Human-in-the-Loop Mixed-Initiative Control under Temporal Tasks

Meng Guo, Sofie Andersson and Dimos V. Dimarogonas

Abstract— This paper considers the motion control and task planning problem of mobile robots under complex high-level tasks and human initiatives. The assigned task is specified as Linear Temporal Logic (LTL) formulas that consist of hard and soft constraints. The human initiative influences the robot autonomy in two explicit ways: with additive terms in the continuous controller and with contingent task assignments. We propose an online coordination scheme that encapsulates (i) a mixed-initiative continuous controller that ensures all-time safety despite of possible human errors, (ii) a plan adaptation scheme that accommodates new features discovered in the workspace and short-term tasks assigned by the operator during run time, and (iii) an iterative inverse reinforcement learning (IRL) algorithm that allows the robot to asymptotically learn the human preference on the parameters during the plan synthesis. The results are demonstrated by both realistic human-in-the-loop simulations and experiments.

I. INTRODUCTION

Autonomous systems are becoming increasingly prevalent in our daily life, with examples such as self-driving vehicles, package delivery drones and household service robots [1]. Nevertheless, these autonomous systems often perform the intended tasks under the supervision or collaboration with human operators [2]. On the high level, the human operator could assign tasks for the robot to execute or monitor the task execution progress during run time. On the low level, the operator could directly influence or even overtake the control commands of the robot from the on-board autonomous controller, which can be useful to guide the robot through difficult parts of the task [2]–[4]. On the other hand, the autonomous controller should take into account possibly erroneous inputs from the operator and ensure that safety constraints are never violated. Thus, addressing properly these online interactions between the autonomous system and the operator during the design process is essential for the safety and efficiency of the overall system.

In this work, we consider the interactions on both levels. Particularly, on the high level, the operator assigns (i) offline a local task as LTL formulas for hard and soft task constraints, and (ii) online temporary tasks with deadlines. On the low level, the operator's control inputs is fused directly with the autonomous controller via a mixed-initiative controller. The proposed motion and task planning scheme ensures that the hard task constraints regarding safety are obeyed at all time, while the soft constraints for performance

are improved gradually as the robot is guided to explore the workspace and more importantly, learn about the human preference over the synthesized plan.

We rely on LTL as the formal language [5] to describe complex high-level tasks beyond the classic point-to-point navigation. Many recent papers can be found that combine robot motion planning with model-checking-based task planning, e.g., a single robot under LTL motion tasks [6]–[9], a multi-robot system under a global task [10], or a multi-robot system under local independent [11] or dependent tasks [12], [13]. However, none of the above addresses directly the human initiative, neither in the continuous control nor in the discrete planning. On the other hand, human inputs are considered in [14] via GR(1) task formulas that require the robot to be reactive to simple sensory signals from the human. The high-level robot-human interaction is modeled as a two-player Markov Decision Process (MDP) game in [15], [16] where they take turns to influence the system evolution. The goal is to design a shared control policy to satisfy a LTL formula and minimize a cost function. Another recent work [17] addresses the control problem of MDP under LTL formulas where the autonomy strategy is blended into the human strategy in a minimal way that also ensures safety and performance. However, the direct interaction on the low level is not investigated in the aforementioned work. More importantly, an all-time involvement of the human is required for these frameworks, while we only assume human intervention whenever preferred by the operator itself.

Furthermore, the notion of mixed-initiative controller is firstly proposed in [4] that combines external human inputs with the traditional navigation controller [18], while ensuring safety of the overall system. The work in [19], [20] proposes a systematic way to compose multiple control initiatives using barrier functions. However, high-level complex temporal tasks are not considered in these work.

The main contribution of this work lies in a novel human-in-the-loop control framework that allows human interaction on both high level as complex tasks and low level as continuous inputs. We ensure all-time safety during the interaction and accommodation of short-term contingent tasks assigned during run time. Lastly, the proposed IRL algorithm enables the robot to asymptotically learn and adapt to the human operator's preference in the plan synthesis.

The rest of the paper is organized as follows: Section II introduces some preliminaries of LTL. Section III formulates the problem. Main algorithmic parts are presented in Section IV, which are integrated into the complete framework in Section V. Numerical and experiment studies are shown in Sections VI. We conclude in Section VII.

II. PRELIMINARIES

A. Linear Temporal Logic (LTL)

A LTL formula over a set of atomic propositions AP that can be evaluated as true or false is defined inductively according to the following syntax [5]: $\varphi ::= \top \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \cup \varphi_2$, where $\top \triangleq \texttt{True}$, $a \in AP$ and $\bigcirc$ (next), $\cup$ (until). There are other useful operators like $\square$ (always), $\diamond$ (eventually), $\Rightarrow$ (implication). The full semantics and syntax of LTL are omitted here due to limited space, see e.g., [5]. Syntactically co-safe LTL (sc-LTL) is a subclass of LTL that can be fulfilled by finite satisfying prefix [21].

B. Büchi Automaton

Given a LTL formula φ, there always exists a Nondeterministic Büchi Automaton (NBA) $\mathcal{A}_\varphi$ that accepts all the languages that satisfy φ [5]. It is defined as $\mathcal{A}_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$, where Q is a finite set of states; $Q_0 \subseteq Q$ is the set of initial states, 2^{AP} is the set of input alphabets; $\delta : Q \times 2^{AP} \to 2^Q$ is a transition relation and $\mathcal{F} \subseteq Q$ is a set of accepting states. There are fast translation algorithms [22] to obtain $\mathcal{A}_\varphi$. Moreover, denote by $\chi(q_m, q_n) = \{\ell \in 2^{AP} \mid q_n \in \delta(q_m, \ell)\}$ the set of all input alphabets that enable the transition from q_m to q_n in δ. Then the distance between $\ell \in 2^{AP}$ and $\chi \subseteq 2^{AP}(\chi \neq \emptyset)$ is defined by $\texttt{Dist}(\ell, \chi) = 0$ if $\ell \in \chi$ and $\min_{\ell' \in \chi} |\{a \in AP \mid a \in \ell, a \notin \ell'\}|$, otherwise. Namely, it returns the minimal difference between ℓ and any element in χ.

III. PROBLEM FORMULATION

A. Dynamic Workspace and Motion Abstraction

The bounded workspace where the robot is deployed is denoted by $\mathcal{W} \subset \mathbb{R}^2$. It consists of $N > 0$ regions of interest, denoted by $\Pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$, where $\pi_n \subset \mathcal{W}$. Furthermore, there is a set of $M > 0$ properties (atomic propositions) associated with Π, denoted by $AP = \{a_0, a_1, \cdots, a_M\}$, e.g., "this is a public area", "this is office room one" and "this meeting room is in use".

