Advisory Temporal Logic Inference and Controller Design for Semiautonomous Robots

Zhe Xu, *Student Member, IEEE*, Sayan Saha, *Student Member, IEEE*, Botao Hu, *Student Member, IEEE*, Sandipan Mishra, *Member, IEEE*, and A. Agung Julius, *Member, IEEE*

***Abstract*— In this paper, we present a method to learn (infer) and refine a set of advices from the trajectories generated in the successful and failed attempts in a task or game, in the form of advisory signal temporal logic (STL) formulas. Each advice consists of an advisory motion STL formula that characterizes the spatial–temporal pattern of the motion as a feature of success and an advisory selection STL formula as a criterion for the environment to select the advice. For the inference of advisory STL formulas, we provide a theoretical framework of perfect classification with a labeled set of trajectories with different time lengths. We design an advisory controller that can drive the robots to satisfy an advisory motion STL formula based on the advice selected according to the advisory selection STL formula. The advisory controller can advise or guide the human operators or the robots for better performance with the shared autonomy between the human operator and the controller. We provide two case studies to test the effectiveness of the advisory controller, one with a Baxter-On-Wheels simulator and the other with two quadrotors in an experimental testbed in iteratively improving the success rates of completing the tasks with the help of the designed advisory controller.**

***Note to Practitioners*—The method described in this paper can be used to obtain knowledge or patterns of the environment based on the trajectories generated by the human operators/demonstrators and utilize the obtained knowledge or patterns for improving the performance of the future operators. The obtained information is the logical statements about the waypoints or subgoals to be reached, and obstacles or dangerous regions to be avoided during certain time intervals, when the environment satisfies certain conditions also expressed in the form of logical statements. The methodology of inferring knowledge from data and designing advisory controllers for guiding or helping future practices is potentially useful in many applications where the dynamic mathematical model of the external environment is unknown.**

***Index Terms*—Advisory controller, advisory signal temporal logic (STL), shared autonomy.**

I. Introduction

WITH the increasing technological advances in robotic autonomy and the improved interaction platforms between humans and the environments, the paradigm of human operators operating the robots is gradually shifting to the new paradigm of robotic autonomy or human–robot collaboration [1]–[3]. In uncertain or adversarial environment, autonomous or semiautonomous robots with all the necessary sensors may still fail to complete tasks within specified time when wrong strategies are employed. To improve the performance of the robots in these situations, we can utilize the trajectories of the robots generated in successful or failed attempts to provide valuable information or knowledge for advising or guiding the future operations. For example, as shown in Fig. 1, the robot aims to reach the goal region within 10 s while avoiding a moving obstacle with the initial position randomly generated in the entire space. The robot has sensors for the position of the moving obstacle, but the dynamic mathematical model of the moving obstacle is unknown. In this case, we can record the trajectories in a few attempts and infer green regions 1 and 2 as waypoints to be reached during certain time intervals, and blue regions 1 and 2 as criterion regions for the moving obstacle to select the waypoint. With such inferred knowledge, the robot can increase the success rate of completing the task by first going to the inferred green region 1 when the moving obstacle is in the inferred blue region 1 at the starting time, and first going to the inferred green region 2 when the moving obstacle is in the inferred blue region 2 at the starting time while avoiding the moving obstacle during the whole time.

One challenge in the learning-based approaches [4], [5] in robotic autonomy or human–robot collaborations is that the knowledge learned in the training process should be more comprehensible to the human operators to enable better monitoring and collaboration. Temporal logics can express complex high-level time-related control specifications such as *"Go to*

Manuscript received March 13, 2018; accepted May 9, 2018. This paper was recommended for publication by Associate Editor H. Liu and Editor K. Saitou upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant CNS-1218109, Grant CNS-1550029, and Grant CNS-1618369. *(Corresponding author: Zhe Xu.)*

Z. Xu, S. Saha, and A. A. Julius are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: xuz8@rpi.edu; sahas3@rpi.edu; juliua2@rpi.edu).

B. Hu and S. Mishra are with the Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: hub2@rpi.edu; mishrs2@rpi.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. This supplementary video contains some test simulations with and without the designed advisory controller in the two case studies in Sections V and VI of the paper "Advisory Temporal Logic Inference and Controller Design for Semi-autonomous Robots." The total size of the file is 16MB.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2836867

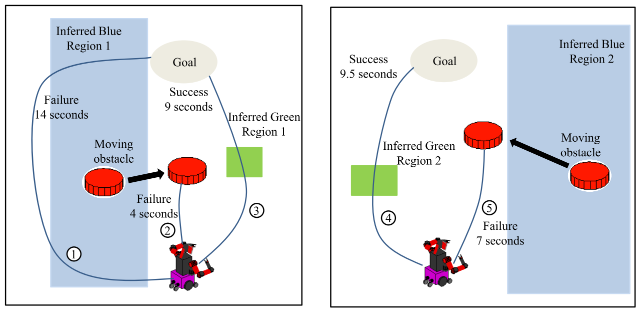


Fig. 1. With a certain number of successful and failed attempts (five of them drawn in the picture), the robot can increase the success rate of completing the task of reaching the goal safely within the specified time (10 s), if the robot can learn the knowledge of first going to the inferred green region 1 when the moving obstacle is in the inferred blue region 1 at the starting time, and first going to the inferred green region 2 when the moving obstacle is in the inferred blue region 2 at the starting time while avoiding the moving obstacle during the whole time.

the green region and the goal region in different time intervals while avoiding the red region in certain time intervals." These temporal logic formulas can capture the feature of success in a form that resembles natural languages [6] and thus improve the comprehensibility of the inferred knowledge.

A. Related Works

Recently, there has been a growing interest in inferring dense-time temporal logic formulas from system trajectories [6]–[13]. Algorithms for this purpose are enabled by the robust semantics of dense-time temporal logics. For example, Fainekos and Pappas [14] introduced a robust semantics for metric temporal logic based on how far a given trajectory stands, in space and time, from satisfying or violating a temporal logic property. With signal temporal logic (STL), Donzé and Maler [15] also defined several variants of robustness measures (more details are in Section II). Kong *et al.* [6] showed that the formulas admit a partial order of complexity, based on which the structure can be chosen and the STL formula that best classifies the trajectories in a desired set and an undesired set can be automatically inferred. Bombara *et al.* [7] proposed a decision tree approach to infer STL formulas for classification that can decrease both the misclassification rate and the computational cost. However, the trajectories to be classified all have the same time length in these papers, while this condition may not hold in actual applications when some trajectories end earlier than others. Besides, very few researches [11] utilize the temporal logic inference approaches in improving the performances of robotic autonomy or human–robot collaboration, and none of the existing works utilize the inferred temporal logic formulas as features of the desired set for iteratively improving the performance of the future generated trajectories.

B. Contributions and Advantages

1) We Provide a Method to Iteratively Learn (Infer) and Refine a Set of Advices from the Trajectories Generated in the Successful and Failed Attempts in a Task, with Each Advice in the Form of an Advisory Motion STL Formula and an Advisory Selection STL Formula: We use the decision tree approach to infer a set of advices, with each inferred advice consisting of an advisory motion STL formula that characterizes the spatial–temporal pattern of the motion of the robot as a feature of success and an advisory selection STL formula as the criterion for the environment to select the advice. The inferred set of advices need to satisfy the coverage, soundness, and uniqueness properties (see Problem 1 in Section IV-A). We have developed a refinement procedure to refine the set of advices by conditionally modifying either the advisory motion STL subformula or the advisory selection STL subformula at each node of the decision tree when newly generated trajectories are added.

2) We Provide a Theoretical Framework of Perfect Classification with a Labeled Set of Trajectories with Different Time Lengths: We present some sufficient and necessary conditions for perfect classification (i.e., with zero misclassification rate) of observed trajectories with different lengths using STL formulas. We treat each observed trajectory as a prefix and it can be inversely projected to the set of all possible virtual completed trajectories with the same prefix. In the literature of temporal logic model checking and monitoring [16]–[18], there are different views of temporal logic satisfaction or violation for trajectories of different lengths, such as the strong and the weak views. We present the Boolean semantics of STL formulas for prefixes in the strong and weak views and utilize them for providing a necessary condition for the parameters of the STL formulas for perfect classification.

3) We Design an Advisory Controller that Can Advise or Guide the Human Operators or the Robots for Better Performance with the Shared Autonomy Between the Human Operator and the Controller: We design an advisory controller based on the inferred set of advices [19]–[21]. The human operator can decide whether to switch ON or switch OFF the advisory controller at any time, and whenever the advisory controller is turned ON, the controller generates a trajectory that satisfies the advisory motion STL formula based on the advice selected according to the advisory selection STL formula.

Our approach is different from the shared control of the human and semiautonomous robots in [1]–[3] as both the motion specifications and the selection of motions are learned from the trajectory data generated from human demonstrators. Our proposed approach is applicable in the scenarios where the followings are both true: 1) there could be known information about the environment (e.g., the robots may have sensors for the positions of the moving obstacles), but the dynamic mathematical model of the environment is unknown and 2) the dynamic mathematical model of the environment remains the same, and knowledge or patterns can be learned from the attempts or experiences of the human demonstrators or operators. We designed a game with a Baxter-On-Wheels (BOW) simulator and a reach-avoid game with two quadrotors in an experimental testbed to test the effectiveness of the advisory controller in iteratively improving the success rates.

The rest of this paper is structured as follows. Section II reviews the framework of STL and the corresponding notations. Section III describes the theory of temporal logic inference for classification with trajectories of different lengths. Section IV outlines the algorithms of the advisory STL formula inference, advisory controller synthesis for shared autonomy, and advisory STL formula refinement. Section V describes the implementation of the algorithms on a game that we designed on a BOW simulator and Section VI describes the implementation of the algorithms on a reach-avoid game with two quadrotors in an experimental testbed. Finally, some conclusions are presented in Section VII.

II. Signal Temporal Logic

In this section, we briefly review the STL [15]. We start with the Boolean semantics of STL. The state of the system that we are studying is described by a set of n variables that can be written as a vector $x = [x_1, x_2, \ldots, x_n]^T$. The domain of x is denoted by $\mathbb{X} = \mathbb{X}_1 \times \mathbb{X}_2 \times \cdots \times \mathbb{X}_n$ ($\mathbb{X}_i$ is a subset of $\mathbb{R}$). The domain $\mathbb{B} = \{\text{True}, \text{False}\}$ is the Boolean domain and the time set is $\mathbb{T} = \mathbb{R}_{\geqslant 0}$. With a slight abuse of notation, we define trajectory (or signal or behavior) x describing an evolution of the system as a function from $\mathbb{T}$ to $\mathbb{X}$. Therefore, x_i refers to both the name of the ith state variable and its valuation in $\mathbb{X}$. A set $\Pi = \{\pi_1, \pi_2, \ldots \pi_n\}$ is a set of atomic predicates, each of which can be either true or false. The atomic predicates can express properties such as "*the robot base is inside the goal region*," or "*the robot end-effector is above 5 m*." The syntax of the (F, G)-fragment of STL is defined recursively as follows[1]:

$$\phi = \top \mid \pi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid F_I\phi \mid G_I\phi$$

where $\top$ stands for the Boolean constant (True), π is an atomic predicate in the form of an inequality $f(x(t)) > 0$ where f is some real-valued function, $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) are the standard Boolean connectives, "F" and "G" are the temporal operators representing "eventually" and "always," and I is an interval of the form $I = [i_1, i_2)$ ($i_1 < i_2, i_1, i_2 \in \mathbb{T}$).

For an STL formula ϕ, the necessary length $\|\phi\|$ is defined recursively as follows [23]:

$$\begin{aligned} \|\pi\| &= 0, \quad \|\neg\phi\| = \|\phi\| \\ \|\phi_1 \wedge \phi_2\| &= \max(\|\phi_1\|, \|\phi_2\|) \\ \left\|F_{[t_1,t_2)}\phi\right\| &= \left\|G_{[t_1,t_2)}\phi\right\| = \|\phi\| + t_2. \end{aligned}$$

For example, for the STL formula $\phi = F_{[0,20)}(x < 3) \wedge F_{[30,100)}(G_{[0,10)}(x > 20))$, $\|\phi\| = \max(20, 100 + 10) = 110$.

In the following definitions of STL in this section, the trajectories are assumed to have the same length $H \geq \|\phi\|$.

Definition 1: We use (x, t) to represent the trajectory x at time t. $(x, t) \models \phi$ means the trajectory x satisfies ϕ at time t, and the Boolean semantics of the (F, G)-fragment of STL are defined recursively as follows:

$$\begin{aligned} &(x,t) \models \pi \quad \text{iff } f(x(t)) > 0 \\ &(x,t) \models \neg\phi \quad \text{iff } (x,t) \not\models \phi \\ &(x,t) \models \phi_1 \wedge \phi_2 \quad \text{iff } (x,t) \models \phi_1 \quad \text{and } (x,t) \models \phi_2 \\ &(x,t) \models F_{[t_1,t_2)}\phi \quad \text{iff } \exists t' \in [t+t_1, t+t_2) \\ &\qquad\qquad\qquad\qquad \text{s.t. } (x,t') \models \phi \\ &(x,t) \models G_{[t_1,t_2)}\phi \quad \text{iff } (x,t') \models \phi \\ &\qquad\qquad\qquad\qquad \forall t' \in [t+t_1, t+t_2). \end{aligned}$$

[1]Although other temporal operators such as "Until " ($\mathcal{U}$) and "Since " ($\mathcal{S}$) may also appear in the full syntax of STL, they are omitted from the syntax here (following [6]) as they can be hard to interpret and are not often used for the inference of STL formulas. Similar terms can be found in [22] for the (F, G)-fragment of linear temporal logic.

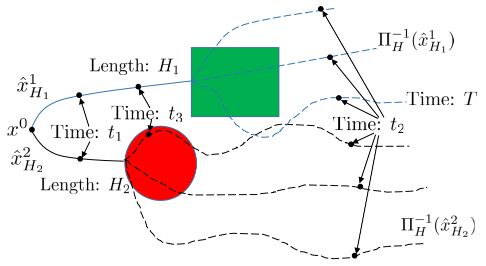


Fig. 2. Two sets of virtual completed trajectories $\Pi_H^{-1}(\hat{x}^1_{H_1})$ and $\Pi_H^{-1}(\hat{x}^2_{H_2})$ with the observed trajectories (prefixes) $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$ (solid lines), respectively (dotted lines: virtual parts of the infinite virtual completed trajectories).

The robustness degree of a trajectory x with respect to an STL formula ϕ at time t is given as $r(x, \phi, t)$, where r can be calculated recursively via the quantitative semantics [15]

$$\begin{aligned} r(x,\pi,t) &= f(x(t)) \\ r(x,\neg\phi,t) &= -r(x,\phi,t) \\ r(x,\phi_1 \wedge \phi_2, t) &= \min(r(x,\phi_1,t), r(x,\phi_2,t)) \\ r(x, F_{[\tau_1,\tau_2)}\phi, t) &= \max_{t+\tau_1 \leq t' < t+\tau_2} r(x,\phi,t') \\ r(x, G_{[\tau_1,\tau_2)}\phi, t) &= \min_{t+\tau_1 \leq t' < t+\tau_2} r(x,\phi,t'). \end{aligned}$$

In Section III, we will deal with trajectories that can have different lengths possibly shorter than $\|\phi\|$ to evaluate the truth value of ϕ.

III. Temporal Logic Inference for Classification With Trajectories of Different Time Lengths

We consider trajectories generated from a moving agent that is completing a certain task. As shown in the example in Fig. 2, for a certain task to be finished (arriving at the green region while avoiding the red region) within the specified time T, the task can be finished ahead of the specified time (the blue trajectory, with length $H_1 < T$), the task can also be failed before the specified time (the black trajectory, with length $H_2 < T$) if a safety specification is violated. A trajectory with length less than the specified time length means that the observed trajectory is sufficient to determine the result of this attempt, i.e., the result will not be affected by anything that may happen after this time. Therefore, we treat each observed trajectory as a prefix and it can be inversely projected to the set of all possible virtual completed trajectories with the same prefix. We denote all the functions (trajectories) mapping from $\mathbb{T}$ to $\mathbb{X}$ as $\mathbb{X}^{\mathbb{T}}$ and denote all the functions (trajectories) mapping from $\mathbb{T}_H$ to $\mathbb{X}$ as $\mathbb{X}^{\mathbb{T}_H}$, where $\mathbb{T}_H \triangleq [0, H]$.

