# 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], [7], a multi-robot system under a global task [8], or a multi-robot system under local independent [9] or dependent tasks [10], [11]. 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 [12] 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 [13] where they take turns to influence the system evolution. Another recent work [14] 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 [15], while ensuring safety of the overall system. The work in [16] 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.

## 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]: $\varphi ::= \top \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq \texttt{True}$, $a \in AP$ and $\bigcirc$ (*next*), $\mathsf{U}$ (*until*). There are other useful operators like $\Box$

The authors are with the ACCESS Linnaeus Center, School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Sweden (Email: mengg, sofa, dimos@kth.se). This work is supported in part by the H2020 ERC Starting Grant BUCOPHSYS, the EU H2020 Research and Innovation Programme under GA No.731869 (Co4Robots), SSF COIN project and the Knut and Alice Wallenberg Foundation (KAW).

(*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 [17].

## B. Büchi Automaton

Given a LTL formula $\varphi$, there always exists a Nondeterministic Büchi Automaton (NBA) $\mathcal{A}_\varphi$ that accepts all the languages that satisfy $\varphi$ [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 [18] to obtain $\mathcal{A}_\varphi$. Moreover, denote by $\chi(q_m,\, q_n) = \{\ell \in 2^{AP} \,|\, q_n \in \delta(q_m,\, \ell)\}$ the set of all input alphabets that enable the transition from $q_m$ to $q_n$ in $\delta$. 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 $\min_{\ell' \in \chi} \, |\{a \in AP \,|\, a \in \ell, a \notin \ell'\}|$, otherwise. Namely, it returns the minimal difference between $\ell$ and any element in $\chi$.

## 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 $\Pi$, 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 $\Pi$, $AP$ 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 $\pi_i$ to region $\pi_j$ without crossing other regions in $\Pi$, $\Pi_0 \in \Pi$ 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 $\pi_i$. 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 $\pi_s$ to $\pi_g$, while staying within $\mathcal{W}$ and without crossing other regions in $\Pi$; 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.

### 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 $\varphi_{\text{soft}}$ and $\varphi_{\text{hard}}$ are "soft" and "hard" sub-formulas that are assigned *offline*. Particularly, $\varphi_{\text{hard}}$ includes safety constraints such as collision avoidance: "avoid all obstacles" or power-supply guarantees: "visit the charging station infinitely often"; $\varphi_{\text{soft}}$ contains additional requirements for performance such as surveillance: "surveil all bases infinitely often". Lastly, $\varphi_{\text{temp}}$ contains short-term tasks that are assigned as sc-LTL formulas *online* and unknown beforehand.

### D. Control Objective

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

### 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 $\varphi_{\text{hard}}$ and $\varphi_{\text{soft}}$, 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 $\delta_p(\langle \pi,\, q_1,\, q_2, c\rangle, \langle \check{\pi},\, \check{q}_1,\, \check{q}_2, \check{c}\rangle) = [\alpha_1, \alpha_2, \alpha_3]^\intercal$, where

- $\alpha_1$ is the control cost for the robot to move from $\pi$ to $\check{\pi}$, where $\alpha_1 > 0$ if $(\pi,\, \check{\pi}) \in \to$, otherwise $\alpha_1 \triangleq \infty$;
- $\alpha_2$ 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, $\alpha_2 \triangleq \infty$.
- $\alpha_3$ is the measure of how much a transition violates the soft constraints, where $\alpha_3 \triangleq \texttt{Dist}(L(\pi),\, \chi_2(q_2, \check{q}_2))$, where the functions $\texttt{Dist}(\cdot)$ 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 [5], [9]. 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 $\delta_p$ along $R$. Similar definitions hold for $\boldsymbol{\alpha}_k(R) \in \mathbb{R}$ as the accumulated $\alpha_k$ cost along $R$, $\forall k = 1, 2, 3$. We consider an accepting run of $\mathcal{A}_p$ with the prefix-suffix structure: $R_p \triangleq R_p^{\text{pre}} \big(R_p^{\text{suf}}\big)^\omega \triangleq q_p^1 q_p^2 \cdots q_p^S \big(q_p^{S+1} q_p^{S+2} \cdots q_p^{S+F}\big)^\omega$, where $q_p^j \in Q_p$, $\forall j = 1, 2, \cdots, S+F$, where $S, F > 0$. The plan prefix $R_p^{\text{pre}}$ is executed only once while the plan suffix $R_p^{\text{suf}}$ is repeated infinitely often. Then the total cost of $R_p$ is defined as:

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

where $\texttt{C}_\beta(R_p) \geq 0$; $\otimes$ 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 $\varphi_{\text{soft}}$. Note that $\gamma$ is normally constant [9], while $\beta$ can change according to the robot's internal model or the operator's *preference*.