The robot's motion within the workspace is abstracted as a labeled transition system $\mathcal{T} \triangleq (\Pi, \to, \Pi_0, AP, L)$, where Π, AP are defined above, $\to \subseteq \Pi \times \Pi$ is the transition relation that $(\pi_i, \pi_j) \in \to$ if the robot can move from region π_i to region π_j without crossing other regions in Π, $\Pi_0 \in \Pi$ is where the robot starts initially, $L : \Pi \to 2^{AP}$ is the labeling function where $L(\pi_i)$ returns the set of properties satisfied by π_i. Since the workspace is assumed to be only *partially-known* and dynamic, the labeling function and the transition relation may change over time.

B. Mixed-initiative Controller

For the simplicity of discussion, we assume that the robot satisfies the single-integrator dynamics, i.e., $\dot{x} = u$, where $x, u \in \mathbb{R}^2$ are the robot position and control inputs. For each transition $(\pi_s, \pi_g) \in \to$, the robot is controlled by the mixed-initiative navigation controller [4] below:

$$u \triangleq u_r(x, \pi_s, \pi_g) + \kappa(x, \Pi)\, u_h(t) \qquad (1)$$

where $u_r(x, \pi_s, \pi_g) \in \mathbb{R}^2$ is a given autonomous controller that navigates the robot from region π_s to π_g, while staying within $\mathcal{W}$ and without crossing other regions in Π; the function $\kappa(x, \Pi) \in [0, \ 1]$ is a smooth function to be designed; and $u_h(t) \in \mathbb{R}^2$ is the human input function, which is *uncontrollable* and *unknown* by the robot.

Remark 1: The proposed motion and task coordination scheme can be readily extended to robotic platforms with other dynamics and different navigation controllers, such as potential-field-based [4] and sampling-based [7], [23]. ∎

C. Robot Task Assignment

The robot is assigned by the human operator a local task as LTL formulas over AP, which has the following structure:

$$\varphi \triangleq \varphi_{\text{hard}} \wedge \varphi_{\text{soft}} \wedge \varphi_{\text{temp}} \qquad (2)$$

where φ_{soft} and φ_{hard} are "soft" and "hard" sub-formulas that are assigned *offline*. Particularly, φ_{hard} includes safety constraints such as collision avoidance: "avoid all obstacles" or power-supply guarantees: "visit the charging station infinitely often"; φ_{soft} contains additional requirements for performance such as surveillance: "surveil all bases infinitely often". Introducing soft and hard constraints is due to the observation that the partially-known workspace might render parts of the task infeasible initially and thus yielding the need for them to be relaxed, while the safety-critical parts should not be relaxed; Lastly, φ_{temp} contains short-term contingent tasks that are assigned as sc-LTL formulas *online* and unknown beforehand. The structure in (2) provides an effective way for the operator to handle both standard operational tasks and contingent demands.

D. Control Objective

Given the abstraction model $\mathcal{T}$ and the task formula φ, the control objective is to design function $\kappa(\cdot)$ and control input u in (1) such that: (I) the hard constraints in φ_{hard} are always satisfied, given all possible human inputs; (II) each time a temporary task φ_{temp} is assigned, it is satisfied in finite time; and (III) the satisfaction of the soft constraints in φ_{soft} adapts to the human inputs.

IV. ALGORITHMIC COMPONENTS

In this section, we present the four algorithmic components of the overall solution presented in Section V. Particularly, we start from constructing a parameterized product automaton for the plan synthesis. Then we present a mixed-initiative controller that guarantees safety and meaningful inputs from the operator. Furthermore, we discuss a plan adaptation algorithms for real-time updates of the workspace model and contingent task assignment. At last, we describe a IRL algorithm to learn about the human preference.

A. Initial Discrete Plan Synthesis

Denote by $\mathcal{A}_{\text{hard}} = (Q_1, 2^{AP}, \delta_1, Q_{1,0}, \mathcal{F}_1)$ and $\mathcal{A}_{\text{soft}} = (Q_2, 2^{AP}, \delta_2, Q_{2,0}, \mathcal{F}_2)$ as the NBAs associated with φ_{hard} and φ_{soft}, respectively, where the notations are defined analogously as in Section II-B. Now we propose a way to compose $\mathcal{T}$, $\mathcal{A}_{\text{hard}}$ and $\mathcal{A}_{\text{soft}}$ into a product automaton.

Definition 1: The *parameterized* product automaton $\mathcal{A}_p \triangleq (Q_p, \delta_p, Q_{p,0}, \mathcal{F}_p)$ is defined as: $Q_p = \Pi \times Q_1 \times Q_2 \times \{1, 2\}$ are the states with $q_p = \langle \pi, q_1, q_2, c \rangle \in Q_p$, $\forall \pi \in \Pi$, $\forall q_1 \in Q_1$, $\forall q_2 \in Q_2$ and $\forall c \in \{1, 2\}$; $\delta_p : Q_p \times Q_p \to (\mathbb{R}_{\geq 0} \cup \{\infty\})^3$ maps each transition to a column vector such that $\delta_p(\langle \pi, q_1, q_2, c \rangle, \langle \check{\pi}, \check{q}_1, \check{q}_2, \check{c} \rangle) = [\alpha_1, \alpha_2, \alpha_3]^\mathsf{T}$, where

- α_1 is the control cost for the robot to move from π to $\check{\pi}$, where $\alpha_1 > 0$ if $(\pi, \check{\pi}) \in \to$, otherwise $\alpha_1 \triangleq \infty$;
- α_2 is the indicator for whether a transition violates the hard constraints. It satisfies that $\alpha_2 \triangleq 0$ if the following conditions hold: (i) $L(\pi) \in \chi_1(q_1, \check{q}_1)$; (ii) $\chi_2(q_2, \check{q}_2) \neq \emptyset$; (iii) $q_1 \notin \mathcal{F}_1$ and $\check{c} = c = 1$; or $q_2 \notin \mathcal{F}_2$ and $\check{c} = c = 2$; or $q_1 \in \mathcal{F}_1$, $c = 1$ and $\check{c} = 2$; or $q_2 \in \mathcal{F}_2$, $c = 2$ and $\check{c} = 1$. Otherwise, $\alpha_2 \triangleq \infty$.
- α_3 is the measure of how much a transition violates the soft constraints, where $\alpha_3 \triangleq \mathtt{Dist}(L(\pi), \chi_2(q_2, \check{q}_2))$, where the functions $\mathtt{Dist}(\cdot)$ and $\chi_2(\cdot)$ for $\mathcal{A}_{\text{soft}}$ are defined in Section II-B.

and $Q_{p,0} = \Pi_0 \times Q_{1,0} \times Q_{2,0} \times \{1\}$, $\mathcal{F}_p = \Pi \times \mathcal{F}_1 \times Q_2 \times \{1\}$ are the sets of initial and accepting states, respectively. ∎

An accepting run of $\mathcal{A}_p$ is an infinite run that starts from any initial state and intersects with the accepting states infinitely often. Note that the component c above to ensure that an accepting run intersects with the accepting states of both $\mathcal{A}_{\text{hard}}$ and $\mathcal{A}_{\text{soft}}$ infinitely often. More details can be found in Chapter 4 of [5]. Furthermore, since the workspace $\mathcal{T}$ is partially-known, we denote by $\mathcal{T}^t$ the workspace model at time $t \geq 0$, and the associated product automaton by $\mathcal{A}_p^t$.