Definition 2: The prefix projection is a mapping $\Pi_H : \mathbb{X}^{\mathbb{T}} \to \mathbb{X}^{\mathbb{T}_H}$, and $\hat{x}_H$ is defined as the prefix (of length H) of a trajectory x, denoted as $\hat{x}_H = \Pi_H(x)$, if

$$\hat{x}_H(t) = x(t) \quad \forall t \in [0, H]. \tag{1}$$

From Definition 2, we denote the set of all possible virtual completed trajectories with the same prefix $\hat{x}_H$ as $\Pi_H^{-1}(\hat{x}_H)$, i.e., $\Pi_H^{-1}(\hat{x}_H) = \{x | \Pi_H(x) = \hat{x}_H\}$.

Next, we introduce the Boolean semantics of an STL prefix in the strong and the weak views, which are modified from the literature of temporal logic model checking and monitoring [16]–[18]. In the following, $(\hat{x}_H, t) \models_S \phi$ (resp. $(\hat{x}_H, t) \models_W \phi$) means the prefix $\hat{x}_H$ strongly (resp. weakly) satisfies ϕ at time t, $(\hat{x}_H, t) \not\models_S \phi$ (resp. $(\hat{x}_H, t) \not\models_W \phi$) means the prefix $\hat{x}_H$ fails to strongly (resp. weakly) satisfy ϕ at time t.

Definition 3: The Boolean semantics of the (F, G)-fragment STL for prefixes in the strong view is defined recursively as follows:

$$\begin{aligned}
&(\hat{x}_H, t) \models_S \pi \quad \text{iff } t \le H \text{ and } f(\hat{x}_H(t)) > 0\\
&(\hat{x}_H, t) \models_S \neg\phi \quad \text{iff } (\hat{x}_H, t) \not\models_W \phi\\
&(\hat{x}_H, t) \models_S \phi_1 \wedge \phi_2 \quad \text{iff } (\hat{x}_H, t) \models_S \phi_1 \text{ and } (\hat{x}_H, t) \models_S \phi_2\\
&(\hat{x}_H, t) \models_S \phi_1 \vee \phi_2 \quad \text{iff } (\hat{x}_H, t) \models_S \phi_1 \text{ or } (\hat{x}_H, t) \models_S \phi_2\\
&(\hat{x}_H, t) \models_S F_{[t_1,t_2)}\phi \quad \text{iff } \exists t' \in [t+t_1, t+t_2)\\
&\qquad\qquad\qquad\qquad \text{s.t. } (\hat{x}_H, t') \models_S \phi\\
&(\hat{x}_H, t) \models_S G_{[t_1,t_2)}\phi \quad \text{iff } (\hat{x}_H, t') \models_S \phi\\
&\qquad\qquad\qquad\qquad \forall t' \in [t+t_1, t+t_2).
\end{aligned}$$

Definition 4: The Boolean semantics of the (F, G)-fragment STL for prefixes in the weak view is defined recursively as follows:

$$\begin{aligned}
&(\hat{x}_H, t) \models_W \pi \quad \text{iff either of the following holds:}\\
&\qquad 1)\ t \le H \ \text{ and } f(\hat{x}_H(t)) > 0\\
&\qquad 2)\ t > H\\
&(\hat{x}_H, t) \models_W \neg\phi \quad \text{iff } (\hat{x}_H, t) \not\models_S \phi\\
&(\hat{x}_H, t) \models_W \phi_1 \wedge \phi_2 \quad \text{iff } (\hat{x}_H, t) \models_W \phi_1 \text{ and } (\hat{x}_H, t) \models_W \phi_2\\
&(\hat{x}_H, t) \models_W \phi_1 \vee \phi_2 \quad \text{iff } (\hat{x}_H, t) \models_W \phi_1 \text{ or } (\hat{x}_H, t) \models_W \phi_2\\
&(\hat{x}_H, t) \models_W F_{[t_1,t_2)}\phi \quad \text{iff } \exists t' \in [t+t_1, t+t_2)\\
&\qquad \text{s.t. } (\hat{x}_H, t') \models_W \phi\\
&(\hat{x}_H, t) \models_W G_{[t_1,t_2)}\phi \quad \text{iff } (\hat{x}_H, t') \models_W \phi\\
&\qquad \forall t' \in [t+t_1, t+t_2).
\end{aligned}$$

From Definitions 3 and 4, it can be seen that a strong satisfaction or violation of a prefix with respect to an (F, G)-fragment STL formula implies a weak satisfaction or violation of the prefix with respect to the same STL formula, as shown in Fig. 3.

Definition 5: Given a labeled set of prefixes $\mathcal{D} = \{(\hat{x}^i_{H_i}, l_i)\}_{i=1}^{N_D}$, $l_i = 1$ represents desired behavior and $l_i = -1$ represents undesired behavior, an STL formula ϕ evaluated at time t (usually $t = 0$) perfectly classifies the desired behaviors

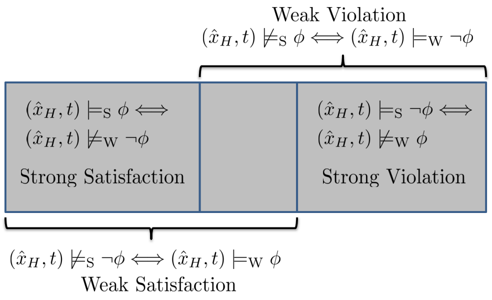

Weak Violation
$(\hat{x}_H, t) \not\models_S \phi \Longleftrightarrow (\hat{x}_H, t) \models_W \neg\phi$
$(\hat{x}_H, t) \models_S \phi \Longleftrightarrow$
$(\hat{x}_H, t) \not\models_W \neg\phi$
Strong Satisfaction
$(\hat{x}_H, t) \models_S \neg\phi \Longleftrightarrow$
$(\hat{x}_H, t) \not\models_W \phi$
Strong Violation
$(\hat{x}_H, t) \not\models_S \neg\phi \Longleftrightarrow (\hat{x}_H, t) \models_W \phi$
Weak Satisfaction

Fig. 3. Venn diagram of strong (weak) satisfaction and strong (weak) violation.

and undesired behaviors if the following condition is satisfied: $(\hat{x}^i_{H_i}, t) \models_S \phi$, if $l_i = 1$; $(\hat{x}^i_{H_i}, t) \models_S \neg\phi$, if $l_i = -1$.

While in general either the strong or the weak views can be taken for $l_i = 1$ and $l_i = -1$ for perfect classification, we choose the strong view in Definition 5 because in the settings of this paper, we are assuming that all the observed trajectories are sufficient to determine the truth value of the STL formulas (these observed trajectories are referred to as informative prefixes in [18]).

We first derive a sufficient condition for perfect classification of prefixes using STL formulas. In this paper, we let $\|\cdot\|$ denote the two-norm and $C([a, b])$ denote the space of continuous, real-valued functions on $[a, b]$.

Theorem 1: For two prefixes, $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$, defined in $C([0, H_1])$ and $C([0, H_2])$ respectively, there exists at least one STL formula that can perfectly classify $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$ if and only if $\sup_{0\le t\le \min(H_1,H_2)} \|\hat{x}^1_{H_1}(t) - \hat{x}^2_{H_2}(t)\| > 0$.

Proof: See the Appendix. ■

Remark 1: If we modify the form of interval I as $I = [i_1, i_2]$ ($i_1 \le i_2, i_1, i_2 \in \mathbb{T}$), then even if the two prefixes are not continuous in their time domains, we can still always find the STL formula $\phi = G_{[t^*,t^*]}(x_i > \hat{x}^1_{H_1,i}(t^*) - \epsilon)$ or $\phi' = F_{[t^*,t^*]}(x_i \ge \hat{x}^2_{H_2,i}(t^*) + \epsilon)$ (see the proof of Theorem 1) that can perfectly classify $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$, as both $G_{[t^*,t^*]}$ and $F_{[t^*,t^*]}$ represent the time point t^*.

Theorem 2: Given two sets of prefixes $\mathcal{A} = \{\hat{x}^1_{H_1}, \ldots, \hat{x}^{N_A}_{H_{N_A}}\}$ and $\mathcal{B} = \{\hat{x}'^1_{H'_1}, \ldots, \hat{x}'^{N_B}_{H'_{N_B}}\}$, there exists at least one STL formula that can perfectly classify the two sets of prefixes if and only if for any prefix $\hat{x}^i_{H_i} \in \mathcal{A}$ and any prefix $\hat{x}'^j_{H'_j} \in \mathcal{B}$, $\sup_{0\le t\le \min(H_i,H'_j)} \|\hat{x}^i_{H_i}(t) - \hat{x}'^j_{H'_j}(t)\| > 0$.

Proof: See the Appendix. ■

Remark 2: While Theorem 2 provides a guarantee that STL formula in the form of $\phi = (\phi_{11} \wedge \phi_{12} \wedge \cdots \wedge \phi_{1N_B}) \vee (\phi_{21} \wedge \phi_{22} \wedge \cdots \wedge \phi_{2N_B}) \vee \ldots (\phi_{N_A1} \wedge \phi_{N_A2} \wedge \cdots \wedge \phi_{N_AN_B})$ (see the Proof of Theorem 2) can perfectly classify the two sets of prefixes when the conditions of Theorem 2 are met, we do not necessarily infer STL formula in this form. For trajectories of the same length, the methods in [6]–[12] actually infer an STL formula from a set of templates and if there is no STL formula in the set of templates that can perfectly classify the two sets of trajectories, then it infers an STL formula from the set of templates that minimizes a performance metric such as the misclassification rate.

Next, we derive a necessary condition for the perfect classification of prefixes using STL formulas. As in the example in Fig. 2, assume that $t_1 < H_2 < t_3 < H_1 < t_2 < T$, the STL formula $\phi_1 = F_{[t_1,t_2)}$ *green region* is not possible to get perfect classification as the prefix $\hat{x}^1_{H_1}$ (successful attempt) arrives at the green region between time t_1 and t_2 (strongly satisfying ϕ_1), but the prefix $\hat{x}^2_{H_2}$ (failed attempt) never enters the green region before it ends at time $H_2 < t_2$, so ϕ_1 is neither strongly satisfied nor strongly violated by $\hat{x}^2_{H_2}$. On the other hand, the STL formula $\phi_2 = G_{[t_1,t_3)}\neg$*red region* is possible to get perfect classification as $\hat{x}^1_{H_1}$ does not enter the red region between time t_1 and t_3 (strongly satisfying ϕ_2), and $\hat{x}^2_{H_2}$ has already violated ϕ_2 when it ends at time $H_2 < t_3$, as the violation of $\neg$*red region* at any time between time t_1 and t_3 will lead to the strong violation of ϕ_2. Intuitively, there are certain constraints for the temporal parameters of an STL formula, if the STL formula is possible to get perfect classification, as presented in Proposition 1.

Proposition 1: If there exists an (F, G)-fragment STL formula ϕ evaluated at time t that can perfectly classify the desired behaviors and undesired behaviors in $\mathcal{D} = \{(\hat{x}^i_{H_i}, l_i)\}_{i=1}^{N_D}$, then $H_i \geq t + \tau(\phi, l_i)$, where $\tau(\phi, l_i)$ is defined recursively as follows:

$$\begin{aligned}
\tau(\pi, l_i) &= 0\\
\tau(\neg\phi, l_i) &= \tau(\phi, -l_i)\\
\tau(\phi_1 \wedge \phi_2, l_i) &= \begin{cases}\max\{\tau(\phi_1, l_i), \tau(\phi_2, l_i)\}, & \text{if } l_i = 1\\ \min\{\tau(\phi_1, l_i), \tau(\phi_2, l_i)\}, & \text{if } l_i = -1\end{cases}\\
\tau(F_{[t_1,t_2)}\phi, l_i) &= \begin{cases}\tau(\phi, l_i) + t_1, & \text{if } l_i = 1\\ \tau(\phi, l_i) + t_2, & \text{if } l_i = -1\end{cases}\\
\tau(G_{[t_1,t_2)}\phi, l_i) &= \begin{cases}\tau(\phi, l_i) + t_2, & \text{if } l_i = 1\\ \tau(\phi, l_i) + t_1, & \text{if } l_i = -1.\end{cases}
\end{aligned}$$

Proof: See the Appendix. ■

We denote the robustness degree of a prefix $\hat{x}^i_{H_i}$ with respect to an STL formula ϕ and the label l_i as $\hat{r}(\hat{x}^i_{H_i}, \phi, l_i, t)$. According to Proposition 1, if $H_i \geq t + \tau(\phi, l_i)$, then $\hat{r}(\hat{x}^i_{H_i}, \phi, l_i, t)$ can be calculated recursively as follows:

$$\begin{aligned}
\hat{r}(\hat{x}^i_{H_i}, \mu, l_i, t) &= f(\hat{x}^i_{H_i}(t))\\
\hat{r}(\hat{x}^i_{H_i}, \neg\phi, l_i, t) &= -\hat{r}\big(\hat{x}^i_{H_i}, \phi, -l_i, t\big)\\
\hat{r}(\hat{x}^i_{H_i}, \phi_1 \wedge \phi_2, l_i, t) &= \min\big(\hat{r}\big(\hat{x}^i_{H_i}, \phi_1, l_i, t\big),\\
&\qquad \hat{r}\big(\hat{x}^i_{H_i}, \phi_2, l_i, t\big)\big)\\
\hat{r}(\hat{x}^i_{H_i}, F_{[t_1,t_2)}\phi, l_i, t) &= \max_{\substack{t+t_1 \leq t'\\ < \min(t+t_2, H_i)}} \hat{r}\big(\hat{x}^i_{H_i}, \phi, l_i, t'\big)\\
\hat{r}(\hat{x}^i_{H_i}, G_{[t_1,t_2)}\phi, l_i, t) &= \min_{\substack{t+t_1 \leq t'\\ < \min(t+t_2, H_i)}} \hat{r}\big(\hat{x}^i_{H_i}, \phi, l_i, t'\big). \quad (2)
\end{aligned}$$

It can be seen that when $l_i = 1$, $\hat{r}(\hat{x}^i_{H_i}, \phi, l_i, t)$ is the robustness margin for strong satisfaction (positive means strong satisfaction and negative means weak violation); when $l_i = -1$, $\hat{r}(\hat{x}^i_{H_i}, \phi, l_i, t)$ is the negative robustness margin for strong violation (positive means weak satisfaction and negative means strong violation).

IV. Advisory STL Formula Inference, Controller Synthesis, and Refinement

A. Advisory STL Formula Inference

We first present the problem formulation and the algorithms for the inference of advisory STL formulas.

Definition 6: We define an advice as a pair $\gamma = (\phi, \psi)$, where ϕ, ψ are both (F, G)-fragment STL formulas, ϕ is called the advisory motion STL formula and ψ is called the advisory selection STL formula.

In the following, we use $\hat{x}_H$ to denote the observed trajectory (prefix) of the robot and use $\hat{y}_H$ to denote the observed trajectory (prefix) of the environment.

Problem 1: Assume $S = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i)\}_{i=1}^{N_S}$ is a training data set of labeled attempts ($l_i = 1$ represents successful attempts and $l_i = -1$ represents failed attempts), we seek to find a set of advices $\Gamma = \{\gamma_j\}_{j=1}^{M}$ ($M \leq N_S$) such that the followings are true:

1) *Coverage:* For every i, if $l_i = 1$, then there exists an advice $\gamma_j = (\phi_j, \psi_j)$ such that $((\hat{x}^i_{H_i}, 0) \models_S \phi_j) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_j)$;
2) *Soundness:* For each advice $\gamma_j = (\phi_j, \psi_j)$ and every i, if $((\hat{x}^i_{H_i}, 0) \models_S \phi_j) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_j)$, then $l_i = 1$;
3) *Uniqueness:* For every i such that $l_i = 1$, if there exists j such that $(\hat{y}^i_{H_i}, 0) \models_S \psi_j$, then for any $k \neq j$, $(\hat{y}^i_{H_i}, 0) \not\models_S \psi_k$.

