or slower the robot can execute the task to make up timing changes. One important point is that the minimum gradient refers to speeding up a behavior and the maximum gradient refers to slowing down the behavior. For example, if $\dot{\psi}_{\mathcal{E}_i}^{\min} = 0.5$, the behavior could run in half as many samples or equivalently could run twice as fast. Thus, to calculate how much faster or slower a robot can go (F and S respectively), we divide the scheduled gradient by the demonstration limits.

$$\dot{\psi}_{\mathcal{E}_i}^{F}(t) = \dot{\psi}_{\mathcal{E}_i}(t_{\mathcal{E}_i}(t)) \,/\, \dot{\psi}^{\min}(t_{\mathcal{E}_i}(t))$$
$$\dot{\psi}_{\mathcal{E}_i}^{S}(t) = \dot{\psi}_{\mathcal{E}_i}(t_{\mathcal{E}_i}(t)) \,/\, \dot{\psi}^{\max}(t_{\mathcal{E}_i}(t)) \tag{17}$$

For example, if $\dot{\psi}_{\mathcal{E}_i} = 3$ and $\dot{\psi}^{\min} = 0.5$ at a given time, the robot was scheduled at one third of the reference demonstration speed and could go six times as fast (i.e., $\dot{\psi}_{\mathcal{E}_i}^{F} = 3 \,/\, 0.5 = 6$).

Algorithm 1 Strategy For Two High-Confidence Robots

1: $a \leftarrow$ ahead robot (or either if no deviation)
2: $b \leftarrow$ behind robot
3: **function** HIGH-CONF($t_a, t_b, i_a, i_b, \cdot$)
4:     $\Delta T_{ab} \leftarrow t_a - t_b$
5:     **if** $\dot{\psi}_{i_b}^{F}(t_b)\Delta t_s \geq \dot{\psi}_{i_a}^{F}(t_a)\Delta t_s + \Delta T_{ab}$ **then**
           ▷ *behind robot can overtake the ahead robot*
6:         $\delta t_b = \dot{\psi}_{i_a}^{F}(t_a)\Delta t_s + \Delta T_{ab}$
7:         $\delta t_a = \dot{\psi}_{i_a}^{F}(t_a)\Delta t_s$
8:     **else**   ▷ *behind robot cannot overtake ahead robot*
9:         $\delta t_b = \dot{\psi}_{i_b}^{F}(t_b)\Delta t_s$
10:        $\delta t_a = \delta t_{i_a}^{\text{resp}}(\Delta T_{ab} - \delta t_b, t_a)$
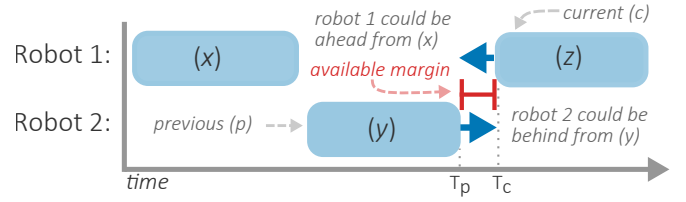11:   **return** $\delta t_a, \delta t_b$

Fig. 3. Illustrating the required margin between a previous low-confidence region (y) and current low-confidence region (z) for a scheduling solution. The worst-case scenario from operator corrections occurs when Robot 1 takes shorter than scheduled on x and Robot 2 takes longer than scheduled on y.

For the real-time strategy, we consider each robot's strategy at discrete time steps, t_n. We assume that the robots are commanded at a fixed sample rate, Δt_s. During execution, corrections will cause the robots to deviate from the original schedule. Thus, we separately track and increment the time of each robot by δt_i based on the execution rate selected by the robot strategy (i.e., $t_i \leftarrow t_i + \delta t_i$). The goal of the two robots is to minimize the relative change in timing between the two executions (e.g., $\Delta T_1 = t_1 - t_2$). For brevity, we use a dot within functions (i.e., $f(\cdot)$) to indicate that there are additional previously defined quantities used within the equation. The real-time strategy of the robots can be summarized as follows:

- If a robot is executing a low-confidence action, its time increment is determined by the operator's correction, δt_i^u. The other robot chooses its response, δt_i^{resp}, based on the current deviation from the other robot and the bounds of how much it can slow down or speed up. If the responding robot is ahead, it simply goes as slow as possible. If the responding robot is behind, it chooses the move within its limits that minimizes the deviation between the two robots. The full response strategy can be summarized as:

$$\delta t_i^{\text{resp}}(\Delta T_i, t_i) = \begin{cases} \dot{\psi}_{\mathcal{E}_i}^{S}(t_i)\Delta t_s & -\Delta T_i < \dot{\psi}_{\mathcal{E}_i}^{S}(t_i)\Delta t_s \\ -\Delta T_i & -\Delta T_i \geq \dot{\psi}_{\mathcal{E}_i}^{S}(t_i)\Delta t_s \text{ and} \\ & -\Delta T_i < \dot{\psi}_{\mathcal{E}_i}^{F}(t_i)\Delta t_s \\ \dot{\psi}_{\mathcal{E}_i}^{F}(t_i)\Delta t_s & else \end{cases} \tag{18}$$

- When both robots are high-confidence, the robots can plan the fastest move possible while reducing any deviation between the two executions, as described in Algorithm 1. The goal of the robot that is running behind is to go as fast as possible without surpassing the robot that is ahead.
- If the first robot induces a change to timing while the second robot hasn't yet started, the second robot can simply shift its start time (i.e., increment by the same step). Similarly, if one robot induces a change when the other robot has finished its behavior, there is no need for adaptation.

With this real-time adaptation strategy, we propose to show that even in the worst case deviation between the two robots, there will still be *no* overlapping low-confidence regions. Consider the example in Figure 3. The worst case scenario occurs when the second robot is running behind and the first robot is running ahead (i.e., both behaviors trend toward causing overlap). For each set of adjacent low-confidence regions, we can check if the worst-case behavior leads to the current low-

confidence region starting before the previous one ends. The full margin-checking procedure, SUFFICIENT_MARGINS(·), is described in Algorithm 2. It is important to note that the implementation details, such as a small sample rate in the discretization, impact the validity of the margin calculation.

6) Optimization: Given the required solution properties, we can summarize the multi-robot shared autonomy optimization and solve for the robot execution warps, $\psi_{\mathcal{E}} = \{\psi_{\mathcal{E}_1}, \psi_{\mathcal{E}_2}\}$, and offset, τ. The execution warps are chosen based on the permissible warp gradients and defined as an array with the same length as the reference demonstration. To simplify the optimization notation, we define the set of time steps from the reference demonstration and the times in the global notation where a robot is executing a low-confidence action.

$$\mathcal{T}_{ref} = \{t_n \in [0, |\psi_{D_1}|]\}$$
$$\mathcal{T}_{lc} = \{t_n \in [0, T_{\text{total}}] \mid \min(\mathbf{conf}(t_n)) = 0\} \quad (19)$$

In our scheduling optimization, the constraints represent strict requirements of the scheduling solution and the minimization quantity expresses additional desiderata for the scheduling solution. The optimization is summarized as:

$$\underset{\psi_{\mathcal{E}}, \tau}{\arg\min} \quad \mathbf{w}^\top \phi(\cdot)$$

$$\begin{aligned}
s.t. \quad & \forall (t_n, \psi) \in \mathcal{T}_{ref} \times \psi_{\mathcal{E}} : \dot{\psi}^{\min}(t_n) \le \dot{\psi}(t_n) \le \dot{\psi}^{\max}(t_n) \\
& 0 \le \tau \le |\psi_{\mathcal{E}_1}| \\
& \forall t_n \in [0, T_{\text{total}}] : \max(\mathbf{conf}(t_n)) = 1 \qquad (20) \\
& \forall (t_n, \psi) \in \mathcal{T}_{lc} \times \psi_{\mathcal{E}} : \dot{\psi}(t_n) = \bar{\dot{\psi}}_{\mathcal{D}}(t_n) \\
& \text{SUFFICIENT\_MARGINS}(\cdot) = True
\end{aligned}$$