To simplify the notation, given a finite run $R = q_p^0 q_p^1 \cdots q_p^S$ of $\mathcal{A}_p$, where $q_p^s \in Q_p$, $\forall s = 0, 1, \cdots, S$, we denote by $\boldsymbol{\delta}(R) = \sum_{s=0}^{S-1} \delta_p(q_p^s, q_p^{s+1})$, where $\boldsymbol{\delta}(R) \in \mathbb{R}^3$ is the accumulated cost vector δ_p along R. Similar definitions hold for $\boldsymbol{\alpha}_k(R) \in \mathbb{R}$ as the accumulated α_k cost along R, $\forall k = 1, 2, 3$. We consider an accepting run of $\mathcal{A}_p$ with the prefix-suffix structure: $R_p \triangleq q_p^1 q_p^2 \cdots q_p^S \big(q_p^{S+1} q_p^{S+2} \cdots q_p^{S+F} \big)^\omega$, where $q_p^j \in Q_p$, $\forall j = 1, 2, \cdots, S + F$, where $S, F > 0$. The plan prefix $R_p^{\text{pre}} \triangleq q_p^1 q_p^2 \cdots q_p^S$ is executed only once while the plan suffix $R_p^{\text{suf}} \triangleq q_p^{S+1} q_p^{S+2} \cdots q_p^{S+F}$ is repeated infinitely often. Then the total cost of R_p is defined as:

$$\mathtt{C}_\beta(R_p) \triangleq [1, \; \gamma] \otimes \begin{bmatrix} 1 \\ 1 \\ \beta \end{bmatrix}^\mathsf{T} \cdot \begin{bmatrix} \boldsymbol{\delta}(R_p^{\text{pre}}) \\ \boldsymbol{\delta}(R_p^{\text{suf}}) \end{bmatrix}, \quad (3)$$

where $\mathtt{C}_\beta(R_p) \geq 0$; $\otimes$ is the Kronector product; $\gamma \geq 0$ is a weighting parameter between the cost of the plan prefix and suffix; $\beta \geq 0$ is a weighting parameter between total control cost of the plan and the satisfaction of the soft task φ_{soft}. Note that γ is normally constant [11] (set to 1 in this work), while β can change according to the robot's internal model or the operator's *preference*. For instance, as the robot has more accurate workspace model, β can be increased to penalize the violation of φ_{soft} such that R_p satisfies φ_{soft} more. Or the operator prefers that β is decreased so that R_p satisfies φ_{soft} less and the robot reserves more power.

Given the initial values of γ and β, an initial accepting run of $\mathcal{A}_p$, denoted by R_p^0, can be found that minimizes

the total cost in (3). The algorithms are based on the nested Dijkstra's search, which are omitted here and details can be found in [11]. As a result, the robot's initial plan, denoted by τ_r^0, can be derived by projecting R_p^0 onto Π, as a sequence of regions that the robot should reach: $\tau_r^0 = \pi^1 \pi^2 \cdots \pi^S \big(\pi^{S+1} \pi^{S+2} \cdots \pi^{S+F} \big)^\omega$, where π^j is the projection of q_p^j onto Π, $\forall j = 1, 2, \cdots, S + F$.

B. Mixed-initiative Controller Design

After the system starts, the robot executes the initial plan τ_r^0 by reaching the sequence of regions defined by it. However, as described in Section III-B, the robot controller is also influenced by the human input. In the following, we show how to construct function $\kappa(\cdot)$ in (1) such that the hard task φ_{hard} is respected at all times for all human inputs.

First, we need to find the set of product states $\mathcal{O}_t \subset Q_p$ in $\mathcal{A}_p^t$ at time $t \geq 0$, such that once the robot belongs to any state in $\mathcal{O}_t$ it means that φ_{hard} can not be satisfied any more.

Lemma 1: Assume that the robot belongs to state $q_p \in Q_p$ at time $t > 0$. Then the hard task φ_{hard} can not be satisfied in the future, if $\mathcal{A}_p^t$ remains unchanged and the minimal cost of all paths from q_p to any accepting state in $\mathcal{F}_p$ is ∞.

Proof: Omitted as it is a simple inference of (3). ∎

Thus denote by $Q_t \subset Q_p$ the set of *reachable* states by the robot at time $t > 0$. For each $q_p \in Q_t$, we perform a Dijkstra search to compute the shortest distance from q_p to all accepting states in $\mathcal{F}_p$. Lastly, $\mathcal{O}_t$ is given as the subset of Q_t that have an infinite cost to *all* accepting states, i.e.,

$$\mathcal{O}_t = \{ q_p \in Q_t \,|\, \mathtt{C}_\beta(\overline{R}_{q_p, q_F}) = \infty, \forall q_F \in \mathcal{F}_p \}, \quad (4)$$

where $\overline{R}_{q_p, q_F}$ is the shortest path from q_p to q_F.

Given $\mathcal{O}_t$ above, we now design the function $\kappa(x, \Pi)$ in (1) such that $\mathcal{O}_t$ can be avoided. Consider the function:

$$\kappa(x, \Pi) = \kappa(x, \mathcal{O}_t) \triangleq \frac{\rho(d_t - d_s)}{\rho(d_t - d_s) + \rho(\varepsilon + d_s - d_t)} \quad (5)$$

where $d_t \triangleq \min_{\langle \pi, q_1, q_2, c \rangle \in \mathcal{O}_t} \|x - \pi\|$ is the minimum distance between the robot and any region within $\mathcal{O}_t$; $\rho(s) \triangleq e^{-1/s}$ for $s > 0$ and $\rho(s) \triangleq 0$ for $s \leq 0$, and $d_s, \varepsilon > 0$ are design parameters as the safety distance and a small buffer. Thus the mixed-initiative controller is given by

$$u \triangleq u_r(x, \pi_s, \pi_g) + \kappa(x, \mathcal{O}_t) \, u_h(t). \quad (6)$$

As discussed in [4], the function $\kappa(\cdot)$ above is 0 on the boundary of undesired regions in $\mathcal{O}_t$, and close to 1 when the robot is away from $\mathcal{O}_t$ to allow for meaningful inputs from the operator. This degree of closeness is tunable via changing d_t and d_s. However, the original definition in [4] only considers static obstacles, instead of the general set $\mathcal{O}_t$.

Lemma 2: Assume that the navigation control u_r is perpendicular to the boundary of regions in $\mathcal{O}_t$ and points inwards. Then the robot can avoid $\mathcal{O}_t$ under the mixed-initiative controller by (6) for all human input u_h.