In Problem 1, 1) guarantees that the set of advices cover all the successful attempts in the training data set; 2) guarantees that when the environment trajectory (prefix) strongly satisfies the advisory selection STL formula and the robot moves in such a way that the generated trajectory (prefix) strongly satisfies the advisory motion STL formula, then it leads to success in the training data set; and 3) guarantees that at most one advice can be selected for each attempt with label 1. It can be seen that a possible trivial solution of Problem 1 is to find $\Gamma = \{\gamma_j\}_{j=1}^{N_S}$, where each ϕ_j perfectly classifies $\hat{x}^j_{H_j}$ and all the other prefixes $\hat{x}^i_{H_i}$ $(i \neq j)$, and each ψ_j perfectly classifies $\hat{y}^j_{H_j}$ and all the other prefixes $\hat{y}^i_{H_i}$ $(i \neq j)$. This trivial solution easily leads to overfitting, and in the following, we present a framework to solve Problem 1 using the decision tree approach that is inspired by [7].

Our approach to infer the set of advices is performed in two stages. In the first stage, we construct a decision tree where each node of a tree is associated with a pair of an advisory motion STL subformula and an advisory selection STL subformula (to avoid confusion, we refer to the advisory STL formula at each node of the decision tree as an advisory STL subformula). In the second stage, the constructed decision tree is transformed to a set of advices, where each advisory motion (selection) logic formula in each advice is constructed from the advisory motion (selection) logic subformulas or the negation of them connected with conjunction operators.

In this paper, we choose to represent the predicates as polyhedral sets as they are more general than rectangular sets and computationally easier to handle than other more complex sets (ellipsoidal sets, nonconvex sets, and so on). Each polytopic predicate ρ is expressed as a conjunction of linear inequalities

$$\rho = \left(\bigwedge_{k=1}^{m} a_k^T x < b_k \right), \quad a_k \in \mathbb{R}^n, \; b_k \in \mathbb{R} \tag{3}$$

where the vector a_k and the number b_k are parameters that define the polytopic predicate, and m is the number of linear inequalities in the polytopic predicate (each linear inequality $a_k^T x < b_k$ is an atomic predicate π_k, where $f_k(x) = -a_k^T x + b_k$ is a real-valued function). We constraint $\|a_k\|_2 = 1$ to reduce redundancy and expedite the searching process.

Definition 7: For an (F, G)-fragment STL formula ϕ, we define the start-effect time $t_s(\phi)$ and end-effect time $t_e(\phi)$ as follows:

$$\begin{aligned} t_s(\pi) &= t_e(\pi) = 0 \\ t_s(\neg\phi) &= t_s(\phi), \quad t_e(\neg\phi) = t_e(\phi) \\ t_s(\phi_1 \wedge \phi_2) &= \min\{t_s(\phi_1), t_s(\phi_2)\} \\ t_e(\phi_1 \wedge \phi_2) &= \max\{t_e(\phi_1), t_e(\phi_2)\} \\ t_s(F_{[t_1,t_2)}\phi) &= t_s(\phi) + t_1, \quad t_e(F_{[t_1,t_2)}\phi) = t_e(\phi) + t_2 \\ t_s(G_{[t_1,t_2)}\phi) &= t_s(\phi) + t_1, \quad t_e(G_{[t_1,t_2)}\phi) = t_e(\phi) + t_2. \end{aligned} \tag{4}$$

Definition 8: We define two types of primitive STL subformulas.

1) *Safety Primitive STL Subformula:* Primitive STL subformula that provides specification on the safety of the agent in avoiding obstacles or other safety hazards.
2) *Liveness Primitive STL Subformula:* Primitive STL subformula that provides specification on the performance of the task to be finished within the specified time.

The safety primitive STL subformulas, denoted by ϕ_S, are chosen from the following hypothesis set of STL templates:

$$\mathcal{P}_S = \{G_{[t_1,t_2)}\neg\rho, \; G_{[t_1,t_2)}F_{[0,t_3)}\neg\rho\} \tag{5}$$

where ρ is a polytopic predicate in the form of (3), $G_{[t_1,t_2)}\neg\rho$ means an unsafe region ρ should not be entered during the time interval $[t_1, t_2)$, while $G_{[t_1,t_2)}F_{[0,t_3)}\neg\rho$ means an unsafe region ρ should not be entered for more than t_3 time units during the time interval $[t_1, t_2)$.

The liveness primitive STL subformulas, denoted by ϕ_P, are chosen from the following hypothesis set of STL templates:

$$\mathcal{P}_P = \{G_{[t_1,t_2)}\rho, F_{[t_1,t_2)}\rho, F_{[t_1,t_2)}G_{[0,t_3)}\rho, G_{[t_1,t_2)}F_{[0,t_3)}\rho\} \tag{6}$$

where the four templates, respectively, mean: during the time interval $[t_1, t_2)$, the agent should always be in region ρ ($G_{[t_1,t_2)}\rho$); the agent should be in region ρ for at least one time instant ($F_{[t_1,t_2)}\rho$); the agent should eventually be in region ρ and stay there for at least t_3 time units ($F_{[t_1,t_2)}G_{[0,t_3)}\rho$); and for every time instant, the agent should eventually be in region ρ within t_3 time units ($G_{[t_1,t_2)}F_{[0,t_3)}\rho$).

Definition 9: Let $S = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i)\}_{i=1}^{N_S}$, ϕ and ψ are the advisory motion STL subformula and the advisory selection STL subformula, respectively, we denote $\text{part}_x(S, \phi) = \{S_{x,\top}, S_{x,\bot}, S_{x,\emptyset}\}$ as the partition of S by ϕ, where $S_{x,\top} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | (\hat{x}^i_{H_i}, 0) \models_S \phi\}$, $S_{x,\bot} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | (\hat{x}^i_{H_i}, 0) \models_S \neg\phi\}$, $S_{x,\emptyset} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | (\hat{x}^i_{H_i}, 0) \not\models_S \phi, (\hat{x}^i_{H_i}, 0) \not\models_S \neg\phi\}$. Similarly, we denote the partition of S by ψ as $\text{part}_y(S, \psi) = \{S_{y,\top}, S_{y,\bot}, S_{y,\emptyset}\}$, where $S_{y,\top} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | (\hat{y}^i_{H_i}, 0) \models_S \psi\}$, $S_{y,\bot} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | (\hat{y}^i_{H_i}, 0) \models_S \neg\psi\}$, $S_{y,\emptyset} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | (\hat{y}^i_{H_i}, 0) \not\models_S \psi, (\hat{y}^i_{H_i}, 0) \not\models_S \neg\psi\}$.

We modified the definition of extended impurity measures in [7, Definition 5.5] specifically for prefixes in the following definition.

Definition 10 (Extended Impurity Measures for Prefixes): We define the following partition weights to describe how the prefixes $\hat{x}^i_{H_i}$ are distributed according to their labels l_i and the formula ϕ:

$$\begin{aligned} p_{x,\top,S}(\phi) &= \frac{\sum_{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S_{x,\top}} \hat{r}\left(\hat{x}^i_{H_i}, \phi, l_i, 0\right)}{\sum_{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S} \left|\hat{r}\left(\hat{x}^i_{H_i}, \phi, l_i, 0\right)\right|} \\ p_{x,\bot,S}(\phi) &= -\frac{\sum_{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S_{x,\bot}} \hat{r}\left(\hat{x}^i_{H_i}, \phi, l_i, 0\right)}{\sum_{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S} \left|\hat{r}\left(\hat{x}^i_{H_i}, \phi, l_i, 0\right)\right|} \end{aligned} \tag{7}$$

where $S_c = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in S | l_i = c\}$.

The partition weights to describe how the prefixes $\hat{y}^i_{H_i}$ are distributed according to their labels l_i and the formula ψ can be defined in a similar way, which are denoted as $p_{y,\top,S}(\psi)$ and $p_{y,\bot,S}(\psi)$, respectively. We also denote

$$\hat{p}(S, c) = \frac{\left|\left\{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) | l_i = c\right\}\right|}{|S|} \tag{8}$$

where $|S|$ denotes the number of attempts in S.

The algorithm for the decision tree construction is shown in Algorithm 1. At each node t, we first obtain the advisory motion STL subformula $t.\phi$. In this paper, we focus on motions of the robot in a sequential order, i.e., one motion ends before another motion starts. For that purpose, we divide the total time into $L_{\max}$ equal parts and first search for the advisory motion STL subformula from the earliest time period at the root node of the tree. There is another constraint $H_i \geq \tau(\phi, l_i)$ on the temporal parameters so that the labeled set of prefixes is possible to be perfectly classified (see Proposition 1 in Section III). We use $\hat{\Theta}_\phi$ to denote the constrained parameter space. $\tau(\phi, l_i)$ for the templates of ϕ in $\mathcal{P}_S$ and $\mathcal{P}_P$ is given in Table I. At each nonterminal (nonleaf) node t, a primitive advisory STL subformula with a structure from the given hypothesis set of templates is parameterized by θ_ϕ, the left child node and the right child node of node t are denoted as t.left and t.right, respectively. For each polytopic predicate, τ, a_k, and b_k will be the elements of θ_ϕ. It should be noted that $H_i \geq \tau(\phi, l_i)$ is only a necessary condition for perfect classification, so there may still exist prefixes that neither strongly satisfy nor strongly violate $\phi(\theta_\phi)$ with θ_ϕ being in the constrained parameter space $\hat{\Theta}_\phi$, which should be strongly penalized in our cost function [see the second term in (9)]. We denote L as the index of the time period starting from 0 to $(L_{\max} - 1)$. It can be proven that $\hat{\Theta}_\phi$ at the root node ($L = 0$) is guaranteed to be nonempty if the lengths of all the prefixes

Algorithm 1 Parameterized Decision Tree Construction for Advisory STL Formula Inference

```
1: procedure $STLtree(\hat{S} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i)\}_{i=1}^{N_{\hat{S}}}, \phi^{\text{path}}, \psi^{\text{path}}, L)$
2: if $stop(\phi^{\text{path}}, L, \hat{S})$ then
3:     Define node $t$ as a leaf (terminal) node
4:     if $\hat{p}(\hat{S}, 1) > \hat{p}(\hat{S}, -1)$ then
5:         $t.\phi \leftarrow \top, t.\psi \leftarrow \top$
6:     else
7:         $t.\phi \leftarrow \bot, t.\psi \leftarrow \bot$
8:     end if
9:     return $t$
10: end if
11: Define node $t$ as a nonterminal node
12: $\hat{\Theta}_\phi \leftarrow \{\theta_\phi | \forall i \in \{1, 2, \ldots, N_{\hat{S}}\}, \tau(\phi(\theta_\phi) \wedge \phi^{\text{path}}, l_i) \leq H_i, t_{\text{s}}(\phi(\theta_\phi)) \in [\frac{T}{L_{\max}} L + \Delta T_{\text{sel}}, \frac{T}{L_{\max}}(L+1)), t_{\text{e}}(\phi(\theta_\phi)) \in (t_{\text{s}}(\phi(\theta_\phi)), \frac{T}{L_{\max}}(L+1)]\}$
13: if $\hat{\Theta}_\phi = \emptyset$ then
14:     $L \leftarrow L - 1$
15:     $\hat{\Theta}_\phi \leftarrow \{\theta_\phi | \forall i \in \{1, 2, \ldots, N_{\hat{S}}\}, \tau(\phi(\theta_\phi) \wedge \phi^{\text{path}}, l_i) \leq H_i, t_{\text{s}}(\phi(\theta_\phi)) \in [\frac{T}{L_{\max}} L + \Delta T_{\text{sel}}, \frac{T}{L_{\max}}(L+1)), t_{\text{e}}(\phi(\theta_\phi)) \in (t_{\text{s}}(\phi(\theta_\phi)), \frac{T}{L_{\max}}(L+1)]\}$
16: end if
17: $t.\phi \leftarrow \arg \min_{\phi \in \mathcal{P}_S \cup \mathcal{P}_P, \theta_\phi \in \hat{\Theta}_\phi} J_x(\hat{S}, \phi^{\text{path}} \wedge \phi(\theta_\phi))$
18: $\{\hat{S}_{x,\top}, \hat{S}_{x,\bot}, \hat{S}_{x,\emptyset}\} \leftarrow part_x(\hat{S}, t.\phi)$
19: if $\hat{p}(\hat{S}_{x,\top}, -1) > 0$ then
20:     $\hat{\Theta}_\psi \leftarrow \{\theta_\psi | \forall i \in \{1, 2, \ldots, N_{\hat{S}}\}, \tau(\psi(\theta_\psi) \wedge \psi^{\text{path}}, l_i) \leq H_i, t_{\text{s}}(\psi(\theta_\psi)) \in [\frac{T}{L_{\max}} L, \frac{T}{L_{\max}} L + \Delta T_{\text{sel}}), t_{\text{e}}(\phi(\theta_\phi)) \in (t_{\text{s}}(\psi(\theta_\psi)), \frac{T}{L_{\max}} L + \Delta T_{\text{sel}}]\}$
21:     $\psi^* \leftarrow \arg \min_{\psi \in \mathcal{P}_S \cup \mathcal{P}_P, \theta_\psi \in \hat{\Theta}_\psi} J_y(\hat{S}_{x,\top}, \psi^{\text{path}} \wedge \psi(\theta_\psi))$
22:     if $J_y(\hat{S}_{x,\top}, \psi^{\text{path}} \wedge \psi^*) = 0$ then
23:         $\{\hat{S}^{\text{r}}_{x,\top}, \hat{S}^{\text{r}}_{x,\bot}\} \leftarrow part_y(\hat{S}_{x,\top}, \psi^*)$
24:     else
25:         $\hat{S}^{\text{r}}_{x,\top} \leftarrow \hat{S}_{x,\top}, \hat{S}^{\text{r}}_{x,\bot} \leftarrow \emptyset, \psi^* \leftarrow \top$
26:     end if
27: else
28:     $\hat{S}^{\text{r}}_{x,\top} \leftarrow \hat{S}_{x,\top}, \hat{S}^{\text{r}}_{x,\bot} \leftarrow \emptyset, \psi^* \leftarrow \top$
29: end if
30: $\psi^*_{\text{s}} \leftarrow STLSelect(\hat{S}, \top, \phi^{\text{path}} \wedge t.\phi, 0, L)$
31: $t.\psi \leftarrow \psi^* \wedge \psi^*_{\text{s}}$
32: $t.left \leftarrow STLtree(\hat{S}^{\text{r}}_{x,\top}, \phi^{\text{path}} \wedge t.\phi, \psi^{\text{path}} \wedge t.\psi, L+1)$
33: $t.right \leftarrow STLtree(\hat{S}_{x,\bot} \cup \hat{S}^{\text{r}}_{x,\bot}, \phi^{\text{path}} \wedge \neg t.\phi, \psi^{\text{path}} \wedge \neg t.\psi, L)$
34: return $t$
35: end procedure
```

are not shorter than ΔT_{sel} ($\Delta T_{\text{sel}} \in [0, (T/L_{\max}))$ is the time horizon for each advisory selection STL subformula). For the left child node of node t, if the new constrained parameter space with L incremented by 1 is not empty, then we search for the advisory motion STL subformula in the next time period; otherwise (Algorithm 1, lines 13–16) we still search the advisory motion STL subformula in the same time period as that for node t (as the set of prefixes at the left child node of node t is a subset of the prefixes at node t, the new constrained parameter space with the same L is guaranteed to be nonempty). For the right child node of node t, as the negation of an advisory motion STL subformula is generally not specific enough for specifying a motion as a feature of success, we still search the advisory motion STL subformula in the same time period as that for node t. As Algorithm 1 is called recursively from the root node to each leaf (terminal) node, we denote the set of attempts at the current node as $\hat{S}$. We find $\phi^* \in \mathcal{P}_P \cup \mathcal{P}_S$ (with $\theta^*_\phi \in \hat{\Theta}_\phi$) that minimizes the following cost function:

TABLE I
$\tau(\phi, l_i)$ FOR DIFFERENT ϕ AND l_i

	$l_i = 1$	$l_i = -1$
$\phi = G_{[t_1,t_2)}\rho$	t_2	t_1
$\phi = F_{[t_1,t_2)}\rho$	t_1	t_2
$\phi = F_{[t_1,t_2)}G_{[0,t_3)}\rho$	$t_1 + t_3$	t_2
$\phi = G_{[t_1,t_2)}F_{[0,t_3)}\rho$	t_2	$t_1 + t_3$
$\phi = G_{[t_1,t_2)}\neg\rho$	t_2	t_1
$\phi = G_{[t_1,t_2)}F_{[0,t_3)}\neg\rho$	t_2	$t_1 + t_3$

$$\begin{aligned} J_x(\hat{S}, \phi^{\text{path}} \wedge \phi(\theta_\phi)) &= \hat{J}_x(\hat{S}, \phi^{\text{path}} \wedge \phi(\theta_\phi)) + \lambda \cdot g_{\text{var}}(\phi, \theta_\phi) \\ &= \sum_{\otimes \in \{\top,\bot\}} (p_{x,\otimes,\hat{S}}(\phi^{\text{path}} \wedge \phi(\theta_\phi)) \cdot MR(\hat{S}_{x,\otimes})) + \eta |\hat{S}_{x,\emptyset}| \\ &\quad + \lambda \cdot g_{\text{var}}(\phi, \theta_\phi) \end{aligned} \tag{9}$$

where $\{\hat{S}_{x,\top}, \hat{S}_{x,\bot}, \hat{S}_{x,\emptyset}\}$ is the partition of $\hat{S}$ by $\phi(\theta_\phi)$

$$MR(\hat{S}) = \min(\hat{p}(\hat{S}, 1), \hat{p}(\hat{S}, -1)),$$

$$g_{\text{var}}(\phi, \theta_\phi) = \begin{cases} \sum_{j=1}^{n} \sum_{k=1}^{m_v} \left(\frac{x_j^k}{\Delta x_j^{\max}} - \frac{1}{m_v} \sum_{k=1}^{m_v} \frac{x_j^k}{\Delta x_j^{\max}} \right)^2 \Big/ m_v \\ \quad \text{if } \phi \in \mathcal{P}_P \\ -\sum_{j=1}^{n} \sum_{k=1}^{m_v} \left(\frac{x_j^k}{\Delta x_j^{\max}} - \frac{1}{m_v} \sum_{k=1}^{m_v} \frac{x_j^k}{\Delta x_j^{\max}} \right)^2 \Big/ m_v \\ \quad \text{if } \phi \in \mathcal{P}_S \end{cases} \tag{10}$$

where ϕ^{path} is the formula associated with the current path (from the root node to the parent node of the current node), η is a large positive number, λ is a positive weighting factor (for tuning of λ, see the example in Section V), n is the dimension of the state x, x_j^k is the jth coordinate value of the kth vertex of the polyhedron enclosed by $\rho(\theta_\phi)$ in (3) and the outer boundaries of the state space, $\Delta x_j^{\max}$ is the maximal variation of the state in the jth dimension in the bounded state space, and m_v is the number of the vertices of this enclosed polyhedron. The first term of the cost function minimizes the extended impurity measure (such that more percentages of the prefixes belong to the same class at the children nodes than those at the parent nodes, so the misclassification rate is decreased by each partition and the subtree generation). The second term of the cost function strongly penalizes the number of attempts that neither strongly satisfy nor strongly violate $\phi(\theta_\phi)$. The third term of the cost function is to slightly increase the conservativeness of the subformula as a control specification by minimizing the sum of variances of the vertices' coordinate values of the obtained region for liveness primitive subformulas as smaller regions are more conservative as waypoints, and maximizing the sum of variances of the vertices' coordinate values of the obtained region for safety primitive subformulas as larger regions are more conservative as obstacles. The attempts are partitioned by ϕ^* into three disjoint sets of attempts (ideally two disjoint sets of attempts, as $|\hat{S}_{x,\emptyset}|$ is strongly penalized).

After the advisory motion STL subformula $t.\phi$ is obtained at each node, we search for the advisory selection STL subformula based on $t.\phi$. If some prefixes that strongly satisfy $t.\phi$ are with label -1 (Algorithm 1, line 19), then we check if there exists a primitive advisory selection STL subformula that can perfectly classify the prefixes of the environment with labels 1 and -1. We use the same hypothesis set of templates for the advisory selection STL subformulas except that we use y (the state of the environment) in (3) instead of x (the state of the robot). We find $\psi^* \in \mathcal{P}_S \cup \mathcal{P}_P$ (with $\theta_\psi^* \in \hat{\Theta}_\psi$, where $\hat{\Theta}_\psi$ is the constrained parameter space for the advisory selection STL subformulas) that minimizes the following cost function:

$$J_y(\hat{S}, \psi^{\text{path}} \wedge \psi(\theta_\psi)) = \sum_{\otimes\in\{\top,\bot\}} (p_{y,\otimes,\hat{S}}(\psi^{\text{path}} \wedge \psi(\theta_\psi)) \cdot MR(\hat{S}_{y,\otimes})) + \eta|\hat{S}_{y,\emptyset}| \tag{11}$$

where $\{\hat{S}_{y,\top}, \hat{S}_{y,\bot}, \hat{S}_{y,\emptyset}\}$ is the partition of $\hat{S}$ by $\psi(\theta_\psi)$, and ψ^{path} is the formula associated with the current path (from the root node to the parent node of the current node).

If such a primitive advisory selection STL subformula ψ^* is found [$J_y(\hat{S}_{x,\top}, \psi^{\text{path}} \wedge \psi^*) = 0$], then ψ^* is set as a part of the advisory selection STL subformula at node t, as when the prefix of the environment strongly satisfies ψ^* and the prefix of the robot strongly satisfies $t.\phi$, it leads to success (label 1) in the training data set. In this case, the attempts in $\hat{S}_{x,\top}$ are further partitioned by ψ^* into two disjoint sets of attempts, denoted as $\hat{S}^r_{x,\top}$ and $\hat{S}^r_{x,\bot}$ (as the third set $\hat{S}^r_{x,\emptyset}$ must be empty if $J_y(\hat{S}_{x,\top}, \psi^{\text{path}} \wedge \psi^*) = 0$). If such ψ^* is not found, then ψ^* is set as $\top$, thus $\hat{S}^r_{x,\top}$ is the same as $\hat{S}_{x,\top}$ and $\hat{S}^r_{x,\bot}$ is empty. To make sure the advisory selection STL subformula is unique for each attempt with label 1 in the training data set, we use Algorithm 2 to first find the set of prefixes of the environment with label 1 in the current set $\hat{S}$ (Algorithm 2, lines 3–12), denoted as $\hat{S}'$, then find ψ_s^* that can best classify the prefixes of the environment in $\hat{S}'$ with the corresponding prefixes of the robot that strongly satisfy the advisory motion STL $t.\phi$ and $\neg t.\phi$, respectively. The advisory selection STL subformula at node t is set as $t.\psi = \psi^* \wedge \psi_s^*$. In this way, at node t, for any attempt $(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i)$ with label $l_i = 1$, if it is assigned to the left subtree $((\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in \hat{S}^r_{x,\top})$, then $((\hat{x}^i_{H_i}, 0) \models_S t.\phi) \wedge ((\hat{y}^i_{H_i}, 0) \models_S t.\psi)$; if it is assigned to the right subtree $((\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i) \in \hat{S}_{x,\bot} \cup \hat{S}^r_{x,\bot})$, then $((\hat{x}^i_{H_i}, 0) \models_S \neg t.\phi) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \neg t.\psi)$ (see the Proof of Proposition 2).

At the beginning of the training process, $\hat{S}$ is set as S (the training data set of attempts), ϕ^{path} and ψ^{path} are set as $\top$, and L is set as 0. The procedure is called recursively to construct the left and right subtrees for $\hat{S}^r_{x,\top}$ and $\hat{S}_{x,\bot} \cup \hat{S}^r_{x,\bot}$, respectively. We set a condition *stop* for determining a leaf (terminal) node, which can be a percentage (e.g., 90%) of the attempts at the current node belonging to the same class (positive or negative) or the last time period (e.g., $L_{\max} = 3$) being reached. After the decision tree is constructed, every node of the tree is associated with a pair of advisory motion and selection STL subformulas (the leaf node is associated with $(\top, \top)$ or $(\bot, \bot)$ depending on whether the attempts at the leaf node are mostly associated with label 1 or -1, as given in lines 4–8 of Algorithm 1).

Algorithm 2 Advisory Selection STL Subformulas Inference Subroutine

```
1: procedure $STLSelect(\hat{S} = \{(\hat{x}^i_{H_i}, \hat{y}^i_{H_i}, l_i)\}_{i=1}^{N_{\hat{S}}}, \psi_s^{\text{path}}, \phi, h, L)$
2: if $h = 0$ then
3:     $j \leftarrow 0$
4:     for $i = 1 : N_{\hat{S}}$ do
5:         if $((\hat{x}^i_{H_i}, 0) \models_S \phi) \wedge (l_i = 1)$ then
6:             $j \leftarrow j+1, \hat{x}'^j_{H'_j} \leftarrow \hat{x}^i_{H_i}, \hat{y}'^j_{H'_j} \leftarrow \hat{y}^i_{H_i}, l'_j = 1$
7:         end if
8:         if $((\hat{x}^i_{H_i}, 0) \models_S \neg\phi) \wedge (l_i = 1)$ then
9:             $j \leftarrow j+1, \hat{x}'^j_{H'_j} \leftarrow \hat{x}^i_{H_i}, \hat{y}'^j_{H'_j} \leftarrow \hat{y}^i_{H_i}, l'_j = -1$
10:        end if
11:    end for
12:    $N_{\hat{S}'} \leftarrow j, \hat{S}' \leftarrow \{(\hat{x}'^j_{H'_j}, \hat{y}'^j_{H'_j}, l'_j)\}_{j=1}^{N_{\hat{S}'}}$
13:    if $\forall j > 0, l'_j = 1$ then
14:        return $\top$
15:    end if
16: else
17:    $\hat{S}' \leftarrow \hat{S}$
18: end if
19: if $stop_{\text{sel}}(\psi_s^{\text{path}}, h, \hat{S}')$ then
20:    if $\hat{p}(\hat{S}', 1) > \hat{p}(\hat{S}', -1)$ then
21:        return $\top$
22:    else
23:        return $\bot$
24:    end if
25: end if
26: $\hat{\Theta}_\psi \leftarrow \{\theta_\psi | \forall i \in \{1, 2, \dots, N_{\hat{S}'}\}, \tau(\psi(\theta_\psi) \wedge \psi^{\text{path}}, l_i)$
        $\leq H_i, t_s(\psi(\theta_\psi)) \in [\frac{T}{L_{\max}}L, \frac{T}{L_{\max}}L + \Delta T_{\text{sel}}),$
        $t_e(\phi(\theta_\psi)) \in (t_s(\psi(\theta_\psi)), \frac{T}{L_{\max}}L + \Delta T_{\text{sel}}]\}$
27: $\psi_s^* = \arg\min_{\psi\in\mathcal{P}_P\cup\mathcal{P}_S, \theta_\psi\in\hat{\Theta}_\psi} J_y(\hat{S}', \psi_s^{\text{path}} \wedge \psi(\theta_\psi))$
28: $\{\hat{S}'_{y,\top}, \hat{S}'_{y,\bot}, \hat{S}'_{y,\emptyset}\} \leftarrow part_y(\hat{S}', \psi_s^*)$
29: $\psi_{s,l} \leftarrow \psi_s^* \wedge STLSelect(\hat{S}'_{y,\top}, \psi_s^{\text{path}} \wedge \psi_s^*, \phi, h+1, L)$
30: $\psi_{s,r} \leftarrow \neg\psi_s^* \wedge STLSelect(\hat{S}'_{y,\bot}, \psi_s^{\text{path}} \wedge \neg\psi_s^*, \phi, h+1, L)$
31: return $\psi_{s,l} \vee \psi_{s,r}$
32: end procedure
```

The complexity of Algorithm 2 for the average case can be computed through the Akra–Bazzi method as follows [7]:

$$\Theta\left(N\cdot\left(1+\int_1^N \frac{g(u)}{u^2}du\right)\right)$$

where $g(N)$ is the complexity of the particle swarm optimization algorithm for N prefixes, $\Theta(\cdot)$ denotes the two-sided asymptotic notation for complexity bound. Thus, the complexity of Algorithm 1 for the average case is as follows:

$$\Theta\left(N\cdot\left(1+\int_1^N \frac{g(u)+s(u)}{u^2}du\right)\right)$$

where $s(u)$ is the complexity of Algorithm 2.

Algorithm 3 Tree to Set of Advices

```
1: procedure $Tree2Advice(t)$
2: if $t$ is a leaf (terminal) node then
3:     return $(t.\phi, t.\psi)$
4: end if
5: for $i = 1 : size(Tree2Advice(t.left))$ do
6:     $\Phi\{i\} \leftarrow t.\phi \wedge Tree2Advice(t.left)\{i\}(1)$
7:     $\Psi\{i\} \leftarrow t.\psi \wedge Tree2Advice(t.left)\{i\}(2)$
8:     $\Gamma\{i\} \leftarrow (\Phi\{i\},\ \Psi\{i\})$
9: end for
10: $i \leftarrow size(Tree2Advice(t.left))$
11: for $j = 1 : size(Tree2Advice(t.right))$ do
12:     $\Phi\{i+j\} \leftarrow \neg(t.\phi) \wedge Tree2Advice(t.right)\{j\}(1)$
13:     $\Psi\{i+j\} \leftarrow \neg(t.\psi) \wedge Tree2Advice(t.right)\{j\}(2)$
14:     $\Gamma\{i+j\} \leftarrow (\Phi\{i+j\},\ \Psi\{i+j\})$
15: end for
16: remove $\perp$ from $\Phi$ and $\Psi$, remove $(\perp,\perp)$ from $\Gamma$
17: return $\Gamma$
18: end procedure
```

The algorithm to transform the obtained decision tree to the set of advices is shown in Algorithm 3. The process is modified from Algorithm 2 in [7], but the advisory motion (selection) logic subformulas (along the left subtree) or the negation of them (along the right subtree) are connected recursively only with conjunction operators, and we obtain a set of advices instead of a single STL formula. We use size(Γ) to denote the number of advices in Γ. For each advice $\gamma = (\phi, \psi)$, we use $\gamma(1)$ and $\gamma(2)$ to denote ϕ and ψ for brevity. We also use Φ and Ψ to denote the set of advisory motion STL formulas and the set of advisory selection STL formulas, respectively.

Proposition 2: By the advices obtained from Algorithms 1–3, Problem 1 is solved if the following conditions are met:

1) $stop(\phi^{\text{path}}, L, \hat{S})$ in Algorithm 1 is achieved as all of the prefixes at the current node belong to the same class;
2) $stop_{\text{sel}}(\psi_s^{\text{path}}, h, \hat{S}')$ in Algorithm 2 is achieved as all of the prefixes at the current node belong to the same class;
3) $\hat{S}'_{y,\emptyset} = \emptyset$, $\hat{S}_{x,\emptyset} = \emptyset$ in each computation for Algorithms 1 and 2.

Proof: See the Appendix. ■

Proposition 2 provides the theoretical guarantee for solving Problem 1. The conditions for solving Problem 1, however, may lead to overfitting in practice. Therefore, we present Proposition 3 for solving a relaxed version of Problem 1.