Given the initial values of $\gamma$ and $\beta$, 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 [9]. As a result, the robot's initial plan, denoted by $\tau_r^0$, can be derived by projecting $R_p^0$ onto $\Pi$, 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 $\pi^j$ is the projection of $q_p^j$ onto $\Pi$, $\forall j = 1, 2, \cdots, S+F$.

## B. Mixed-initiative Controller Design

After the system starts, the robot executes the initial plan $\tau_r^0$ 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 $\varphi_{\text{hard}}$ 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 $\varphi_{\text{hard}}$ 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 $\varphi_{\text{hard}}$ 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 $\infty$.

*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.,

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

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 $d_t \triangleq \mathbf{min}_{\langle \pi, q_1, q_2, c\rangle \in \mathcal{O}_t} \ \|x - \pi\|$ 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 $\mathcal{O}_t$, it holds that $\dot{x}^\intercal u_r = \|u_r(x)\|^2 + \kappa(x)\, u_h(t)^\intercal u_r(x) > 0$, 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 (6). Thus, if the navigation control avoids $\mathcal{O}_t$, the same property is ensured by (6). ■

## C. Discrete Plan Adaptation

In this section, we describe how the discrete plan $\tau_r^t$ 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 $\tau_r^t$ 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; (ii) the labeling function $L(\pi)$ is changed based on the feedback from the robot's sensing module. Given the updated $\mathcal{T}^t$, the mapping function $\delta_p$ of the product automaton $\mathcal{A}_p^t$ is re-evaluated. Consequently, the current plan $\tau_r^t$ might not be optimal anymore regarding the cost in (3). Thus we consider the following problem.

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

For each reachable state $q_p \in Q_t$, we perform a nested Dijkstra's search [9] to find the accepting run that starts from $q_p$ and minimizes the cost 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 the one with the minimum cost, which is the updated run $R_p^{t+}$:

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

Thus the updated plan $\tau_r^t$ is given by the projection of $R_p^{t+}$ onto $\Pi$. 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 $\varphi_{\text{temp}}$ to the robot during run time. Particularly, we consider the following "*pick-up and deliver*" task with deadlines, i.e.,

$$(\varphi^t_{\text{temp}},\, T_{sg}) \triangleq (\Diamond(\pi_s \wedge \Diamond \pi_g),\, T_{sg}), \tag{8}$$

where $\varphi^t_{\text{temp}} \triangleq \Diamond(\pi_s \wedge \Diamond \pi_g)$ is the temporary task assigned at time $t > 0$, meaning that the robot needs to pick up some objects at region $\pi_s$ and deliver them to $\pi_g$ (note that action propositions are omitted here, see [10]), where $\pi_s, \pi_g \in \Pi$; $T_{sg} > 0$ is the *preferred* deadline that task $\varphi^t_{\text{temp}}$ is accomplished. It can be verified that $\varphi^t_{\text{temp}}$ are sc-LTL formulas and can be fulfilled in finite time. Assume that $\varphi^t_{\text{temp}}$ 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 $\varphi^t_{\text{temp}}$.

*Problem 2:* Update $R^t_p$ such that $\varphi^t_{\text{temp}}$ can be fulfilled *without* delay (if possible) and with *minimum* extra cost, while respecting the hard constraints $\varphi_{\text{hard}}$. ■