Proof: The proof follows from Proposition 3 of [4]. Namely, for $x \in \partial \mathcal{O}_t$ on the boundary of $\mathcal{O}_t$,

$$\dot{x}^\mathsf{T} u_r = \|u_r(x)\|^2 + \kappa(x) \, u_h(t)^\mathsf{T} u_r(x) > 0$$

since $\kappa(x) = 0$ for $x \in \partial\mathcal{O}_t$. Thus the workspace excluding all regions in $\mathcal{O}_t$ is positive invariant under controller (6). In other words, if the navigation control avoids $\mathcal{O}_t$, the same property is ensured by (6) for all human inputs. ∎

C. Discrete Plan Adaptation

In this section, we describe how the discrete plan τ_r^t at $t \geq 0$ can be updated to (i) accommodate changes in the partially-known workspace model, and (ii) fulfill contingent tasks that are assigned by the operator during run time.

1) Updated Workspace Model: The robot can explore new features of the workspace while executing the discrete plan τ_r^t or being guided by the human operator. Thus the motion model $\mathcal{T}^t$ can be updated as follows: (i) the transition relation $\rightarrow$ is modified based on the status feedback from the navigation controller, i.e., whether it is feasible to navigate from region π_i to π_j without human inputs; (ii) the labeling function $L(\pi)$ is changed based on the feedback from the robot's sensing module, i.e., the properties that region π satisfies. For example, "the corridor connecting two rooms is blocked" or "the object of interest is no longer in one room". Given the updated $\mathcal{T}^t$, the mapping function δ_p of the product automaton $\mathcal{A}_p^t$ is re-evaluated. Consequently, the current plan τ_r^t might not be optimal anymore regarding the cost in (3). Thus we consider the following problem.

Problem 1: Update τ_r^t such that it has the minimum cost in (3) given the updated model $\mathcal{T}^t$. ∎

Given the set of reachable states $Q_t \subset Q_p$ at time $t > 0$, for each state $q_p \in Q_t$, we perform a nested Dijkstra's search [11] to find the accepting run that starts from q_p and has the prefix-suffix structure with the minimum cost defined in (3). Denote by $R_+^t(q_p)$ this optimal run for $q_p \in Q_t$. Moreover, let $\mathbf{R}_+^t \triangleq \{R_+^t(q_p), \forall q_p \in Q_t\}$ collect all such runs. Then we find among $\mathbf{R}_+^t$ the accepting run with the minimum cost, which becomes the updated run R_p^{t+}:

$$R_p^{t+} \triangleq \mathbf{argmin}_{R_p \in \mathbf{R}_+^t} \mathsf{C}_\beta(R_p). \qquad (7)$$

Thus the updated plan τ_r^t is given by the projection of R_p^{t+} onto Π. Note that the above procedure is performed whenever the motion model $\mathcal{T}^t$ is updated.

2) Contingent Task Fulfillment: As defined in (2), the operator can assign contingent and short-term tasks φ_{temp} to the robot during run time. Particularly, we consider the following "*pick-up and deliver*" task with deadlines, i.e.,

$$(\varphi_{\text{temp}}^t, T_{sg}) \triangleq (\Diamond(\pi_s \wedge \Diamond\pi_g), T_{sg}), \qquad (8)$$

where $\varphi_{\text{temp}}^t \triangleq \Diamond(\pi_s \wedge \Diamond\pi_g)$ is the temporary task assigned at time $t > 0$, meaning that the robot needs to pick up some objects at region π_s and deliver them to π_g (note that action propositions are omitted here, see [12]), where $\pi_s, \pi_g \in \Pi$; $T_{sg} > 0$ is the *preferred* deadline that task φ_{temp}^t is accomplished. It can be verified that φ_{temp}^t are sc-LTL formulas and can be fulfilled in finite time. Assume that φ_{temp}^t is satisfied at time $t' > t$ then the delay is defined as $\bar{t}_{sg} \triangleq t' - T_{sg}$. We consider to following problem to incorporate φ_{temp}^t.

Problem 2: Update R_p^t such that φ_{temp}^t can be fulfilled *without* delay (if possible) and with *minimum* extra cost, while respecting the hard constraints φ_{hard}. ∎

Assume the remaining optimal run at time $t > 0$ is given by $R_p^t = q_p^{k_0} q_p^{k_0+1} \cdots (q_p^S \cdots q_p^{S+F})^\omega$ and $q_p^{k_s}, q_p^{k_g} \in Q_p$ are the k_s-th, k_g-th state of R_p^t, where $k_s \geq k_g \geq k_0$. Since R_p^t is optimal for the current $\mathcal{T}^t$, we search for the index k_s where the robot can deviate from R_p^t to reach region π_s and back, and another index k_g where the robot can deviate from R_p^t to reach π_g and back. Denote by R_p^{t+} the updated run after incorporating π_s, π_g. In this way, φ_{temp}^t is satisfied when π_g is reached after π_s is reached, where $t' = \sum_{j=k_0}^{k_g} \alpha_1(q_p^j, q_p^{j+1})$ is the total time. Moreover, the total cost of R_p^t in (3) is changed by $\overline{\mathsf{C}}_\beta(R_p^t) \triangleq \mathsf{C}_\beta(R_p^{t+}) - \mathsf{C}_\beta(R_p^t)$. Thus we formulate a 2-stage optimization below: first, we solve

$$\overline{d}_{sg} = \mathbf{min}_{\{k_g > k_s \geq 0\}} \{\overline{\mathsf{C}}_\beta(R_p^t)\}, \quad \textbf{s.t.} \quad \overline{t}_{sg} \leq 0, \qquad (9a)$$

in order to find out whether it is possible to avoid delay while satisfying φ_{temp}. If no solution is found, we solve the relaxed optimization that allows the deadline to be missed:

$$\overline{d}_{sg} = \mathbf{min}_{\{k_g > k_s \geq 0\}} \{\overline{t}_{sg}' + \overline{\mathsf{C}}_\beta(R_p^t)\}, \qquad (9b)$$

where $\overline{d}_{sg} \geq 0$; $\overline{t}_{sg}' = 0$ if $\overline{t}_{sg} \leq 0$ and $\overline{t}_{sg}' = \overline{t}_{sg}$, otherwise. Note that $\overline{\mathsf{C}}_\beta(R_p^t)$ is ∞ if φ_{hard} is violated by R_p^{t+}.

Since the suffix of R_p^t is repeated infinitely often, the choice of indices k_s, k_g for (9) is finite. Thus (9) can be solved as follows: starting from $k_s = k_0$, we iterate through $k_g \in \{k_0 + 1, \cdots, S + F\}$ and compute the corresponding $\overline{t}_{sg}$ and $\overline{d}_{sg}$ for both cases in (9). Then we increase k_s incrementally by $k_s = k_0 + 1$, iterate through $k_g \in [k_0 + 2, \cdots, S + F]$ and compute $\overline{t}_{sg}$ and $\overline{d}_{sg}$ for both cases. This procedure repeats itself *until* $k_s = S + F - 1$. Then, we find among these candidates if there is a pair $k_s^\star, k_g^\star$ that solves (9a). If so, they are the optimal choice of k_s, k_g. Otherwise, we search for the optimal solution to (9b), of which the solution always exists as it is unconstrained. At last, R_p^{t+} is derived by inserting the product states associated with π_s, π_g at indices $k_s^\star, k_g^\star$ of R_p^t, respectively.