Proposition 3: By the advices obtained from Algorithms 1–3, the coverage property 1) and the uniqueness property 3) of Problem 1 are completely satisfied, and the soundness property 2) is satisfied for no less than $(1-\zeta)$ fraction of the prefixes in the training data set, if the following conditions are met:

1) $stop(\phi^{\text{path}}, L, \hat{S})$ in Algorithm 1 is achieved as no less than $(1-\zeta)$ fraction of the prefixes at the current node belong to the same class;
2) $stop_{\text{sel}}(\psi_s^{\text{path}}, h, \hat{S}')$ in Algorithm 2 is achieved as all of the prefixes at the current node belong to the same class;
3) $\hat{S}'_{y,\emptyset} = \emptyset$, $\hat{S}_{x,\emptyset} = \emptyset$ in each computation for Algorithms 1 and 2.

Proof: See the Appendix. ■

In Sections V and VI, it is demonstrated that in both the two case studies, the three conditions can be met through our proposed method.

B. Advisory Controller Synthesis for Shared Autonomy

In order to utilize the inferred knowledge or specifications as features of success for improving the performances of the human operators, we design an advisory controller to drive the robot to satisfy the inferred advisory motion STL formula based on the advice selected according to the inferred advisory selection STL formula.

We first discretize the system into the following form:

$$x^{k+1} = F(x^k, u^k) \tag{12}$$

where x^k and u^k are the states and the control inputs at the time indices $k = 0, 1, \ldots$

For each advisory motion STL formula $\Phi\{i\}$, the optimization problem is formulated as follows:

$$\begin{aligned} &\arg\min_{\mathbf{u}}\ J(\mathbf{u}) \\ &\text{s.t. } r(x(\mathbf{u}), \Phi\{i\}, 0) \geq 0 \end{aligned} \tag{13}$$

where $\mathbf{u} = \{u^0, u^1, \ldots, u^N\}$ is the control input signal (N is the number of time points in the control horizon, and we assume that the control horizon is sufficiently long such that $NT_s \geq \max_i \|\Phi\{i\}\|$, where T_s is the sampling time), $x(\mathbf{u}) = \{x^0, x^1, \ldots, x^N\}$ is the system trajectory starting from a given initial condition $x^0 \in \mathbb{X}$ under the control input signal $\mathbf{u}$. The performance measure $J(\mathbf{u})$ can be set as the control effort $\|\mathbf{u}\|_2$ or $\|\mathbf{u}\|_1$. For linear systems, the above-mentioned optimization problem can be converted to a mixed-integer linear programming (MILP) problem [20], [21] and it can be solved efficiently by MILP solvers.

Next, we elaborate the algorithm of shared autonomy between the human operator and the controller, which is shown in Algorithm 4. The human operator can decide whether to switch ON or switch OFF the advisory controller at anytime except for two scenarios that will be elaborated later. At specific time instant t_k such that $|t_k - \frac{LT}{L_{\max}} - \Delta T_{\text{sel}}| < (T_s/2)$, where T is the specified time for the task, T_s is the sampling time (i.e., $\forall k \geq 0$, $t_{k+1} - t_k = T_s$), if the prefix of the

Algorithm 4 Shared Autonomy Between the Human Operator and the Controller

```
1: $ControllerState \leftarrow 0$, $AdviceIndex \leftarrow 0$
2: $k \leftarrow 0$, $SafetyAlert \leftarrow 0$
3: while $t_k < T$ do
4:   if there exists a safety advisory motion STL formula that is about to be strongly violated in the next $\Delta N_t$ time interval if continuing with the current state and input then
5:     $SafetyAlert \leftarrow 1$
6:   end if
7:   if there exists integer $L \in [0, L_{\max}]$ such that $|t_k - \frac{LT}{L_{\max}} - \Delta T_{\mathrm{sel}}| < \frac{T_s}{2}$ then
8:     for $i = 1 : size(\Psi)$ do
9:       if the prefix of the environment up to $t_k$ does not strongly violate $\Psi\{i\}$ then
10:        $AdviceIndex \leftarrow i$
11:      end if
12:    end for
13:  end if
14:  if $ControllerState = 1$ then
15:    if $SwitchingOffCommand = 1$ or $t_e(\Phi\{AdviceIndex\}) \leq t_k$ then
16:      $ControllerState \leftarrow 0$
17:      $u^k \leftarrow$ Human input
18:    else
19:      $u^k \leftarrow \hat{u}^{*k}$
20:    end if
21:  end if
22:  if $ControllerState = 0$ then
23:    if $AdviceIndex \neq 0 \wedge (SwitchingOnCommand = 1 \vee SafetyAlert = 1)$ then
24:      $ControllerState \leftarrow 1$
25:      $\Phi\{AdviceIndex\} \leftarrow [\Phi\{AdviceIndex\}]_0^{t_k}$
26:      Solve (13) with $i = AdviceIndex$
27:      if (13) is infeasible then
28:        $ControllerState \leftarrow 0$
29:        $u^k \leftarrow$ Human input
30:      else
31:        obtain $\mathbf{u}^*$ from (13)
32:        $\hat{u}^{*k+q} \leftarrow u^{*q}$ $(q = 0, 1, \ldots)$, $u^k \leftarrow \hat{u}^{*k}$
33:      end if
34:    else
35:      $u^k \leftarrow$ Human input
36:    end if
37:  end if
38:  $k \leftarrow k + 1$
39: end while
```

environment up to the current time t_k does not strongly violate a advisory selection STL formula $\Psi\{i\}$ (the prefix may not strongly satisfy the advisory selection STL formula $\Psi\{i\}$ yet as $\Psi\{i\}$ may have subformulas that contain future time intervals), then the ith advice is selected (note that i may not be unique, but the subformula at the current time period is unique). Then, whenever the advisory controller is switched ON, the advisory controller computes inputs that satisfy a modified version of the ith advisory motion STL formula $\Phi\{i\}$ until another time instant $t_{k'}$ such that $|t_{k'} - \frac{(L+1)T}{L_{\max}} - \Delta T_{\mathrm{sel}}| < \frac{T_s}{2}$. The formula is modified as the trajectory before the current time is already fixed and the formula should not specify any time instant that is already passed (Algorithm 4, line 25). We use $[\phi]_t^{t_k}$ to denote the formula modified from the STL formula ϕ when ϕ is evaluated at time t and the current time is t_k. $[\phi]_t^{t_k}$ can be calculated recursively as follows (we use π_t to denote the atomic predicate π evaluated at time t):

$$[\pi]_t^{t_k} := \begin{cases} \pi_t, & \text{if } t > t_k \\ \top, & \text{if } t \leq t_k \text{ and } f(x(t)) > 0 \\ \bot, & \text{if } t \leq t_k \text{ and } f(x(t)) \leq 0 \end{cases}$$

$$[\neg\phi]_t^{t_k} := \neg[\phi]_t^{t_k}$$

$$[\phi_1 \wedge \phi_2]_t^{t_k} := [\phi_1]_t^{t_k} \wedge [\phi_2]_t^{t_k}$$

$$[F_{[t_1,t_2)}\phi]_t^{t_k} := \bigvee_{t' \in (t+[t_1,t_2))} [\phi]_{t'}^{t_k}$$

$$[G_{[t_1,t_2)}\phi]_t^{t_k} := \bigwedge_{t' \in (t+[t_1,t_2))} [\phi]_{t'}^{t_k}.$$

If an advisory motion STL formula $\Phi\{i\}$ is evaluated at time 0 (which is the usual case when the task starts at time 0), then the modified formula is $[\Phi\{i\}]_0^{t_k}$. The control inputs are recomputed for the remaining time to satisfy the modified formula $[\Phi\{i\}]_0^{t_k}$ based on the current state (Algorithm 4, lines 31 and 32).

There are two scenarios where the advisory controller is automatically switched ON or switched OFF.

1) Controller Mode Automatically Switched ON *for Safety:* When the advisory controller is OFF ($ControllerState = 0$), if the robot is going to violate a near future safety specification given the current state and input, the advisory controller is automatically switched ON to avoid safety specification violations. Specifically, for a given short time interval ΔN_t, if there exists a safety advisory subformula that is about to be strongly violated in the next ΔN_t time interval with the current state and input (Algorithm 4, lines 4–6), then the advisory controller is automatically switched ON to avoid the imminent safety violation.

2) Controller Mode Automatically Switched OFF *for unsatisfiability or End of Effect Time:* When the advisory controller is ON (*ControllerState* = 1), if the advisory motion STL formula is computed as unsatisfiable, or the current time passes the end-effect time of the advisory motion STL formula, then the advisory controller is automatically switched OFF and the human operator takes the control of the robot.

C. Advisory STL Formula Refinement

After designing the advisory controller, we can use the newly generated trajectories (prefixes) from the shared autonomy between the human operator and the advisory controller to further refine the set of advices. In generating those trajectories, the advisory controller is always kept ON until it is automatically switched OFF or the human operators have to switch it OFF to achieve tasks that cannot be done by the advisory controller (such as grasping or releasing a bottle). The algorithm for the refinement process is shown in

Algorithm 5 Advisory STL Formula Refinement With Newly Generated Prefixes

```
1: procedure $RefineTree(\tilde{S}, \phi^{\text{path}}, \psi^{\text{path}}, L, t)$
2: if $\widetilde{stop}(\phi^{\text{path}}, L, \tilde{S})$ then
3:     Define node $t_{\text{new}}$ as a leaf (terminal) node
4:     if $\tilde{S}$ is empty then
5:         $t_{\text{new}}.\phi \leftarrow \bot$, $t_{\text{new}}.\psi \leftarrow \bot$, return $t_{\text{new}}$
6:     else
7:         if $\hat{p}(\tilde{S}, 1) > \hat{p}(\tilde{S}, -1)$ then
8:             if $t$ is a leaf (terminal) node then
9:                 $t_{\text{new}}.\phi \leftarrow \top$, $t_{\text{new}}.\psi \leftarrow \top$, return $t_{\text{new}}$
10:            end if
11:        else
12:            $t_{\text{new}}.\phi \leftarrow \bot$, $t_{\text{new}}.\psi \leftarrow \bot$, return $t_{\text{new}}$
13:        end if
14:    end if
15:    Return $t_{\text{new}}$
16: end if
17: Define node $t_{\text{new}}$ as a nonterminal node
18: if $t$ is a leaf (terminal) node then
19:     $t_{\text{new}} \leftarrow STLtree(\tilde{S}, \phi^{\text{path}}, \psi^{\text{path}}, L)$
20:     Return $t_{\text{new}}$
21: else
22:     $\{\hat{S}_{y,\top}, \hat{S}_{y,\bot}, \hat{S}_{y,\emptyset}\} \leftarrow part_y(\hat{S}, t.\psi)$
23:     $\{\tilde{S}_{y,\top}, \tilde{S}_{y,\bot}, \tilde{S}_{y,\emptyset}\} \leftarrow part_y(\tilde{S}, t.\psi)$
24:     $\tilde{\Theta}_{\phi} \leftarrow \{\theta_{\phi} | \forall i \in \{1, 2, \ldots, N_{\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top}}\}, \tau(\phi(\theta_{\phi}) \wedge \phi^{\text{path}}, l_i) \leq H_i, t_{\text{s}}(\phi(\theta_{\phi})) \in [\frac{T}{L_{\max}} L + \Delta T_{\text{sel}}, \frac{T}{L_{\max}}(L+1)), t_{\text{e}}(\phi(\theta_{\phi})) \in (t_{\text{s}}(\phi(\theta_{\phi})), \frac{T}{L_{\max}}(L+1)]\}$
25:     $\tilde{\Theta}_{\psi} \leftarrow \{\theta_{\psi} | \forall i \in \{1, 2, \ldots, N_{\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top}}\}, \tau(\psi(\theta_{\psi}) \wedge \psi^{\text{path}}, l_i) \leq H_i, t_{\text{s}}(\psi(\theta_{\psi})) \in [\frac{T}{L_{\max}} L, \frac{T}{L_{\max}} L + \Delta T_{\text{sel}}), t_{\text{e}}(\phi(\theta_{\psi})) \in (t_{\text{s}}(\psi(\theta_{\psi})), \frac{T}{L_{\max}} L + \Delta T_{\text{sel}}]\}$
26:     if $\tilde{\Theta}_{\phi} = \emptyset$ or $\tilde{\Theta}_{\psi} = \emptyset$ then
27:         $L \leftarrow L - 1$, repeat Lines 24-25 end if
28:     if $\hat{J}_x(\tilde{S}_{y,\top}, \phi^{\text{path}} \wedge t.\phi) > \varepsilon$ and $\hat{J}_x(\tilde{S}_{y,\top}, \phi^{\text{path}} \wedge t.\phi) > \hat{J}_x(\hat{S}_{y,\top}, \phi^{\text{path}} \wedge t.\phi)$ then
29:         $\phi^* \leftarrow \arg\min_{\phi \in \mathcal{P}_{\text{P}} \cup \mathcal{P}_{\text{S}}, \theta_{\phi} \in \tilde{\Theta}_{\phi}} J_x(\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top}, \phi^{\text{path}} \wedge \phi(\theta_{\phi}))$
30:         $\psi^* \leftarrow \arg\min_{\psi \in \mathcal{P}_{\text{P}} \cup \mathcal{P}_{\text{S}}, \theta_{\psi} \in \tilde{\Theta}_{\psi}} J_y(\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top}, \psi^{\text{path}} \wedge \psi(\theta_{\psi}))$
31:         if $J_y(\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top}, \psi^{\text{path}} \wedge \psi^*) \leq \varepsilon$ then
32:             $t_{\text{new}}.\phi \leftarrow t.\phi$
33:             $\{\tilde{S}^{\text{r}}_{y,\top}, \tilde{S}^{\text{r}}_{y,\bot}, \tilde{S}^{\text{r}}_{y,\emptyset}\} \leftarrow part_y(\tilde{S}_{y,\top}, t_{\text{new}}.\psi)$
34:         else if $\hat{J}_x(\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top}, \phi^{\text{path}} \wedge \phi^*) \leq \varepsilon$
35:             $t_{\text{new}}.\phi \leftarrow t.\phi \wedge \phi^*$, $\psi^* \leftarrow \top$
36:             $\{\tilde{S}^{\text{r}}_{y,\top}, \tilde{S}^{\text{r}}_{y,\bot}, \tilde{S}^{\text{r}}_{y,\emptyset}\} \leftarrow part_x(\tilde{S}_{y,\top}, t_{\text{new}}.\phi)$
37:         else
38:             $t_{\text{new}}.\phi \leftarrow t.\phi$, $\psi^* \leftarrow \top$
39:             $\tilde{S}^{\text{r}}_{y,\top} \leftarrow \tilde{S}_{y,\top}$, $\tilde{S}^{\text{r}}_{y,\bot} \leftarrow \emptyset$
40:         end if
41:     else
42:         $t_{\text{new}}.\phi \leftarrow t.\phi$, $\psi^* \leftarrow \top$, $\tilde{S}^{\text{r}}_{y,\top} \leftarrow \tilde{S}_{y,\top}$, $\tilde{S}^{\text{r}}_{y,\bot} \leftarrow \emptyset$
43:     end if
44: end if
```

Algorithm 5 STL Refinement From New Prefixes (Continued)

```
45: if $STLSelect(\tilde{S}, \psi^{\text{path}} \wedge t.\psi, \phi^{\text{path}} \wedge \phi^*, 0, L)$ returns $\top$ then $\psi^*_{\text{s}} \leftarrow t.\psi$
46: else
47:     $\psi^*_{\text{s}} \leftarrow STLSelect(\tilde{S}, \top, \phi^{\text{path}} \wedge \phi^*, 0, L)$
48: end if
49: $t_{\text{new}}.\psi \leftarrow \psi^* \wedge \psi^*_{\text{s}}$
50: $t_{\text{new}}.left \leftarrow RefineTree(\tilde{S}^{\text{r}}_{y,\top}, \phi^{\text{path}} \wedge t.\phi, \psi^{\text{path}} \wedge t.\psi, L+1, t.left)$
51: $t_{\text{new}}.right \leftarrow RefineTree(\tilde{S}_{y,\bot} \cup \tilde{S}^{\text{r}}_{y,\bot}, \phi^{\text{path}} \wedge \neg t.\phi, \psi^{\text{path}} \wedge \neg t.\psi, L, t.right)$
52: return $t_{\text{new}}$
53: end procedure
```