Assume the remaining optimal run at time $t > 0$ is given by $R^t_p = q_p^{k_0} q_p^{k_0+1} \cdots (q_p^S \cdots q_p^{S+F})^\omega$ and $q_p^{k_s}, q_p^{k_g} \in Q_p$ are the $k_s$-th, $k_g$-th state of $R^t_p$, where $k_s \geq k_g \geq k_0$. Since $R^t_p$ is optimal for the current $\mathcal{T}^t$, we search for the index $k_s$ where the robot can deviate from $R^t_p$ to reach region $\pi_s$ and back, and another index $k_g$ where the robot can deviate from $R^t_p$ to reach $\pi_g$ and back. Denote by $R^{t+}_p$ the updated run after incorporating $\pi_s, \pi_g$. In this way, $\varphi^t_{\text{temp}}$ is satisfied when $\pi_g$ is reached after $\pi_s$ 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 $R^t_p$ in (3) is changed by $\overline{\mathtt{C}}_\beta(R^t_p) \triangleq \mathtt{C}_\beta(R^{t+}_p) - \mathtt{C}_\beta(R^t_p)$. Thus we formulate a 2-stage optimization below: first, we solve

$$\bar{d}_{sg} = \textbf{min}_{\{k_g > k_s \geq 0\}} \, \{\overline{\mathtt{C}}_\beta(R^t_p)\}, \quad \textbf{s.t.} \quad \bar{t}_{sg} \leq 0, \tag{9a}$$

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

$$\bar{d}_{sg} = \textbf{min}_{\{k_g > k_s \geq 0\}} \, \{\bar{t}'_{sg} + \overline{\mathtt{C}}_\beta(R^t_p)\}, \tag{9b}$$

where $\bar{d}_{sg} \geq 0$; $\bar{t}'_{sg} = 0$ if $\bar{t}_{sg} \leq 0$ and $\bar{t}'_{sg} = \bar{t}_{sg}$, otherwise. Note that $\overline{\mathtt{C}}_\beta(R^t_p)$ is $\infty$ if $\varphi_{\text{hard}}$ is violated by $R^{t+}_p$. Since the suffix of $R^t_p$ is repeated infinitely often, the choice of indices $k_s, k_g$ for (9) is finite. Thus (9) can be solved by enumerating these choices.

## D. Human Preference Learning

As discussed in Section IV-A, different choices of $\beta$ may result in different $R^0_p$. The initial plan $R^0_p$ 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 $\beta$ 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: $\zeta_h|_t^{t'} \triangleq \pi'_1 \pi'_2 \cdots \pi'_H$, where $\pi'_h \in \Pi$, $\forall h = 1, 2 \cdots H$ and $H \geq 1$ is the length of $\zeta_h$ that can vary each time the operator acts. Afterwards, the robot continues executing its current plan $\tau^t_r$. 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 $\beta$ as $\beta^\star_h$ such that $\zeta_h|_0^{t'}$ corresponds to the optimal plan under $\beta^\star_h$. ■

The above problem is closely related to the inverse reinforcement learning (IRL) problem [19], [20], where the robot learns about the cost functions of the system model based on demonstration of the preferred plans. As mentioned in [19], most problems of IRL are ill-posed. 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 [20].

First, we compute the set of all finite runs within $\mathcal{A}^{t'}_p$, denoted by $\mathbf{R}^{t'}_h$, that are associated with $\zeta_h|_0^{t'}$. It can be derived iteratively via a breadth-first graph search [5]. Among $\mathbf{R}^{t'}_h$, we find the one with the minimal cost over $\alpha_3$, i.e.,

$$R^\star_h \triangleq \textbf{argmin}_{R \in \mathbf{R}^{t'}_h} \, \boldsymbol{\alpha}_3(R). \tag{10}$$

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

$$\hat{R}^\star_{\beta_k} \triangleq \textbf{argmin}_{R \in \mathbf{R}_{q_1 q_H}} \, \Big(\mathtt{C}_{\beta_k}(R) - M(R, R^\star_h)\Big) \tag{11}$$

where $\mathbf{R}_{q_1 q_H}$ is the set of *all* runs from $q_1$ to $q_H$ in $\mathcal{A}^{t'}_p$; and $M : Q^H \times Q^H \to \mathbb{N}$ is the margin function [20]:

$$M(R,\, R^\star_h) = |\{(q_s, q_t) \in R \,|\, (q_s, q_t) \notin R^\star_h\}|, \tag{12}$$