D. Human Preference Learning

As discussed in Section IV-B, the mixed-initiative controller (6) allows the operator to interfere the robot's trajectory such that it deviates from its discrete plan τ_r^t, while always obeying φ_{hard}. This is beneficial as the robot could be guided to (i) explore *unknown* features to update its workspace model, as described in Section IV-C.1; and (ii) follow the trajectory that is *preferred* by the operator.

Particularly, as discussed in Section IV-A, the initial run R_p^0 is a balanced plan between reducing the control cost and improving the satisfaction of φ_{soft}, where the weighting parameter is β in (3). Clearly, different choices of β may result in different R_p^0. The initial plan R_p^0 is synthesized under the initial value $\beta_0 \geq 0$, which however might *not* be what the operator prefers. In the following, we present how the robot could learn about the preferred β from the operator's inputs during run time.

Consider that at time $t \geq 0$, the robot's past trajectory is given by $\zeta|_0^t \triangleq \pi_0 \pi_1 \cdots \pi_{k_t}$. Assume now that during time $[t, t']$, where $t' > t > 0$, via the mixed-initiative controller in (6), the operator guides the robot to reach a sequence of regions that s/he prefers, which is defined by:

$$\zeta_h|_t^{t'} \triangleq \pi_1' \pi_2' \cdots \pi_H' \tag{10}$$

where $\pi_h' \in \Pi$, $\forall h = 1, 2 \cdots H$ and $H \geq 1$ is the length of ζ_h that can vary each time the operator acts. Afterwards, the robot continues executing its current plan τ_r^t. Thus, the actual robot trajectory until time t' is given by $\zeta_h|_0^{t'} \triangleq \zeta|_0^t \zeta_h|_t^{t'}$, which is the concatenation of $\zeta|_0^t$ and $\zeta_h|_t^{t'}$.

Problem 3: Given the actual robot trajectory $\zeta_h|_0^{t'}$, design an algorithm to estimate the preferred value of β as $\beta_h^\star$ such that $\zeta_h|_0^{t'}$ corresponds to the optimal plan under $\beta_h^\star$. ∎

The above problem is closely related to the inverse reinforcement learning (IRL) problem [24], [25], where the robot learns about the cost functions of the system model based on demonstration of the preferred plans. On the other hand, in reinforcement learning [26], [27] problem, the robot learns the optimal plan given these functions.

As mentioned in [24], most problems of IRL are ill-posed. In our case, it means that there are more than one $\beta_h^\star$ that render $\zeta_h|_0^{t'}$ to be the optimal plan under $\beta_h^\star$. In order to improve *generalization* such that the robot could infer the human preference based on the human's past inputs (instead of simply repeating them), our solution is based on the maximum margin planning algorithm from [25]. The general idea is to iteratively update β via a sub-gradient descent, where the gradient is computed based on the difference in cost between $\zeta_h|_0^{t'}$ and the optimal plan under the current β.

First, we compute the set of all finite runs within $\mathcal{A}_p^{t'}$, denoted by $\mathbf{R}_h^{t'}$, that are associated with $\zeta_h|_0^{t'}$. It can be derived iteratively via a breadth-first graph search [5]. Among $\mathbf{R}_h^{t'}$, we find the one with the minimal cost over α_3, i.e.,

$$R_h^\star \triangleq \mathbf{argmin}_{R \in \mathbf{R}_h^{t'}} \ \boldsymbol{\alpha}_3(R). \tag{11}$$

Let $R_h^\star \triangleq q_1 q_2 \cdots q_H$, where $q_h \in Q_p$, $\forall h = 1, 2, \cdots, H$. Denote by β_k the value of β at the k-th iteration, for $k \geq 0$. Note that $\beta_0 \triangleq \beta_t$, where β_t is the value of β at time $t > 0$. For the k-th iteration, we find the optimal run from q_1 to q_H under β_k with certain margins, i.e.,

$$\hat{R}_{\beta_k}^\star \triangleq \mathbf{argmin}_{R \in \mathbf{R}_{q_1 q_H}} \left(\mathsf{C}_{\beta_k}(R) - M(R, R_h^\star) \right) \tag{12}$$

where $\mathbf{R}_{q_1 q_H}$ is the set of *all* runs from q_1 to q_H in $\mathcal{A}_p^{t'}$; and $M : Q^H \times Q^H \to \mathbb{N}$ is the margin function [25]:

$$M(R, R_h^\star) = |\{(q_s, q_t) \in R \,|\, (q_s, q_t) \notin R_h^\star\}|, \tag{13}$$

which returns the number of edges within R that however do not belong to $R_h^\star$. The margin function decreases the total cost $\mathsf{C}_{\beta_k}(R)$ by the difference between R and $R_h^\star$. It can improve generalization and help address the ill-posed nature of Problem 3. To solve (12), we first modify $\mathcal{A}_p^{t'}$ by reducing the α_1 cost of each edge $(q_s, q_t) \in R_h^\star$ by one. Then a Dijkstra shortest path search can be performed over

---

Algorithm 1: On-line IRL algorithm for β.

Input: $\mathcal{A}_p^t$, $\zeta_h|_0^{t'}$, β_t, ε
1 Initialize $\beta_k = \beta_t$ for iteration $k = 0$;
2 **while** $|\beta_{k+1} - \beta_k| > \varepsilon$ **do**   // Iteration k
3   Compute $R_h^\star$ in (11) given $\zeta_h|_0^{t'}$;
4   Find $\hat{R}_{\beta_k}^\star$ in (12) given β_k and $R_h^\star$;
5   Compute $\nabla \beta_k$ by (14) and update β_k by (15);
6 **return** $\beta_t^+ = \beta_{k+1}$

---

the modified $\mathcal{A}_p$ to find the shortest run from q_1 to q_H that minimizes the cost with margins in (12). Given $\hat{R}_{\beta_k}^\star$, we can compute the sub-gradient [28] that β_k should follow:

$$\nabla \beta_k = \lambda \cdot \beta_k + \left(\boldsymbol{\alpha}_3(R_h^\star) - \boldsymbol{\alpha}_3(\hat{R}_{\beta_k}^\star) \right), \tag{14}$$

where $\nabla \beta_k \in \mathbb{R}$ and $\lambda > 0$ is a design parameter. Thus, at this iteration the value of β_k is updated by

$$\beta_{k+1} = \beta_k - \theta_k \cdot \nabla \beta_k, \tag{15}$$

where $\theta_k > 0$ is the step size or learning rate [26]. Given the updated β_{k+1}, the same process in (11)-(15) is repeated until the difference $|\beta_{k+1} - \beta_k|$ is less than a predefined threshold $\varepsilon > 0$. At last, the value of β_t is updated to β_{k+1}. The discussion above is summarized in Alg. 1. Each time the human operator guides the robot to reach a new sequence of regions, the estimation of the value of β_t is updated by running Alg. 1. In the following, we show that Alg. 1 ensures the convergence of $\{\beta_k\}$.