where ϕ are the costs to minimize and $\mathbf{w}$ are the corresponding weights. The constraints enforce that (1) all warps are within the demonstration limits; (2) that the second execution is offset appropriately relative to the first execution; (3) that no unsupervised robot is performing a low-confidence action (i.e., that one of the robots is high-confidence); (4) that the low-confidence robot follows the mean time warping; and (5) that the scheduling solution accommodates variations in timing from corrections. For solutions satisfying the constraints, we can also specify features that differentiate the quality of solutions. In this work, we start with one feature minimizing the total time on task (e.g., $\phi(\cdot) = [T_{\text{total}}]$). In the future, we will expand the minimization quantity to focus on aspects that improve human factors [2]. For example, it may be desirable to minimize switches or to favor solutions with time for the operator to gain situational awareness after switches.

C. Implementation of Scheduling

We solve the scheduling optimization offline prior to each use of the system based on the current confidence estimates of the robot behavior. Given the non-convexity of our scheduling problem and dependencies between our optimization variables, we use the anytime sampling approach described below. During each iteration, we sample the offset based on the length of

---

Algorithm 2 Margin Between Low Confidence (LC) Regions

1: **function** SUFFICIENT_MARGINS(·)
2:     $i_p, T_p, t_{n-1} \leftarrow \varnothing, \varnothing, \varnothing$ ▷ *no previous yet*
3:     **for** $t_n \in [0, T_{\text{total}}]$ **do**
4:       start_LC $\leftarrow\,!\min(\mathbf{conf}(t_n))$ & $\min(\mathbf{conf}(t_{n-1}))$
5:       end_LC $\leftarrow \min(\mathbf{conf}(t_n))$ & $!\min(\mathbf{conf}(t_{n-1}))$
6:       **if** start_LC **then**
7:         $i_c \leftarrow \arg\min(\mathbf{conf}(t_n))$
8:         $T_c \leftarrow t_n$
9:         **if** i_p **then**
10:           $ok \leftarrow$ CHECK_MARGIN$(i_p, T_p, i_c, T_c, \cdot)$
11:           **if** $!ok$ **then**
12:             **return** False
13:       **if** end_LC **then**
14:         $i_p \leftarrow \arg\min(\mathbf{conf}(t_{n-1}))$
15:         $T_p \leftarrow t_{n-1}$
16:       $t_{n-1} \leftarrow t_n$
17:     **return** True
18: **function** CHECK_MARGIN$(i_p, T_p, i_c, T_c, \cdot)$
19:     **if** $i_p = i_c$ **then**   ▷ *same robot, no issue*
20:       **return** True
21:     $done \leftarrow$ False
22:     $t_p, t_c \leftarrow \tau$ ▷ *only times when both robots are running*
23:     **while** $!done$ **do**
24:       **if** $\mathbf{conf}_{i_c}(t_c) = 0$ **or** $\mathbf{conf}_{i_p}(t_p) = 0$ **then**
25:         **if** $!\,\mathbf{conf}_{i_c}(t_c)$ **then**   ▷ *worst case is faster*
26:           $t_c \leftarrow t_c + \dot{\psi}_{i_c}^F(t_c)\Delta t_s$
27:           $t_p \leftarrow t_p + \delta t_{i_p}^{\text{resp}}(t_p - t_c, t_p)$
28:         **else**   ▷ *worst case is slower*
29:           $t_p \leftarrow t_p + \dot{\psi}_{i_p}^S(t_p)\Delta t_s$
30:           $t_c \leftarrow t_c + \delta t_{i_c}^{\text{resp}}(t_c - t_p, t_c)$
31:       **else**
32:         **if** $(t_c - t_p) > 0$ **then**   ▷ *previous is behind*
33:           $\delta t_c, \delta t_p \leftarrow$ HIGH-CONF$(t_c, t_p, i_c, i_p, \cdot)$
34:         **else** ▷ *current is behind or the two are equal*
35:           $\delta t_p, \delta t_c \leftarrow$ HIGH-CONF$(t_p, t_c, i_p, i_c, \cdot)$
36:         $t_p \leftarrow t_p + \delta t_p$
37:         $t_c \leftarrow t_c + \delta t_c$
38:       **if** $t_c \ge T_c$ **then**
39:         **return** False   ▷ *possible overlap*
40:       **if** $t_p \ge T_p$ **then**
41:         $done \leftarrow$ True   ▷ *no overlap*
42:     **return** True