which returns the number of edges within $R$ that however do not belong to $R^\star_h$. The margin function decreases the total cost $\mathtt{C}_{\beta_k}(R)$ by the difference between $R$ and $R^\star_h$. It can improve generalization and help address the ill-posed nature of Problem 3. To solve (11), we first modify $\mathcal{A}^{t'}_p$ by reducing the $\alpha_1$ cost of each edge $(q_s, q_t) \in R^\star_h$ by one. Then a Dijkstra shortest path search can be performed over the modified $\mathcal{A}_p$ to find the shortest run from $q_1$ to $q_H$ that minimizes the cost with margins in (11). Given $\hat{R}^\star_{\beta_k}$, we can compute the sub-gradient [21] that $\beta_k$ should follow:

$$\nabla \beta_k = \lambda \cdot \beta_k + \big(\boldsymbol{\alpha}_3(R^\star_h) - \boldsymbol{\alpha}_3(\hat{R}^\star_{\beta_k})\big), \tag{13}$$

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

$$\beta_{k+1} = \beta_k - \theta_k \cdot \nabla \beta_k, \tag{14}$$

where $\theta_k > 0$ is the step size or learning rate [22]. Given the updated $\beta_{k+1}$, the same process in (10)-(14) is repeated until the difference $|\beta_{k+1} - \beta_k|$ is less than a predefined threshold $\varepsilon > 0$. At last, the value of $\beta_t$ is updated to $\beta_{k+1}$. Each time the human operator guides the robot to reach a new sequence of regions, the estimation of the value of $\beta_t$ is updated by this algorithm. A detailed analysis of convergence can be found in [20], [21].

**Algorithm 1:** Mixed-initiative Motion and Task Planning

```
   Input: $\mathcal{T}^t$, $\varphi_{\text{hard}}$, $\varphi_{\text{soft}}$, $\beta_0$, $u_h(t)$, $(\varphi^t_{\text{temp}}, T_{sg})$
1  Compute $\mathcal{A}_p^0$ and construct initial plan $\tau_r^0$ under $\beta_0$;
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 $\tau_r^t$ by (7);
6      if $(\varphi^t_{\text{temp}}, T_{sg})$ received then     // Temp. task
7          Update plan $\tau_r^t$ by solving (9);
8      if $\|u_h(t)\| > 0$ then                   // Human input
9          Compute control $u(t)$ by (6);
10         Learn $\beta_l^\star$ by 14 and set $\beta_t^+ = \beta_l^\star$;
11         Update $\tau_r^t$ by (7) given the learned $\beta_t^+$;
12     return $u(t)$, $\tau_r^t$, $\beta_t^+$
```

## V. The Integrated System

In this section, we describe the the real-time execution of the integrated system and the computational complexity.

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

The complete algorithm is shown in Alg. 1. Before the system starts, given the initial model $\mathcal{T}^0$ and the task formulas in (2), the initial plan $\tau_r^0$ is synthesized under the initial $\beta_0$. From $t = 0$, the robot executes $\tau_r^0$ 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 a result, the plan $\tau_r^t$ is updated by (7) accordingly, see Lines 4-5. Moreover, the operator can assign temporary tasks with deadlines as in (8), for which $\tau_r^t$ is modified by solving (9), see Lines 6-7. Lastly, each time the operator guides the robot to follow a new trajectory, the parameter $\beta$ is updated via (14). Then, its current plan $\tau_r^t$ is updated using the updated $\beta$, see Lines 8-12. The above procedure repeats until the system is terminated.

### B. Computational Complexity

The process to synthesize $\tau_r^t$ given $\mathcal{A}_p^t$ (in Line 1) and the plan revision given $\mathcal{T}^t$ (in Line 5) both have complexity $\mathcal{O}(|\mathcal{A}_p^t|^2)$ [9]. The adaptation algorithm for temporary tasks (in Line 7) has complexity $\mathcal{O}(|R_p^t|^2)$. Lastly, the learning algorithm (in Line 11) has complexity $\mathcal{O}(|R_h^\star|^2)$.

## VI. Case Study

We present here numerical studies both in simulation and experiment. All algorithms are implemented in Python 2.7 and available online [18]. 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 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.