Lemma 3: The sequence $\{\beta_k\}$ in Alg. 1 converges to a fixed $\beta_l^\star \geq 0$ and the optimal plan under $\beta_l^\star$ is $\zeta_h|_0^{t'}$.

Proof: Firstly, the optimal run $R_h^\star$ associated with $\zeta_h|_0^{t'}$ under $\beta_h^\star$ minimizes the balanced cost C_β from (3), i.e.,

$$\mathsf{C}_\beta(R_h^\star) \leq \mathsf{C}_\beta(R), \quad \forall R \in \mathbf{R}_{q_1 q_H}, \tag{16}$$

where $\mathbf{R}_{q_1 q_H}$ is defined in (12). Solving (16) directly can be computationally expensive due to the large set $\mathbf{R}_{q_1 q_H}$. We introduce a slack variable $\xi \in \mathbb{R}$ to relax the constraints:

$$\begin{aligned} \mathbf{min}_{\beta \geq 0} \quad & \frac{\lambda}{2}\beta^2 + \xi \\ \mathbf{s.t.} \quad & \mathsf{C}_\beta(R_h^\star) - \xi \leq \min_{R \in \mathbf{R}_{q_1 q_H}} \left(\mathsf{C}_\beta(R) - M(R, R_h^\star) \right), \end{aligned} \tag{17}$$

where $\lambda > 0$ is the same as in (14) and the margin function $M(\cdot)$ is from (13). Thus, by enforcing the slack variables to be tight, β also minimizes the combined cost function:

$$\frac{\lambda}{2}\beta^2 + \mathsf{C}_\beta(R_h^\star) - \min_{R \in \mathbf{R}_{q_1 q_H}} \left(\mathsf{C}_\beta(R) - M(R, R_h^\star) \right), \tag{18}$$

which is convex but non-differentiable. Instead, we compute the sub-gradient [28] of (18): $\nabla \beta = \lambda \beta + \left(\boldsymbol{\alpha}_3(R_h^\star) - \boldsymbol{\alpha}_3(\hat{R}_\beta^\star) \right)$ and $\hat{R}_\beta^\star = \mathbf{argmin}_{R \in \mathbf{R}_{q_1 q_H}} (\mathsf{C}_\beta(R) - M(R, R_h^\star))$, which is equivalent to (14).

Lastly, by the strong convexity of (18) and Theorem 1 of [25], the estimation β_t approaches the optimal $\beta_l^\star$ with linear convergence rate under constant stepsize $\theta_k = \theta$, i.e.,

Method	$|\mathcal{A}_p|$	$\beta_l^\star$	NO. of Dijkstra	Time[s]
Alg.1	25	13.4	8	3.8
M1	25	10.0	200	124.4
M2	25	11.7	350	337.2
Alg.1	100	16.5	12	150.8
M1	100	14.2	200	2203.5
M2	100	–	800+	3000+

TABLE I: Comparison of computational complexity and performance of Alg. 1 and two alternative methods in Example 1.

$|\beta_k - \beta_l^\star|^2 \leq (1-\theta\lambda)^{k+1}|\beta_0 - \beta_l^\star|^2 + \frac{\theta|\nabla\beta|_{\max}}{\lambda}$. A detailed analysis on this deviation can be found in [25], [28]. ∎

Remark 2: It is worth noting that the convergent value $\beta_l^\star$ might be *different* from the preferred $\beta_h^\star$, while they both satisfy (16) with the same optimal run $R_h^\star$. However, the margin function in (17) ensures that $\beta_l^\star$ is *better* than or at least equivalent to $\beta_h^\star$ in terms of the similarity between $R_h^\star$ and the run with the second minimum cost by (3).

Now we show the computational efficiency of Alg. 1 compared with two straight-forward solutions: (M1) choose the optimal β among a set of guessed values of β, denoted by S_β; (M2) solve (16) directly by enumerating all runs in $\mathbf{R}_{q_1 q_H}$. The first method's accuracy relies on S_β being large, which however results in high computational cost. Similarly, the second method relies on evaluating *every* run in $\mathbf{R}_{q_1 q_H}$, the size of which is combinatorial to the size of $\mathcal{A}_p^t$. The following example shows some numerical comparison.

Example 1: Assume that $\beta_h^\star = 15$ and initially $\beta_0 = 0$. We use three methods: Alg. 1, M1 and M2 above to estimate $\beta_h^\star$. As shown in Table I, we compare the final convergence $\beta_l^\star$ and the computation time under varying sizes of $\mathcal{A}_p$. It can be seen that the computation time for Alg. 1 is significantly less than M1 and M2, where for the second case M2 fails to converge within 50min. ∎

V. The Integrated System

In this section, we describe the the real-time execution of the integrated system given the components in Section IV. Then we discuss the computational complexity.

A. Human-in-the-loop Motion and Task Planning

The complete algorithm is shown in Alg. 2. Before the system starts, given the initial model $\mathcal{T}^0$ and the task formulas in (2), the initial plan τ_r^0 is synthesized by the algorithm from Section IV-A under the initial β_0. From $t = 0$, the robot executes τ_r^0 by following the sequence of goal regions, see Lines 1-3. Meanwhile, the operator can directly modify the control input $u(\cdot)$ via (6) to change the robot's trajectory. Thus, the robot can explore regions that are not in its initial plan and update the model $\mathcal{T}^t$ as described in Section IV-C.1. As a result, the plan τ_r^t is updated by (7) accordingly, see Lines 4-5. Moreover, as described in Section IV-C.2, the operator can assign temporary tasks with deadlines as in (8), for which τ_r^t is modified by solving (9), see Lines 6-7. Last but not least, each time the operator guides the robot to follow a new trajectory, the parameter β is updated via

---

Algorithm 2: Mixed-initiative Motion and Task Planning

Input: $\mathcal{T}^t$, φ_{hard}, φ_{soft}, β_0, $u_h(t)$, $(\varphi_{\text{temp}}^t, T_{sg})$

1 Compute $\mathcal{A}_p^0$ and construct initial plan τ_r^0 under β_0;

2 **forall** $t \geq 0$ **do**

3   Compute $u_r(\cdot)$ in (6) to reach next $\pi^j \in \tau_r^t$;

4   **if** $\mathcal{T}^t$ *updated* **then**   // Model update

5   Update product $\mathcal{A}_p^t$ and plan τ_r^t by (7);

6   **if** $(\varphi_{\text{temp}}^t, T_{sg})$ *received* **then**   // Temp. task

7   Update plan τ_r^t by solving (9);

8   **if** $\|u_h(t)\| > 0$ **then**   // Human input

9   Compute control $u(t)$ by (6);

10   Compute $\zeta_h|_0^t$ by (10);

11   Learn $\beta_l^\star$ by Alg. 1 and set $\beta_t^+ = \beta_l^\star$;

12   Update τ_r^t by (7) given the learned β_t^+;

13   **return** $u(t)$, τ_r^t, β_t^+

Alg. 1 to estimate the human preference. Then, its current plan τ_r^t is updated using the updated β, see Lines 8-12. The above procedure repeats until the system is terminated.