Algorithm 5. We use S_{new} to denote the set of newly generated trajectories (prefixes) from the shared autonomy and denote $S_{\text{all}} \triangleq S \cup S_{\text{new}}$. We use $\tilde{S}$ and $\hat{S}$ to denote the sets of attempts in S_{all} and S at the current node t_{new}, respectively. At the start of the refinement process, $\tilde{S}$ is set as S_{all}, ϕ^{path} and ψ^{path} are both set as $\top$, L is set as 0, and t is set as the root node of the original tree. During the refinement process, for each node t_{new} that corresponds to a leaf (terminal) node t in the original tree, Algorithm 1 is called instead to search for the subtree of advisory STL formulas (Algorithm 5, lines 18–20). For each node t_{new} that corresponds to a nonterminal node t in the original tree, we first partition the attempts in $\tilde{S}$ and $\hat{S}$ according to the original advisory selection STL subformula $t.\psi$. For the attempts in $\tilde{S}_{y,\top}$ and $\hat{S}_{y,\top}$ (i.e., the attempts of which the prefixes of the environment strongly satisfy $t.\psi$ in $\tilde{S}$ and $\hat{S}$, respectively), if the cost function $\hat{J}_x(\tilde{S}_{y,\top}, \phi^{\text{path}} \wedge t.\phi)$ (which measures how well the original advisory motion STL subformula classifies the successful and failed attempts in $\tilde{S}_{y,\top}$) is larger than an error threshold ε and also larger than $\hat{J}_x(\hat{S}_{y,\top}, \phi^{\text{path}} \wedge t.\phi)$ (which measures how well the original advisory motion STL subformula classifies the successful and failed attempts in $\hat{S}_{y,\top}$), then either the advisory motion STL subformula or the advisory selection STL subformula at the current node needs to be modified. If the successful and failed attempts in the trajectory space of the environment in $(\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top})$ can be classified by a primitive advisory selection STL subformula with the cost function value no larger than ε, then the obtained advisory selection STL subformula ψ^* is kept as a modification to the advisory selection STL subformula for node t_{new}, while $t_{\text{new}}.\phi$ is set as $t.\phi$ (Algorithm 5, lines 31–33). On the other hand, if the successful and failed attempts in the trajectory space of the environment in $(\tilde{S}_{y,\top} \setminus \hat{S}_{y,\top})$ are mixed together while those in the trajectory space of the robot can be classified by a primitive advisory motion STL subformula with the cost function value no larger than ε, then the obtained advisory motion STL subformula ϕ^* is added as a modification to the advisory motion STL subformula for node t_{new}, while ψ^* is set as $\top$ (Algorithm 5, lines 34–36).

To make sure the advisory selection STL subformula is unique for each prefix with label 1, we do the following: if

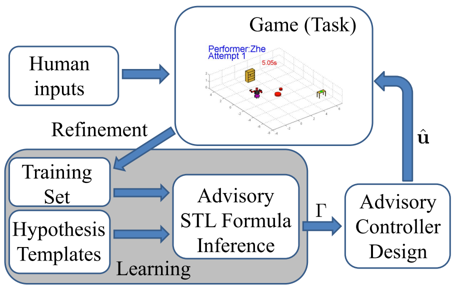


Fig. 4. Block diagram of the advisory STL formula inference (learning), controller design, and refinement process.

the original node t is a nonterminal node, and the original advisory selection STL subformula $t.\psi$ can perfectly classify the prefixes of the environment with label 1 with the corresponding prefixes of the robot that strongly satisfy the advisory motion STL $t.\phi$ and $\neg t.\phi$, respectively, in $\tilde{S}$, then ψ_s^* is set as $t.\psi$ (Algorithm 5, line 45); otherwise, we still use Algorithm 2 to find ψ_s^* that can best classify the prefixes of the environment with label 1 with the corresponding prefixes of the robot that strongly satisfy the advisory motion STL $t.\phi$ and $\neg t.\phi$, respectively (Algorithm 5, line 47). Finally, the advisory selection STL subformula at node t_{new} is set as $t_{\text{new}}.\psi = \psi^* \wedge \psi_s^*$.

The stopping criterion $\tilde{stop}$ is similar with *stop* in Algorithm 1 except for the following: for an advisory motion STL subformula $\phi_j = \hat{\phi}_{j,1} \wedge \hat{\phi}_{j,2} \wedge \cdots \wedge \hat{\phi}_{j,q_j}$, the generated trajectories satisfying $\hat{\phi}_{j,k}$ could imply that they also satisfy $\hat{\phi}_{j,k'}(k' > k)$ when the advisory controller is kept ON, so if an original node t is nonterminal while the new node t_{new} already satisfies the purity criterion (e.g., 90% of the attempts at node t_{new} belong to the same class) with label 1, then the node t_{new} should not terminate (earlier motions leading to success does not mean that subsequent motions are not necessary for achieving success); on the other hand, if the new node t_{new} already satisfies the purity criterion with label -1, then the node t_{new} should terminate (earlier motions leading to failure implies that the whole sequence of motions lead to failure). The block diagram of the inference (learning), controller design, and refinement process is shown in Fig. 4.

V. Case Study I

In this first case study, we apply the proposed approach on a game we designed on a BOW simulator.[2]

The BOW simulator was developed in [24] to help mobility impaired individuals accomplish certain day-to-day tasks using the BOW robot. The system consists of a dual arm BOW robot (manipulator arm) mounted on a powered wheelchair base [25]. The input provided by the human operator can be either desired velocity of a specified point on the robot or desired angular velocity of a specific frame. In our designed interface, human operators only provide desired velocity of the end-effector of the left arm of the BOW robot using a keyboard as the input device. A constrained convex quadratic optimization problem (provided in the simulator) at the kinematic level is then solved to automatically compute the joint angle motion of both the manipulator arm and the base, so that the actual end-effector velocity is as close to the human operator input as possible. In addition, the problem also respects different physical constraints, such as collision avoidance, joint limits, and singularity avoidance, and prevents unnatural integrated motion of the arm and the base.

[2]https://github.com/rpiRobotics/baxter-on-wheels-sim

The advisory controller is designed for the mobile wheelchair base, keeping the joint angles of the manipulator arm fixed. When the advisory controller is turned OFF, the human operator steers the BOW robot by providing the velocity input for the end-effector of the left arm of the BOW robot. This results in both the arm and the base being controlled. On the other hand, when the advisory controller is turned ON, either by the human operator or automatically, the wheelchair base is controlled in order to move the BOW robot to satisfy the inferred advisory motion STL formula.

As the wheelchair base of the robot is moving in a 2-D plane, the dynamics of the wheelchair base can be expressed by the following equations:

$$\begin{aligned} \dot{x} &= \frac{v_r + v_l}{2}\cos(\theta_w) = v\cos(\theta_w) \\ \dot{y} &= \frac{v_r + v_l}{2}\sin(\theta_w) = v\sin(\theta_w) \\ \dot{\theta}_w &= \frac{v_r - v_l}{2d} = \omega \end{aligned} \tag{14}$$

where θ_w denotes the orientation of the wheelchair base, v_r and v_l are the wheel speeds of the right and left wheels, respectively, v and ω are the linear and angular velocities of the robot, respectively, and d is the distance of any one wheel from the center of the robot base. We feedback linearize the system as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos(\theta_w) & -\sin(\theta_w) \\ \sin(\theta_w) & \cos(\theta_w) \end{bmatrix} \begin{bmatrix} \dot{v} \\ v\omega \end{bmatrix}. \tag{15}$$

We choose the intermediate control inputs to the robot to be $\dot{v}$ and $v\omega$ such that

$$\begin{bmatrix} \dot{v} \\ v\omega \end{bmatrix} = \begin{bmatrix} \cos(\theta_w) & \sin(\theta_w) \\ -\sin(\theta_w) & \cos(\theta_w) \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{16}$$

where u_x and u_y are the new control inputs to be determined, we have

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}. \tag{17}$$

These new control inputs u_x and u_y are then generated by solving (13) when both u_x and u_y are bounded by $[-1, 1]$.

We designed a game based on the BOW simulator. At each attempt, the locations of the shelf and the table are generated randomly in the work space, the orientations of the shelf and the table are either aligned with the x-axis or y-axis. We set two moving obstacles (one faster and smaller moving obstacle that moves at twice the speed of the other slower and larger moving obstacle) and these moving obstacles are moving toward the BOW robot from the start of the game. If the slower moving obstacle is blocking the way in front of the faster moving obstacle, the faster moving obstacle moves in the direction perpendicular to the direction toward the slower

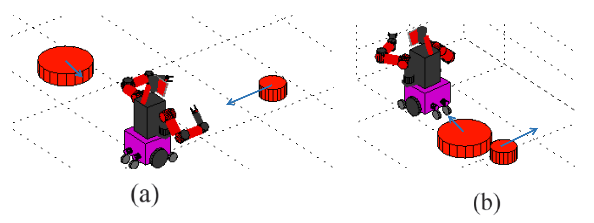


Fig. 5. (a) Two moving obstacles are moving toward the BOW robot. (b) Faster (smaller) moving obstacle is moving in the direction perpendicular to the direction toward the slower (larger) moving obstacle when the slower moving obstacle is blocking the way in front of the faster moving obstacle.

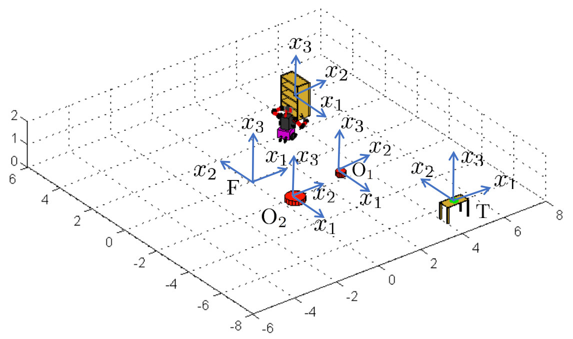


Fig. 6. Coordinate transformation between the fixed reference frame and object-centered reference frames.

moving obstacle (as shown in Fig. 5). The initial positions of the two moving obstacles are randomly generated in the entire workspace.

The success of the task is achieved by the BOW robot if and only if the following goals are achieved within 40 s: 1) first, go to the shelf and grasp the bottle on the shelf; 2) then, go to the table and put the bottle on the target spot on the table; and 3) avoid two moving obstacles during the whole process. Whenever the BOW robot touches the moving obstacles, the game is ended immediately due to safety failure. If the BOW robot successfully avoids the moving obstacles, but does not finish the task within the specified time, then the game is ended due to performance failure.

To ensure the generality of the inferred set of advices when the locations of the objects change, we infer the advisory STL subformula in different object-centered reference frames (the objects in this case are the bottle on the shelf, the target spot on the table, the center of the faster moving obstacle, and the center of the slower moving obstacle). The locations of the center of the wheelchair base in the fixed reference frame and the object-centered reference frame can be transformed as follows (see Fig. 6):

$$x^o = R_{\mathrm{FO}}^T\big(x^f - x_o^f\big) \tag{18}$$

where R_{FO} is the rotation matrix from the fixed reference frame to the object-centered reference frame, x^f and x^o are the position (state) of the BOW robot in the fixed reference frame and the object-centered reference frame, respectively, and x_o^f is the position (state) of the object in the fixed reference frame.

The task can be decomposed into two subtasks: grasping the bottle on the shelf and releasing the bottle on the target spot on the table. We first infer the advisory formulas from the training data set for grasping the bottle on the shelf, and the inference of advisory formulas for releasing the bottle on the target spot on the table is performed in a similar manner. Based on the average time needed to grasp the bottle, we define success of the first subtask as grasping the bottle on the shelf within 24 s while avoiding the two moving obstacles during the whole process. A human demonstrator first generates 30 trajectories (prefixes) from 30 different attempts (19 successes and 11 failures) to form the training data set. The principles of generating the trajectories in the training data set are as follows:

1) keep the variances of the trajectories as large as possible, i.e., always try to use different approaches in each attempt;
2) only keep the trajectories that have the results of clear success or failure, i.e., discard the trajectories that lead to a success in small robustness margins (e.g., the time for grasping the bottle lasts for less than 2 s) or trajectories that lead to a failure due to operational reasons (e.g., the time for grasping the bottle lasts for more than 4 s).

For the advisory motion STL inference in Algorithm 1, we search for the best primitive subformula for the prefixes of the robot in three different object-centered reference frames (the bottle on the shelf, the center of the faster moving obstacle, and the center of the slower moving obstacle, which are differentiated using the subscripts s, o_1, and o_2, respectively). We use P_r, P_{o_1}, and P_{o_2} to denote the position state of the BOW robot, the faster moving obstacle, and the slower moving obstacle, respectively. We set the condition *stop* as either 90% of the prefixes at the current node belonging to the same class or $L = L_{\max} - 1 = 2$ being reached. We checked afterward that over 90% of the prefixes at each leaf (terminal) node associated with $(\top, \top)$ belong to the same class and the conditions for Proposition 3 are met (otherwise $L_{\max}$ needs to be increased for the training process). In (9), we set $\eta = 100$, and λ is tuned with different numbers (0, 4, 8). For Algorithm 2, we search for the best primitive subformula for the prefixes of the faster or slower moving obstacle in the object-centered reference frame of the bottle on the shelf at each node of the decision tree. We set $\Delta T_{\mathrm{sel}} = 0.2$ s and the condition $stop_{\mathrm{sel}}$ as all of the prefixes at the current node belonging to the same class. For testing the performance of the advisory controller designed from the inferred set of advices, we set $J(\mathbf{u}) = \|\mathbf{u}\|_2$, $\Delta N_t = 2$, and we generate 30 new trajectories from the shared autonomy between the advisory controller and a human operator. Besides the inferred advisory STL subformulas, we added two safety primitive subformulas for avoiding the two moving obstacles (specifically, avoiding the square regions with side length 1 around the centers of the two moving obstacles) throughout the entire time. The advisory controller is always kept ON until it is automatically switched OFF or the human operator has to switch it OFF to grasp the bottle. The success rates and advice cover rates in the test after the training process with $\lambda = 0$, $\lambda = 4$, and $\lambda = 8$ are listed in Table II (the advice cover rate is the fraction of the attempts when an advice is selected) and

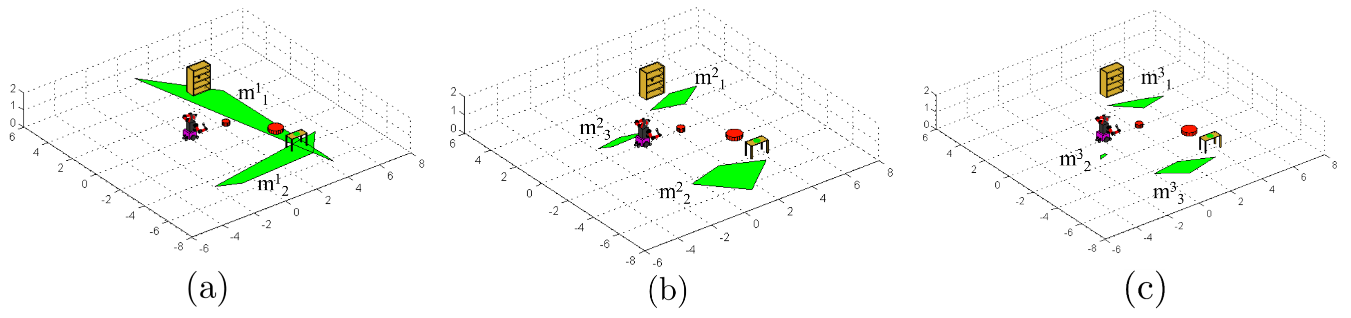


Fig. 7. Regions in the obtained advisory motion STL subformulas in case study I when (a) $\lambda = 0$, (b) $\lambda = 4$, and (c) $\lambda = 8$.