---

the first execution and sample the warps from the permissible bounds during high-confidence actions. The warps follow the mean demonstration warp during low-confidence times. The solution that minimizes the total time while meeting the constraints is kept. If no satisfying solution is found, we can fall back the fastest serial execution or the previous scheduling solution. As high-confidence time steps cannot transition back to low-confidence time steps (i.e., cannot be given corrections), the previous scheduling solution will always remain valid.

After each iteration of running the two-robot system, we acquire two new demonstrations of the task and two new
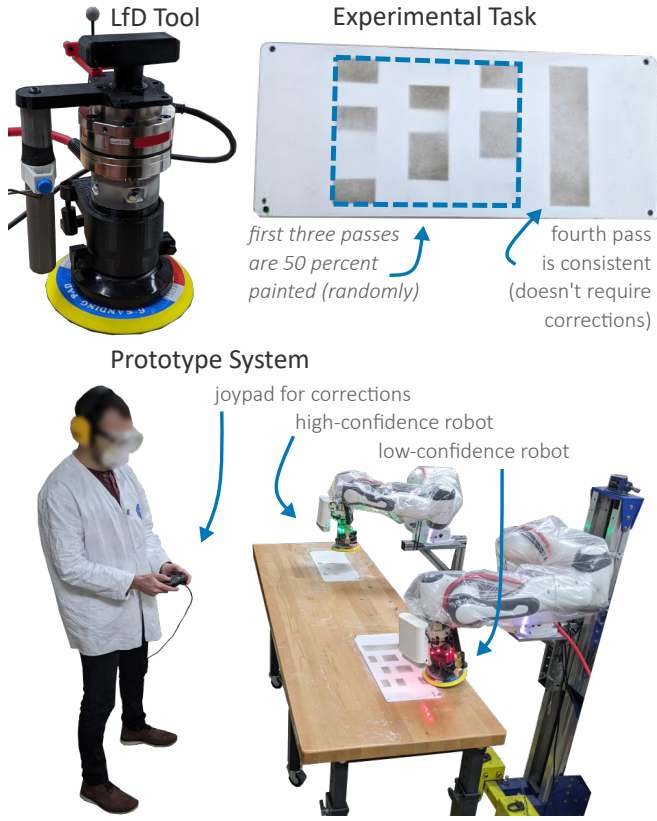
Fig. 4. Instrumented LfD tool, experiment task, and prototype system.

Boolean arrays of whether corrections were given during low-confidence samples. With these new data, we can shift the mean behavior and append our correction observations.

$$\mathcal{D}^{iter+1} \leftarrow \mathcal{D}^{iter} + \{\mathbf{X}_1^{iter+1}, \mathbf{X}_2^{iter+1}\}$$
$$\hat{\boldsymbol{\mu}}^{iter+1}(t) \leftarrow \mathbf{f}(\mathcal{D}^{iter+1}) \qquad (21)$$
$$\forall t : \mathbf{z}_t^{iter+1} \leftarrow \mathbf{z}_t^{iter} + \{z_{t,1}^{iter+1}, z_{t,2}^{iter+1}\}$$

The updated correction observations can be used to update the confidence. In practice, the corrections to each robot occur to fractional samples of the reference robot behavior, thus we interpolate when determining the empirical corrections.