Theorem 4: Alg. 2 above fulfills the three control objectives of Section III-D, i.e., (I) φ_{hard} is satisfied for all time; (II) each φ_{temp} is satisfied in finite time; and (III) the satisfaction of φ_{soft} adapts to the human inputs.

Proof: (I) Firstly, both the initial synthesis algorithm in Section IV-A and the plan adaptation algorithm in Section IV-C.1 ensure φ_{hard} is satisfied by minimizing the total cost in (3). Then Lemma 2 ensures that φ_{hard} is respected for all possible inputs from the human operator. (II) The combined cost (9) ensures that φ_{temp} is satisfied within finite time. (III) Convergence of the learning Alg. 1 is shown in Lemma (3). Thus, the updated plan τ_r^t under the learned value of β adapts to the plan preferred by the operator. ∎

B. Computational Complexity

The process to synthesize τ_r^t given $\mathcal{A}_p^t$ via Alg. 2 (in Line 1) and the plan revision given $\mathcal{T}^t$ (in Line 5) both have complexity $\mathcal{O}(|\mathcal{A}_p^t|^2)$ [11]. The adaptation algorithm for temporary tasks (in Line 7) has complexity $\mathcal{O}(|R_p^t|^2)$. Lastly, the learning Alg. 1 (in Line 11) has complexity $\mathcal{O}(|R_h^\star|^2)$, where $|R_h^\star|$ is the length of the optimal run from (11).

VI. Case Study

In this section, we present numerical studies both in simulation and experiment. The Robot Operation System (ROS) is used as the simulation and experiment platform. All algorithms are implemented in Python 2.7 and available online [29]. All computations are carried out on a laptop (3.06GHz Duo CPU and 8GB of RAM).

A. Simulation

1) Workspace and Robot Description: Consider the simulated office environment in Gazebo as shown in Fig. 1 with dimension $100m \times 45m$, in which there are 9 regions of interest (denoted by $r_0, \cdots, r_8$) and 4 corridors (denoted
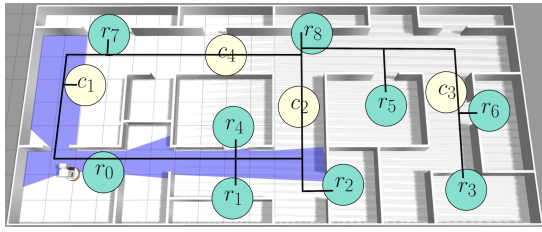
Fig. 1: Office environment in Gazebo with TIAGo robot, where the regions of interest and allowed transitions are marked.

Fig. 2: The robot's trajectory in simulation case One, where the robot's initial plan is in blue, the human-guided part is in red, and the updated plan is in green.

Fig. 3: The mixed control for linear velocity during two time periods when the human control (in red) is active.

Fig. 4: The robot's trajectory in simulation case Two. The robot's trajectory while performing the temporary task is in magenta.

by $c_1, \cdots, c_4$). The transition relations are determined by whether there exists a collision-free path from the center of one region to another, without crossing other regions.

We simulate the TIAGo robot from PAL robotics, of which the navigation control $u_r(\cdot)$ with obstacle avoidance, localization and mapping are all based on the ROS navigation stack. The human operator monitors the robot motion through Rviz. Moreover, the control $u_h(\cdot)$ from the operator can be generated from a keyboard or joystick, while the temporary task in LTL formulas φ_{temp} are specified via ROS messages. More details can be found in the software implementation [29] and simulation video [30].

2) Case One: The hard task for *delivery* is given by $\varphi_{1,\text{hard}} = \big(\Box\Diamond(r_0 \wedge \Diamond(r_7 \wedge \Diamond r_8))\big) \wedge \big(\Box\Diamond(r_2 \wedge \Diamond(r_3 \vee r_6))\big) \wedge \big(\Box\neg r_5\big)$, i.e., to transfer objects from r_0 to r_7 (then r_8) and from r_2 to r_3 (or r_6), while avoiding r_5 for all time. The soft task is $\varphi_{1,\text{soft}} = (\Box\neg c_4)$, i.e., to avoid c_4 if possible. It took $0.2s$ to compute the parameterized product automaton, which has 312 states and 1716 transitions. The parameter β is initially set to a large value 30, thus the initial plan satisfies both the soft and hard tasks but with a large cost due to the long traveling distance, as shown in Fig. 2. During $[700s, 950s]$, the operator drives the robot to go through corridor c_4 and reach r_8, which violates the soft task $\varphi_{1,\text{soft}}$. As a result, β is updated by Alg. 1 and the final value is 16.35 after 20 iterations with $\varepsilon = 0.2$, as shown in Fig. 5. Namely, the robot has learned that the operator allows *more violation* of the soft task to reduce the total cost. The resulting updated plan is shown in Fig. 2. Moreover, to demonstrate the ensured safety in Lemma 2, the human operator drives the robot towards r_5 during $[1250s, 1350s]$, which is not allowed by φ_{hard}. The weighting function $\kappa(\cdot)$ in the mixed controller (6) approaches 0. Thus the robot still follows its updated plan and avoids r_5. The mixed control inputs during these periods are shown in Fig. 3.