TABLE II
SUCCESS RATES AND ADVICE COVER RATES IN THE TEST AFTER THE TRAINING PROCESS WITH DIFFERENT λ IN CASE STUDY I

λ	Γ	subformulas in Φ	subformulas in Ψ	success rate	advice cover rate
0	$\gamma_1^1 = (\phi_1^1, \psi_1^1)$ $\gamma_2^1 = (\neg\phi_1^1 \wedge \phi_2^1, \neg\psi_1^1 \wedge \psi_2^1)$	$\phi_1^1 = F_{[2,8)}(P_{\mathrm{r}} \in (m_1^1)_{\mathrm{s}})$ $\phi_2^1 = F_{[4,8)}(P_{\mathrm{r}} \in (m_2^1)_{\mathrm{s}})$	$\psi_1^1 = G_{[0.1,0.2)}\neg(P_{\mathrm{o}_1} \in (s_{1,1}^1)_{\mathrm{s}})$ $\vee\ G_{[0.1,0.2)}\neg(P_{\mathrm{o}_1} \in (s_{1,2}^1)_{\mathrm{s}})$ $\psi_2^1 = \top$	46.67%	100%
4	$\gamma_1^2 = (\phi_1^2, \psi_1^2)$ $\gamma_2^2 = (\neg\phi_1^2 \wedge \phi_2^2, \neg\psi_1^2 \wedge \psi_2^2)$	$\phi_1^2 = F_{[3.59,8)}(P_{\mathrm{r}} \in (m_1^2)_{\mathrm{s}})$ $\phi_2^2 = F_{[4,8)}(P_{\mathrm{r}} \in (m_2^2)_{\mathrm{s}})$	$\psi_1^2 = G_{[0,0.13)}\neg(P_{\mathrm{o}_1} \in (s_{1,1}^2)_{\mathrm{s}})$ $\vee\ F_{[0.1,0.2)}(P_{\mathrm{o}_1} \in (s_{1,2}^2)_{\mathrm{s}})$ $\psi_2^2 = \top$	70%	100%
8	$\gamma_1^3 = (\phi_1^3, \psi_1^3)$ $\gamma_2^3 = (\neg\phi_1^3 \wedge \phi_2^3, \neg\psi_1^3 \wedge \psi_2^3)$ $\gamma_3^3 = (\neg\phi_1^3 \wedge \neg\phi_2^3 \wedge \phi_3^3,$ $\neg\psi_1^3 \wedge \neg\psi_2^3 \wedge \psi_3^3)$	$\phi_1^3 = F_{[2,7.87)}(P_{\mathrm{r}} \in (m_1^3)_{\mathrm{s}})$ $\phi_2^3 = F_{[4,8)}(P_{\mathrm{r}} \in (m_2^3)_{\mathrm{s}})$ $\phi_3^3 = F_{[2,8)}(P_{\mathrm{r}} \in (m_3^3)_{\mathrm{s}})$	$\psi_1^3 = G_{[0,0.17)}\neg(P_{\mathrm{o}_1} \in (s_1^3)_{\mathrm{s}})$ $\psi_2^3 = G_{[0.1,0.2)}\neg(P_{\mathrm{o}_1} \in (s_2^3)_{\mathrm{s}})$ $\psi_3^3 = \top$	56.67%	100%

TABLE III
SUCCESS RATES AND ADVICE COVER RATES IN THE TEST AFTER EACH DIFFERENT STAGE IN CASE STUDY I

Stage	Γ	subformulas in Φ	subformulas in Ψ	success rate	advice cover rate
refinement I	$\gamma_1^{\mathrm{I}} = (\phi_1^2, \psi_1'^2)$ $\gamma_2^{\mathrm{I}} = (\neg\phi_1^2 \wedge \phi_2^2 \wedge \phi_3^2,$ $\neg\psi_1'^2 \wedge \psi_2^2 \wedge \psi_3^2)$	$\phi_1^2 = F_{[3.59,8)}(P_{\mathrm{r}} \in (m_1^2)_{\mathrm{s}})$ $\phi_2^2 = F_{[4,8)}(P_{\mathrm{r}} \in (m_2^2)_{\mathrm{s}})$ $\phi_3^2 = F_{[9.49,16)}(P_{\mathrm{r}} \in (m_3^2)_{\mathrm{o}_1})$	$\psi_1'^2 = \big(G_{[0,0.13)}\neg(P_{\mathrm{o}_1} \in (s_{1,1}^2)_{\mathrm{s}})$ $\vee\ F_{[0.1,0.2)}(P_{\mathrm{o}_1} \in (s_{1,2}^2)_{\mathrm{s}})\big)$ $\wedge G_{[0,0.13)}\neg(P_{\mathrm{o}_2} \in (s_{1,3}^2)_{\mathrm{s}})$ $\psi_2^2 = \psi_3^2 = \top$	83.3%	100%
refinement II	$\gamma_1^{\mathrm{II}} = (\phi_1^2, \psi_1''^2)$ $\gamma_2^{\mathrm{II}} = (\neg\phi_1^2 \wedge \phi_2^2 \wedge \phi_3^2,$ $\neg\psi_1''^2 \wedge \psi_2^2 \wedge \psi_3^2)$	$\phi_1^2 = F_{[3.59,8)}(P_{\mathrm{r}} \in (m_1^2)_{\mathrm{s}})$ $\phi_2^2 = F_{[4,8)}(P_{\mathrm{r}} \in (m_2^2)_{\mathrm{s}})$ $\phi_3^2 = F_{[9.49,16)}(P_{\mathrm{r}} \in (m_3^2)_{\mathrm{o}_1})$	$\psi_1''^2 = \big(G_{[0,0.13)}\neg(P_{\mathrm{o}_1} \in (s_{1,1}^2)_{\mathrm{s}})$ $\vee\ F_{[0.1,0.2)}(P_{\mathrm{o}_1} \in (s_{1,2}^2)_{\mathrm{s}})\big)$ $\wedge G_{[0,0.13)}\neg(P_{\mathrm{o}_2} \in (s_{1,3}^2)_{\mathrm{s}})$ $\wedge G_{[0,0.13)}\neg(P_{\mathrm{o}_1} \in (s_{1,4}^2)_{\mathrm{s}})$ $\psi_2^2 = \psi_3^2 = \top$	90%	100%

shown in Figs. 7 and 8. When $\lambda = 0$, the obtained regions are overly large and the two different motions are not conservative enough, which leads to low success rate (46.67%) in the test. When $\lambda = 8$, some of the obtained regions are overly small (conservative), which also leads to low success rate (56.67%) in the test due to overfitting. We choose $\lambda = 4$ finally for the training process as it has the highest success rate (70%) in the test. We add the 30 newly generated trajectories for refinement and the refined set of advices are further used to design the advisory controller for another round of testing and refinement. We set the error bound $\varepsilon = 0.05$ in the refinement process. The obtained decision tree after the training process and two rounds of the refinement process are as shown in Fig. 9 and the set of advisory STL formulas is listed in Table III. The regions in the obtained advisory selection STL formulas are shown in Fig. 10. It can be seen that after the training process, the first advice (γ_1^2) is intuitively going straight to region m_1^2 sometime between 3.59 s and 8 s to grasp the bottle if the trajectory of the faster moving obstacle strongly satisfies ψ_1^2, the second advice (γ_2^2) is intuitively going to region m_2^2 sometime between 4 s and 8 s (thus, attracting the moving obstacles to move away from the shelf) if the trajectory of the faster moving obstacle strongly satisfies $\neg\psi_1^2$. In the first round of refinement, the advisory selection STL subformula of the slower moving obstacle is added for the first advice (γ_1^{I}), and the advisory motion STL subformula is added for coming

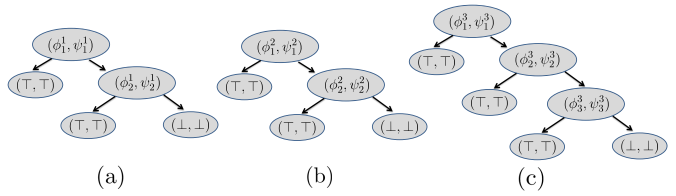


Fig. 8. Obtained decision tree in case study I when (a) $\lambda = 0$, (b) $\lambda = 4$, and (c) $\lambda = 8$.

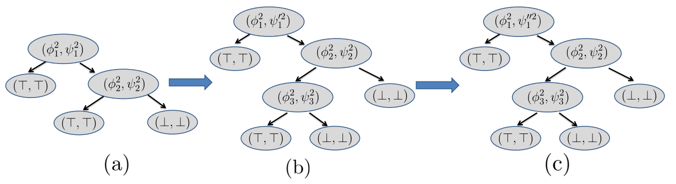


Fig. 9. Obtained decision tree in case study I after (a) training process, (b) refinement process I, and (c) refinement process II.

back to region m_3^2 sometime between 9.49 s and 16 s in the reference frame of the faster moving obstacle for the second advice (γ_2^{I}). In the second round of refinement, the advisory selection STL subformula of the faster moving obstacle is added for the first advice (γ_1^{II}). The success rate has been iteratively improved with the help of the advisory controller and reached 90% after two rounds of refinement process. Some test simulations with and without the designed advisory controller can be found in the uploaded **supplementary video**.

VI. Case Study II

In the second case study, we apply the proposed approach on a reach-avoid game that was designed with two Parrot AR drone quadrotors[3] in an experimental testbed.

The quadrotor is modeled as a 3-D six degrees of freedom rigid body. We denote the system state as $x_q = [p_q, \dot{p}_q, \theta_q, \boldsymbol{\Omega}_q]^T \in \mathbb{R}^{12}$, where $p_q = [x_1, x_2, x_3]^T$ and $\dot{p}_q = [\dot{x}_1, \dot{x}_2, \dot{x}_3]^T$ are the position and velocity vectors of the quadrotor. The vector $\theta_q = [\alpha_q, \beta_q, \gamma_q]^T \in \mathbb{R}^3$ includes the roll, pitch, and yaw Euler angles of the quadrotor. The vector $\boldsymbol{\Omega}_q \in \mathbb{R}^3$ includes the angular velocities rotating around its body frame axes. The general nonlinear dynamic model of a quadrotor is as follows:

$$\begin{aligned} m\ddot{p}_q &= R(\theta_q)T_q\mathbf{e}_3 - mg\mathbf{e}_3 \\ \dot{\theta}_q &= H(\theta_q)\boldsymbol{\Omega}_q \\ I\dot{\boldsymbol{\Omega}}_q &= -\boldsymbol{\Omega}_q \times I\boldsymbol{\Omega}_q + \tau_q \end{aligned} \tag{19}$$

where m is the mass, g is the gravitational acceleration, I is the inertia matrix, $R(\theta_q)$ is the rotation matrix representing the body frame with respect to the inertia frame (which is a function of the Euler angles), $H(\theta_q)$ is the nonlinear mapping matrix that projects the angular velocity $\boldsymbol{\Omega}_q$ to the Euler angle rate $\dot{\theta}_q$, $\mathbf{e}_3 = [0, 0, 1]^T$, T_q is the thrust of the quadrotor, and $\tau_q \in \mathbb{R}^3$ is the torque on the three axes. For the AR drone quadrotor used in this case study, only four remote control commands are used and interpreted by the low-level onboard controller of the AR drone quadrotors. We denote the control command input as $u_k = [u_1, u_2, u_3, u_4]^T$, where u_1 is the vertical velocity command, u_2, u_3 and u_4 are the angular velocity commands around its three body axes. The input values u_1, u_2, u_3, u_4 are all bounded by $[-1, 1]$. By adopting the small angle assumption and then linearizing the dynamics model around the hover state, a linear kinematics model can be obtained as follows:

$$\dot{x}_k = A_k x_k + B_k u_k \tag{20}$$

where $x_k = [x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \alpha_q, \beta_q, \gamma_q] \in \mathbb{R}^8$ is the state of the kinematics model, $A_k \in \mathbb{R}^{8\times 8}$, $B_k \in \mathbb{R}^{8\times 4}$. We assume that the low-level onboard controller is executed at a sufficiently fast speed. Based on the kinematics model, the desired remote control command vector can be obtained as $u_k = K_{\mathrm{LQR}}(x_d - x_k)$, where K_{LQR} is the state feedback gain derived from a linear quadratic regulator (LQR) and x_d is the reference state vector interpreted from the input of the user joystick or the advisory controller. The remote control command vector u_k is sent through a user datagram protocol connection between the AR drone and the desktop computer at 100 Hz.

The experiment area is shown in Fig. 11(a). For each attempt, the pursuer quadrotor always starts from (0, 0, 0.7), while the initial position of the evader quadrotor is generated randomly in the whole area except that its center should be at least 1 m away from the center of the pursuer quadrotor. The evader quadrotor succeeds if it first goes through the window [centered at (2, 2, 0.85)] from either side and then returns to any point in the yellow region, while avoiding the pursuer quadrotor throughout the whole time. The pursuer quadrotor succeeds if it intercepts the evader (the center of the pursuer quadrotor is within distance 0.5 m from the center of the evader quadrotor) before the evader quadrotor returns to the yellow regions. In this game, we only focus on the inference of the advisory STL formulas and the synthesis of the advisory controller for the pursuer quadrotor. The evader quadrotor adopts the A^* search algorithm [26] iteratively to find its way through the window and returns to the nearest point in the yellow region while avoiding the pursuer quadrotor as an obstacle. The maximal speeds of both quadrotors are set as 0.25 m/s. Two human demonstrators first generate 20 prefixes from 20 different attempts (8 successes and 12 failures) using the joystick control to form the training data set.

In Algorithm 1, the advisory motion STL subformula is searched for the pursuer quadrotor (the robot) in the fixed reference frame and the object-centered reference frame of the evading quadrotor (differentiated using the subscripts f and e, respectively) at each node of the decision tree. We use P_p and P_e to denote the position state of the pursuer quadrotor and the evader quadrotor, respectively. In (9), we set $\eta = 100$ and $\lambda = 4$. In Algorithm 2, the advisory selection STL subformula is searched for the evader quadrotor (the environment) in the fixed reference frame. The conditions *stop* and *stop*$_{\mathrm{sel}}$ in Algorithms 1 and 2 are the same as those in Case Study I, and it is checked that the conditions for Proposition 3 are met. After the training process, the obtained decision tree is shown

[3]https://www.parrot.com/us/drones/parrot-ardrone-20-elite-edition#parrot-ardrone-20-elite-edition

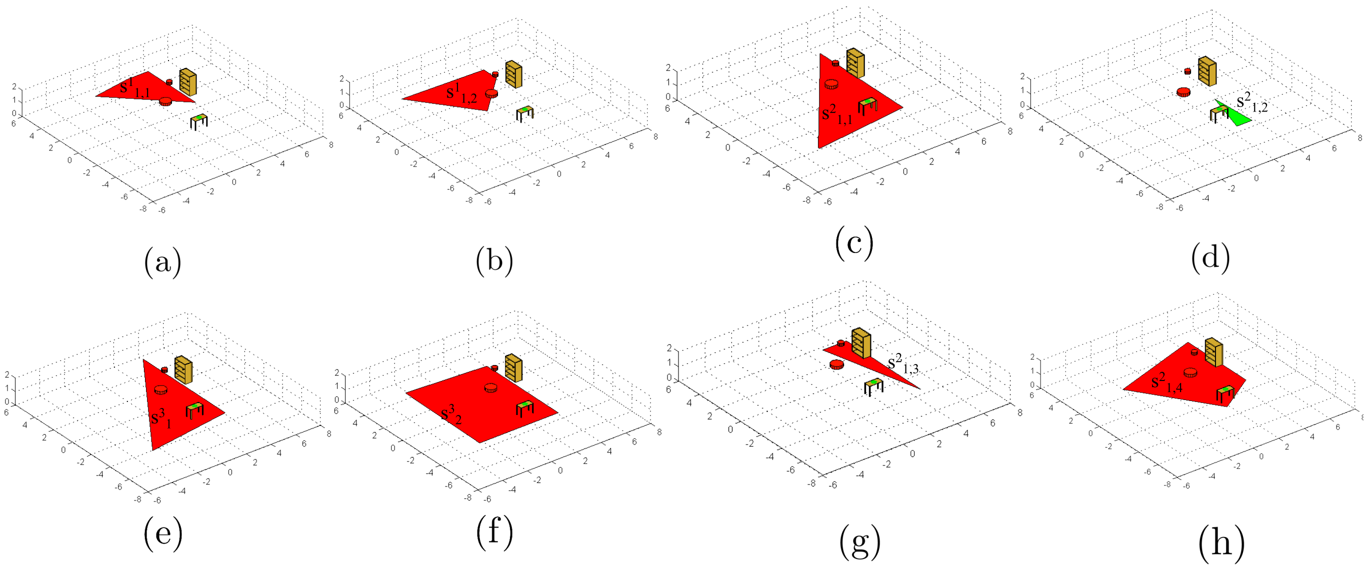


Fig. 10. Regions in the obtained advisory selection STL formulas in case study I.