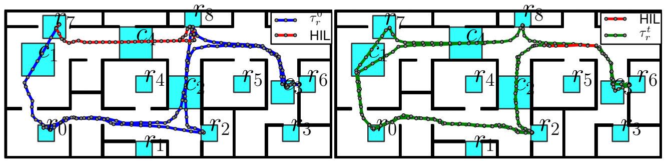

**Fig. 1:** The robot's trajectory in simulation case One, where the robot's initial plan is in blue, the human-guided part is in red, and the updated plan is in green.

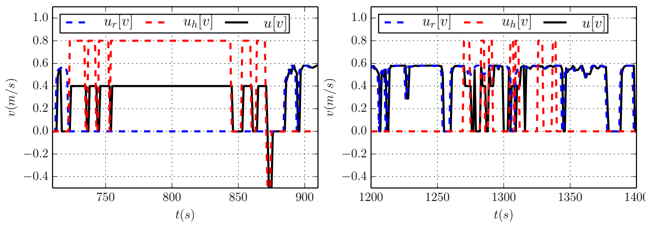

**Fig. 2:** The mixed control for linear velocity during two time periods when the human control (in red) is active.

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 $\varphi_{\text{temp}}$ are specified via ROS messages. More details can be found in the software implementation [18] and simulation video [23].

*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 product automaton, which has 312 states and 1716 transitions. The parameter $\beta$ 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. 1. 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, $\beta$ is updated by (14) and the final value is 16.35 after 20 iterations with $\varepsilon = 0.2$. 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. 1. Moreover, the human operator drives the robot towards $r_5$ during $[1250s, 1350s]$, which is not allowed by $\varphi_{\text{hard}}$. 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 are shown in Fig. 2.

*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

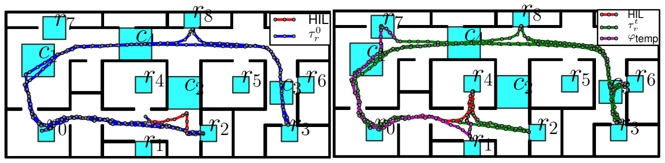

**Fig. 3:** The robot's trajectory in simulation case Two. The robot's trajectory while performing the temporary task is in magenta.

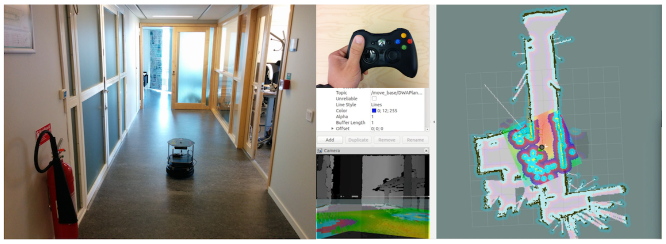

**Fig. 4:** The human-in-the-loop experiment setup, where the robot is controlled by both its autonomous controller and the human inputs.

performance is $\varphi_{2,\text{soft}} = \Box\Diamond\big(r_4 \rightarrow (\neg r_5 \mathsf{U} \Diamond r_6)\big)$, 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. Initially, $\beta = 0$ meaning that $\varphi_{2,\text{soft}}$ 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 updated plan allows the robot to reach $r_8$ from $r_2$ via $c_1$, as shown in Fig. 3. 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, $\beta$ is increased by (14) to 11.3 after 12 iterations with $\varepsilon = 0.1$. 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 $2700s$. This temporary task is incorporated and fulfilled at $2400s$, as shown in Fig. 3.

## B. Experiment

The experiment setup involves a TurtleBot within the office environment at Automatic Control Lab, KTH. Details are omitted here, see [18] and [23].

*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. 4. The robot's 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 $\beta$ is initially set to 0, the robot only surveils $r_0$ and $r_1$ for the hard task. 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 $\beta_t$ is updated by (14) given the robot's past trajectory. The final convergence value is 1.58 with $\varepsilon = 0.01$ after 15 iterations. Then updated plan 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

We present a human-in-the-loop task and motion planning strategy for mobile robots with mixed-initiative control. The proposed 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] 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.
[9] 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.
[10] ——, "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.
[11] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.
[18] Software, `http://github.com/MengGuo/mix_initiative`.
[19] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.
[20] 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.
[21] N. Z. Shor, *Minimization methods for non-differentiable functions*. Springer Science & Business Media, 2012, vol. 3.
[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
[23] Video, `https://vimeo.com/230487800,232727691`.