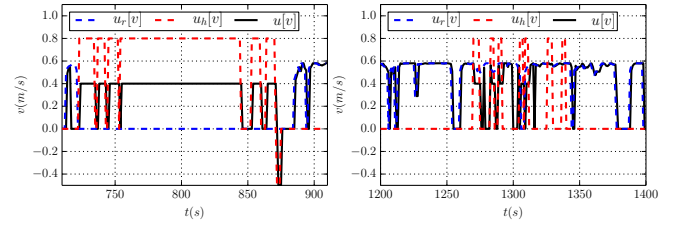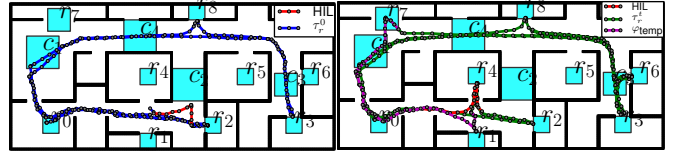*3) Case Two:* The hard task for *surveillance* is given by $\varphi_{2,\text{hard}} = (\Box\Diamond r_2) \wedge (\Box\Diamond r_3) \wedge (\Box\Diamond r_8)$, i.e., to surveil regions r_2, r_3 and r_8 infinitely often. The soft task for extra performance is $\varphi_{2,\text{soft}} = \Box\Diamond\big(r_4 \to (\neg r_5 \cup \Diamond r_6)\big)$, i.e., to collect goods from region r_4 and drop it at r_6 (without crossing r_5 before that). Moreover, the workspace model in this case is *different* from the initial model that the corridor c_2 has been blocked. By following Alg. 2, it took $0.17s$ to compute the product automaton, which has 418 states and 3360 transitions. Initially, $\beta = 0$ meaning that the initial plan τ_r^0 only satisfies $\varphi_{2,\text{hard}}$ while $\varphi_{2,\text{soft}}$ is fully relaxed. During $[150s, 250s]$, the operator drives the robot to sense that the corridor c_2 has been blocked. As a result, the discrete plan τ_r^0 is updated such that the robot chooses to reach r_8 from r_2 via c_1, as shown in Fig. 4. Afterwards, during $[1100s, 1200s]$, the operator drives the robot to r_4 after reaching r_2, which satisfies part of $\varphi_{2,\text{soft}}$. As a result, β is increased by Alg. 1 to 11.3 after 12 iterations with $\varepsilon = 0.1$, as shown in Fig. 5. Namely, the robot has learned that the soft task should be satisfied *more*. Lastly, at time $2100s$, the operator assigns a temporary task $\varphi_{\text{temp}} = \Diamond(r_1 \wedge \Diamond r_7)$ with a deadline $2700s$, i.e., to deliver an object from r_1 to r_7. This temporary task is incorporated into τ_r^t and is fulfilled at $2400s$, which is shown in Fig. 4.

B. Experiment

The experiment setup involves a TurtleBot within the office environment at Automatic Control Lab, KTH. Details are omitted here due to limited space, which are given in the the software implementation [29] and experiment video [30].

1) Workspace and Task Specification: The office environment consists of three office rooms (r_1, r_2, r_3) and one corridor r_0, as shown in Fig. 6. The robot's task specification is similar to case study Two above, i.e., the hard task is given by $\varphi_{\text{hard}} = \Box\Diamond r_0 \wedge \Box\Diamond r_1$ (to surveil regions r_0 and r_1) while the soft task is $\varphi_{\text{soft}} = \Box\Diamond r_2 \wedge \Box\Diamond r_3$ (to surveil regions r_2 and r_3). The TurtleBot is controlled via ROS navigation stack and behaves similarly to the TIAGo robot in Section VI-A.

2) Experiment Results: Since β is initially set to 0, the robot only surveils r_0 and r_1 for the hard task, as shown in
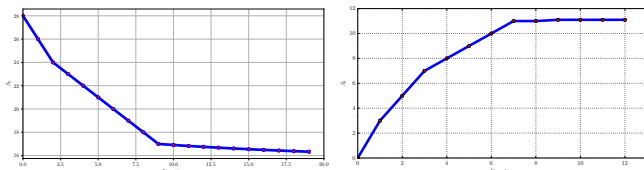
Fig. 5: Evolution of the learned value of β in simulation case One (left) and case Two (right).

Fig. 6: The human-in-the-loop experiment setup, where the robot is controlled by both its autonomous controller and the human inputs.

Fig. 7: The robot's trajectory in the experiment study, where the robot's initial plan is in blue (left), the human-guided segment is in red (middle), and the updated plan is in green (right).

Fig. 7. From $t = 59s$, the operator starts driving the robot towards r_2 and back to r_0 until $t = 137s$. As a result, the estimated β_t is updated by Alg. 1 given the robot's past trajectory. The final convergence value is 1.58 with $\varepsilon = 0.01$ after 15 iterations. Then updated plan is shown in Fig. 7 which intersects with not only regions r_0 and r_1 for the hard task, but also regions r_2 and r_3 for the soft task. Notice that the operator only needs to interfere the robot's motion for a small fraction of the operation time.

VII. SUMMARY AND FUTURE WORK

In this paper, we present a human-in-the-loop task and motion planning strategy for mobile robots with mixed-initiative control. The proposed coordination scheme ensures the satisfaction of high-level LTL tasks given the human initiative both through continuous control inputs and discrete task assignments. Future work includes consideration of multi-robot systems.

REFERENCES

[1] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 1, pp. 24–39, 2012.

[2] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and autonomous systems*, vol. 42, no. 3, pp. 143–166, 2003.

[3] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[4] S. G. Loizou and V. Kumar, "Mixed initiative control of autonomous vehicles," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2007, pp. 1431–1436.

[5] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008.

[6] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[7] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2010, pp. 2689–2696.

[8] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "Mdp optimal control under temporal logic constraints," in *Decision and Control (CDC), IEEE Conference on*, 2011, pp. 532–538.
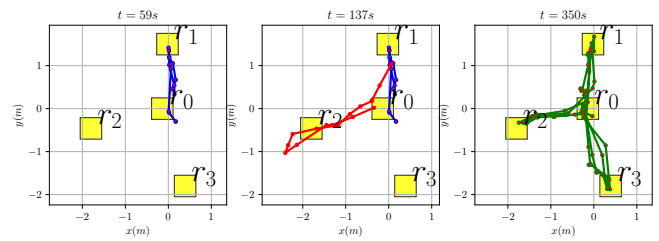
[9] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017, vol. 89.

[10] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[11] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[12] ——, "Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 797–808, 2017.

[13] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[14] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *Robotics, IEEE Transactions on*, vol. 25, no. 6, pp. 1370–1381, 2009.

[15] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu, "Synthesis of human-in-the-loop control protocols for autonomous systems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 450–462, 2016.

[16] J. Fu and U. Topcu, "Pareto efficiency in synthesizing shared autonomy policies with temporal logic constraints," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2015, pp. 361–368.

[17] N. Jansen, M. Cubuktepe, , and U. Topcu, "Synthesis of shared control protocols with provable safety and performance guarantees," in *In Proc. of the American Control Conference (ACC)*, 2017.

[18] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[19] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris, "Multi-objective control for multi-agent systems using lyapunov-like barrier functions," in *Decision and Control (CDC), IEEE Conference on*. IEEE, 2013, pp. 1478–1483.

[20] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *Decision and Control (CDC), IEEE Conference on*. IEEE, 2016, pp. 2659–2664.

[21] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[22] P. Gastin and D. Oddoux, "Fast LTL to büchi automata translation," in *Computer Aided Verification*. Springer, 2001, pp. 53–65.

[23] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[24] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[25] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 729–736.

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[27] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, 1st ed. Athena Scientific, 1996.

[28] N. Z. Shor, *Minimization methods for non-differentiable functions*. Springer Science & Business Media, 2012, vol. 3.

[29] Software, http://github.com/MengGuo/mix_initiative.

[30] Video, https://vimeo.com/230487800,232727691.