III. PRELIMINARY EVALUATION

As a preliminary assessment of our method, we designed an experimental task, collected demonstrations, and ran a simulated experiment to assess how our scheduling solution adapts with use of the shared autonomy system. We also developed a prototype system, shown in Figure 4 (bottom), that is demonstrated in the supplemental video. The experimental task was inspired by industrial sanding applications (e.g., aircraft and automotive manufacturing) where portions of the task have variable sanding needs. The task consisted of four sanding passes of gray spray paint on a piece of white acrylic, as shown in Figure 4 (top-right). The first three passes were variable, where 50% of the piece was painted in randomly chosen sections. The fourth pass was consistently painted (i.e., did not require corrections). During demonstrations, the operator applied more force and moved
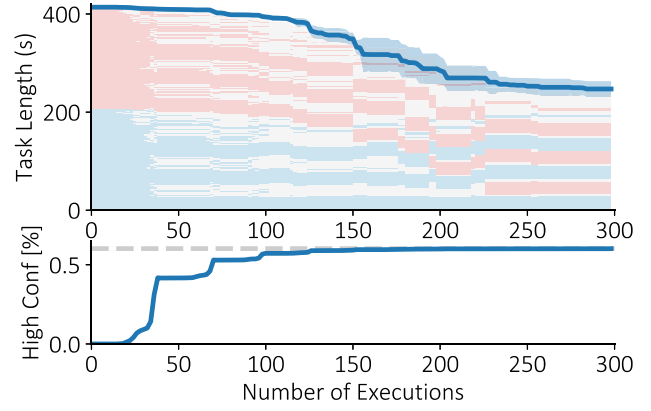

Fig. 5. Evolution of scheduling with simulated use of the system. *Top:* The task length decreases over repeated interactions. The dark blue line shows the mean task length (with shading for the standard deviation). For each iteration, we show an example of the proposed schedule where blue means the first robot is low confidence, red means the second robot is low confidence, and gray means neither robot is low confidence. Initially, the robots are mostly low confidence and execute the task one at a time. By the end, the robots alternate executing low-confidence actions. *Bottom:* Percent of high-confidence samples. The gray dotted line is the ground-truth percentage.

the tool more slowly over the painted sections and applied very little force and moved quickly when there was no paint. As a result, the learned nominal program *lightly* sanded the first three passes and required corrections by the operator to adjust the abrasiveness. During each demonstration, the tool was moved slowly between passes as a simple way to ensure the task was amenable to a highly-parallel execution (i.e., one robot sanding while the other moved between passes). In practice, other high-confidence actions between passes could enable parallel scheduling, such as changing sanding discs. The authors collected a total of six sanding demonstrations using the instrumented sanding tool shown in Figure 4 (top-left). We collected the pose of the tool, the applied force, and the state of the tool (i.e., on/off) during each demonstration.

The simulated experiment consisted of five trials with 300 simulated executions (in pairs of two) of the system with operator corrections. Consistent with the task, corrections were given 50% of the time on samples during the first three passes. We chose to include a human error rate of one percent in our simulation (i.e., the human failed to provide needed corrections 1% of the time and provided corrections when unnecessary 1% percent of the time). After each pair of executions, the confidence was updated and the schedule was resampled with 5000 iterations. In our experiment (dependent on behavior length), the schedule sampling took 3 minutes (*i5-10400F*, 10 threads). We recorded both the percentage of high-confidence samples after each set of executions and the task length. The results, shown in Figure 5, demonstrate that as the robots receive corrections, the system can refine and leverage its confidence to optimize the task scheduling.

IV. DISCUSSION

Our preliminary findings suggest that our proposed method enables users to efficiently share autonomy with multiple robots. We demonstrated how the scheduling algorithm re-

duces the time to complete a sanding task as new high-confidence times are identified. As the high-confidence samples approached the ground-truth percentage, our example found a schedule with alternating sanding passes. This observation highlights the value of our optimization-based scheduling. The system will always find solutions that respect the optimization constraints (e.g., no overlapping, low-confidence actions) and reduce the total time by either speeding up high-confidence actions or overlapping executions when possible.

A. Limitations & Future Work

Our method inherits limitations from LfD and time warping, which means that the demonstrations need to be performed in a similar way. To avoid these assumptions, we are interested in exploring a reward-based structure that can better generalize to heterogeneous demonstrations. We are also interested in scaling our technical methods from two robots to larger fleets and also exploring the practical considerations of controlling larger numbers of agents [2]. Finally, our preliminary evaluation presented only one example sanding task to highlight the benefits of the proposed method. In the future, we will conduct user studies and explore a range of tasks, including different tasks for each robot.

REFERENCES

[1] Yifan Cai, Abhinav Dahiya, Nils Wilde, and Stephen L Smith. Scheduling operator assistance for shared autonomy in multi-robot teams. *arXiv preprint arXiv:2209.03458*, 2022.

[2] Jessie YC Chen and Michael J Barnes. Human–agent teaming for multirobot control: A review of human factors issues. *IEEE Transactions on Human-Machine Systems*, 44(1):13–29, 2014.

[3] Jacob W Crandall, Mary L Cummings, Mauro Della Penna, and Paul MA De Jong. Computing the effects of operator attention allocation in human control of multiple robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):385–397, 2010.

[4] Abhinav Dahiya, Nima Akbarzadeh, Aditya Mahajan, and Stephen L Smith. Scalable operator allocation for multi-robot assistance: A restless bandit approach. *IEEE Transactions on Control of Network Systems*, 2022.

[5] Dave Deriso and Stephen Boyd. A general optimization framework for dynamic time warping. *Optimization and Engineering*, pages 1–22, 2022.

[6] Fei Gao, Missy L Cummings, and Luca F Bertuccelli. Teamwork in controlling multiple robots. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 81–88. IEEE, 2012.

[7] Matthew Gombolay, Ronald Wilcox, and Julie Shah. Fast scheduling of multi-robot teams with temporospatial constraints. 2013.

[8] Arjun K Gupta and Saralees Nadarajah. *Handbook of beta distribution and its applications*. CRC press, 2004.

[9] Michael Hagenow, Emmanuel Senft, Robert Radwin, Michael Gleicher, Bilge Mutlu, and Michael Zinn. Corrective shared autonomy for addressing task variability. *IEEE Robotics and Automation Letters*, 6(2):3720–3727, 2021.

[10] Michael Hagenow, Emmanuel Senft, Robert Radwin, Michael Gleicher, Bilge Mutlu, and Michael Zinn. Informing real-time corrections in corrective shared autonomy through expert demonstrations. *IEEE Robotics and Automation Letters*, 6(4):6442–6449, 2021.

[11] Ryan Hoque, Lawrence Yunliang Chen, Satvik Sharma, Karthik Dharmarajan, Brijen Thananjeyan, Pieter Abbeel, and Ken Goldberg. Fleet-dagger: Interactive robot fleet learning with scalable human supervision. *arXiv preprint arXiv:2206.14349*, 2022.

[12] Tianchen Ji, Roy Dong, and Katherine Driggs-Campbell. Traversing supervisor problem: An approximately optimal approach to multi-robot assistance. *arXiv preprint arXiv:2205.01768*, 2022.

[13] Dylan P Losey, Craig G McDonald, Edoardo Battaglia, and Marcia K O'Malley. A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction. *Applied Mechanics Reviews*, 70(1), 2018.

[14] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

[15] Ariel Rosenfeld, Noa Agmon, Oleg Maksimov, and Sarit Kraus. Intelligent agent supporting human–multi-robot team collaboration. *Artificial Intelligence*, 252:211–231, 2017.

[16] Mario Selvaggio, Marco Cognetti, Stefanos Nikolaidis, Serena Ivaldi, and Bruno Siciliano. Autonomy in physical human-robot interaction: A brief survey. *IEEE Robotics and Automation Letters*, 2021.

[17] Gokul Swamy, Siddharth Reddy, Sergey Levine, and Anca D Dragan. Scaled autonomy: Enabling human operators to control robot fleets. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5942–5948. IEEE, 2020.

[18] Emanuel Todorov and Michael Jordan. A minimal intervention principle for coordinated movement. *Advances in neural information processing systems*, 15, 2002.

[19] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12): 399, 2013.

[20] Sebastián A Zanlongo, Peter Dirksmeier, Philip Long, Taskin Padir, and Leonardo Bobadilla. Scheduling and path-planning for operator oversight of multiple robots. *Robotics*, 10(2):57, 2021.

[21] Feng Zhou and Fernando De la Torre. Generalized time warping for multi-modal alignment of human motion. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1282–1289. IEEE, 2012.