TABLE IV
SUCCESS RATES AND ADVICE COVER RATES IN THE TEST AFTER EACH DIFFERENT STAGE IN CASE STUDY II

Stage	Γ	subformulas in Φ	subformulas in Ψ	success rate	advice cover rate
training	$\gamma_1^{\mathrm{t}} = (\phi_1, \psi_1)$ $\gamma_2^{\mathrm{t}} = (\neg\phi_1 \wedge \phi_2, \neg\psi_1 \wedge \psi_2)$	$\phi_1 = F_{[7,8)}(P_{\mathrm{p}} \in (m_1)_{\mathrm{f}})$ $\phi_2 = F_{[5.54,7.89)}(P_{\mathrm{p}} \in (m_2)_{\mathrm{f}})$	$\psi_1 = G_{[0.02,0.12)}\neg(P_{\mathrm{e}} \in (s_1)_{\mathrm{f}})$ $\psi_2 = \top$	80%	100%
refinement	$\gamma_1^{\mathrm{r}} = (\phi_1, \psi_1')$ $\gamma_2^{\mathrm{r}} = (\neg\phi_1 \wedge \phi_2, \neg\psi_1' \wedge \psi_2)$	$\phi_1 = F_{[7,8)}(P_{\mathrm{p}} \in (m_1)_{\mathrm{f}})$ $\phi_2 = F_{[5.54,7.89)}(P_{\mathrm{p}} \in (m_2)_{\mathrm{f}})$	$\psi_1' = G_{[0.02,0.12)}\neg(P_{\mathrm{e}} \in (s_1)_{\mathrm{f}}) \wedge G_{[0,0.14)}\neg(P_{\mathrm{e}} \in (s_2)_{\mathrm{f}})$ $\psi_2 = \top$	95%	100%

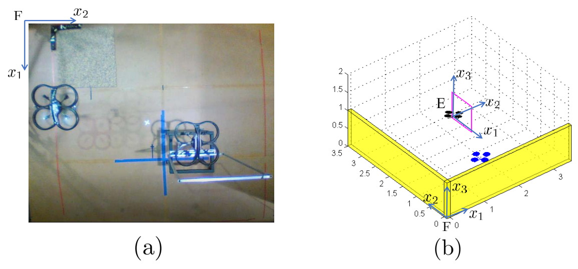


Fig. 11. (a) Experimental testbed area. (b) Reference frames in the visualization interface.

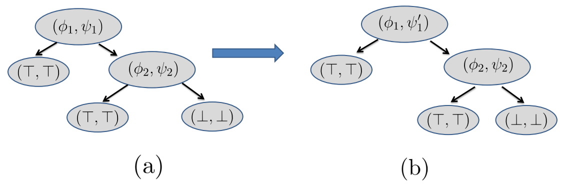


Fig. 12. Obtained decision tree in case study II after (a) training process and (b) refinement process.

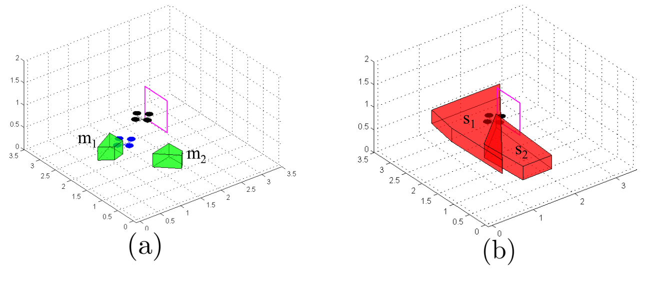


Fig. 13. Regions in the obtained advisory (a) motion and (b) selection STL formulas in case study II.

in Fig. 12(a) and the set of advices is listed in Table IV, with the regions shown in Fig. 13. Using the designed advisory controller from the inferred set of advices, we perform one round of the refinement process with 20 newly generated trajectories (we set $\varepsilon = 0.05$) and the obtained decision tree is shown in Fig. 12(b) and the set of advices is listed in Table IV, with the regions shown in Fig. 13. The success rate has been iteratively improved with the help of the advisory controller and reached 95% after one round of the refinement process. Some test simulations with and without the designed advisory controller can be found in the uploaded **supplementary video**.

VII. CONCLUSION

In this paper, we presented a method for learning a set of advices from the successful and failed experiences in the form of advisory STL formulas. The advices specified the successful feature of a task and it is learned (inferred) from human demonstrations. The same methodology of inferring a temporal logic formula from the trajectories of both desired and undesired sets as features of success, and designing a mechanism based on the inferred temporal logic formula to guide or advise human operators for better practice can be used in much broader applications.

Appendix

A. Proof of Theorem 1

If: without loss of generality, we assume that $H_1 \geq H_2$. If $\sup_{0\leq t\leq \min(H_1,H_2)} \|\hat{x}^1_{H_1}(t) - \hat{x}^2_{H_2}(t)\| > 0$, then there exists at least one time point $t^* \in [0, H_2]$ and one dimension i such that $\hat{x}^1_{H_1,i}(t^*) \neq \hat{x}^2_{H_2,i}(t^*)$ (we denote the i-th dimension of the state $\hat{x}_H(t)$ as $\hat{x}_{H,i}(t)$). Without loss of generality, we assume that $\hat{x}^1_{H_1,i}(t^*) > \hat{x}^2_{H_2,i}(t^*)$, so there exists $\epsilon > 0$ such that $\hat{x}^1_{H_1,i}(t^*) - \epsilon \geq \hat{x}^2_{H_2,i}(t^*)$. As the prefix $\hat{x}^1_{H_1}$ is continuous in the time interval $[0, H_2]$, then for any $\epsilon > 0$, however small, there exists some number $\delta > 0$ such that for all $t \in [\min(0, t^* - \delta), \max(H_2, t^* + \delta)]$, $\hat{x}^1_{H_1,i}(t^*) - \epsilon < \hat{x}^1_{H_1,i}(t) < \hat{x}^1_{H_1,i}(t^*) + \epsilon$. So $\hat{x}^1_{H_1}$ strongly satisfies the STL formula $\phi = G_{[\min(0,t^*-\delta),\max(H_2,t^*+\delta))}(x_i > \hat{x}^1_{H_1,i}(t^*) - \epsilon)$ while $\hat{x}^2_{H_2}$ strongly violates ϕ as $\hat{x}^2_{H_2,i}(t^*) \leq \hat{x}^1_{H_1,i}(t^*) - \epsilon$. So the formula ϕ can perfectly classify $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$. Similarly, it can be shown that there exists $\delta' > 0$ such that $\phi' = F_{[\min(0,t^*-\delta'),\max(H_2,t^*+\delta'))}(x_i \geq \hat{x}^2_{H_2,i}(t^*) + \epsilon)$ perfectly classifies $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$.

Only if: according to Definitions 3 and 4, $(\hat{x}_H, t) \models_S \pi$ if and only if $t \leq H$ and $f(\hat{x}_H(t)) > 0$; $(\hat{x}_H, t) \models_S \neg\pi$ if and only if $(\hat{x}_H, t) \not\models_W \pi$, i.e., $t \leq H$ and $f(\hat{x}_H(t)) \leq 0$. Therefore, if there exists an STL formula that can perfectly classify $\hat{x}^1_{H_1}$ and $\hat{x}^2_{H_2}$, then there exists time $t^* \in [0, \min(H_1, H_2)]$ such that $f(\hat{x}_{H_1}(t^*)) > 0$ and $f(\hat{x}_{H_2}(t^*)) \leq 0$. So $\sup_{0\leq t\leq \min(H_1,H_2)} \|\hat{x}^1_{H_1}(t) - \hat{x}^2_{H_2}(t)\| \geq \|\hat{x}^1_{H_1}(t^*) - \hat{x}^2_{H_2}(t^*)\| > 0$.

B. Proof of Theorem 2

If: according to Theorem 1, for any prefix $\hat{x}^i_{H_i} \in \mathcal{A}$ and any prefix $\hat{x}'^j_{H'_j} \in \mathcal{B}$, if $\sup_{0\leq t\leq \min(H_i,H'_j)} \|\hat{x}^i_{H_i}(t) - \hat{x}'^j_{H'_j}(t)\| > 0$, then we can always find ϕ_{ij} such that ϕ_{ij} is strongly satisfied by $\hat{x}^i_{H_i}$ and strongly violated by $\hat{x}'^j_{H'_j}$. Therefore, the STL formula $\phi = (\phi_{11} \wedge \phi_{12} \wedge \cdots \wedge \phi_{1N_B}) \vee (\phi_{21} \wedge \phi_{22} \wedge \cdots \wedge \phi_{2N_B}) \vee \ldots (\phi_{N_A1} \wedge \phi_{N_A2} \wedge \cdots \wedge \phi_{N_AN_B})$ can perfectly classify the two sets of prefixes, because ϕ is strongly satisfied by $\hat{x}^1_{H_1}, \ldots, \hat{x}^{N_A}_{H_{N_A}}$ and strongly violated by $\hat{x}'^1_{H'_1}, \ldots, \hat{x}'^{N_B}_{H'_{N_B}}$.

Only if: if there exists an STL formula that can perfectly classify any prefix $\hat{x}^i_{H_i} \in \mathcal{A}$ and any prefix $\hat{x}'^j_{H'_j} \in \mathcal{B}$, then according to Theorem 1, $\sup_{0\leq t\leq \min(H_i,H'_j)} \|\hat{x}^i_{H_i}(t) - \hat{x}'^j_{H'_j}(t)\| > 0$.

C. Proof of Proposition 1

According to Definition 5, to prove Proposition 1, we only need to prove the following lemma.

Lemma 1: With $\tau(\phi, l_i)$ defined as in Proposition 1, for any (F, G)-fragment STL formula ϕ, if $H_i < t + \tau(\phi, l_i)$, then if $l_i = 1$, $(\hat{x}^i_{H_i}, t) \models_W \neg\phi$; if $l_i = -1$, $(\hat{x}^i_{H_i}, t) \models_W \phi$.

We use induction to prove Lemma 1.

1) We first prove that Lemma 1 holds for atomic predicate π. As $\tau(\pi, l_i) = 0$, if $H_i < t + \tau(\phi, l_i)$, then $H_i < t$, so according to Definition 4, for $l_i = 1$ and $l_i = -1$, $(\hat{x}^i_{H_i}, t) \models_W \pi$ and $(\hat{x}^i_{H_i}, t) \models_W \neg\pi$, Lemma 1 trivially holds.
2) We assume that Lemma 1 holds for ϕ and prove Lemma 1 holds for $\neg\phi$. If Lemma 1 holds for ϕ, then if $H_i < t + \tau(\phi, l_i)$, we have for $l_i = 1$, $(\hat{x}^i_{H_i}, t) \models_W \neg\phi$; for $l_i = -1$, $(\hat{x}^i_{H_i}, t) \models_W \phi$. Thus, if $H_i < t + \tau(\phi, -l_i)$, we have for $l_i = -1$, $(\hat{x}^i_{H_i}, t) \models_W \neg\phi$; for $l_i = 1$, $(\hat{x}^i_{H_i}, t) \models_W \phi$. Therefore, if $H_i < t + \tau(\neg\phi, l_i) = t + \tau(\phi, -l_i)$, we have for $l_i = 1$, $(\hat{x}^i_{H_i}, t) \models_W \phi$, i.e., $(\hat{x}^i_{H_i}, t) \models_W \neg(\neg\phi)$; for $l_i = -1$, $(\hat{x}^i_{H_i}, t) \models_W \neg\phi$. Therefore, Lemma 1 holds for $\neg\phi$.
3) We assume that Lemma 1 holds for ϕ_1, ϕ_2 and prove Lemma 1 holds for $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$.
   For $l_i = 1$, if Lemma 1 holds for ϕ_1 and ϕ_2, then if $H_i < t + \tau(\phi_1, l_i)$, we have $(\hat{x}^i_{H_i}, t) \models_W \neg\phi_1$; if $H_i < t + \tau(\phi_2, l_i)$, we have $(\hat{x}^i_{H_i}, t) \models_W \neg\phi_2$. If $H_i < t + \max\{\tau(\phi_1, l_i), \tau(\phi_2, l_i)\}$, then $H_i < t+\tau(\phi_1, l_i)$ or $H_i < t+\tau(\phi_2, l_i)$, thus $(\hat{x}^i_{H_i}, t) \models_W \neg\phi_1$ or $(\hat{x}^i_{H_i}, t) \models_W \neg\phi_2$. So we have $(\hat{x}^i_{H_i}, t) \models_W \neg\phi_1 \vee \neg\phi_2$, i.e., $(\hat{x}^i_{H_i}, t) \models_W \neg(\phi_1 \wedge \phi_2)$.
   For $l_i = -1$, if Lemma 1 holds for ϕ_1 and ϕ_2, then if $H_i < t + \tau(\phi_1, l_i)$, we have $(\hat{x}^i_{H_i}, t) \models_W \phi_1$; if $H_i < t + \tau(\phi_2, l_i)$, we have $(\hat{x}^i_{H_i}, t) \models_W \phi_2$. If $H_i < t + \min\{\tau(\phi_1, l_i), \tau(\phi_2, l_i)\}$, then $H_i < t + \tau(\phi_1, l_i)$ and $H_i < t+\tau(\phi_2, l_i)$, thus $(\hat{x}^i_{H_i}, t) \models_W \phi_1$ and $(\hat{x}^i_{H_i}, t) \models_W \phi_2$. Therefore, we have $(\hat{x}^i_{H_i}, t) \models_W \phi_1 \wedge \phi_2$. Therefore, Lemma 1 holds for $\phi_1 \wedge \phi_2$.
   Similarly, it can be proved by induction that Lemma 1 holds for $\phi_1 \vee \phi_2$.
4) We assume that Lemma 1 holds for ϕ and prove Lemma 1 holds for $F_{[t_1,t_2)}\phi$ and $G_{[t_1,t_2)}\phi$.
   For $l_i = 1$, if $H_i < t + \tau(\phi, l_i) + t_1$, then for any $t' \in [t + t_1, t + t_2)$, we have $H_i < \tau(\phi, l_i) + t'$, so if Lemma 1 holds for ϕ, then for any $t' \in [t + t_1, t + t_2)$, we have $(\hat{x}^i_{H_i}, t') \models_W \neg\phi$, thus $(\hat{x}^i_{H_i}, t) \models_W G_{[t_1,t_2)}\neg\phi$, i.e., $(\hat{x}^i_{H_i}, t) \models_W \neg F_{[t_1,t_2)}\phi$.
   For $l_i = -1$, if $H_i < t + \tau(\phi, l_i) + t_2$, then there exists $t' \in [t + t_1, t + t_2)$ such that $H_i < \tau(\phi, l_i) + t'$, so if Lemma 1 holds for ϕ, we have $(\hat{x}^i_{H_i}, t') \models_W \phi$, thus $(\hat{x}^i_{H_i}, t) \models_W F_{[t_1,t_2)}\phi$. Therefore, Lemma 1 holds for $F_{[t_1,t_2)}\phi$.

Similarly, it can be proved by induction that Lemma 1 holds for $G_{[t_1,t_2)}\phi$.

Therefore, it is proven by induction that Lemma 1 holds for any (F,G)-fragment STL formula ϕ.

D. Proof of Proposition 2

1) *Coverage:* From Algorithm 3, we express each advisory motion STL formula as
$$\phi_j = \hat{\phi}_{j,1} \wedge \hat{\phi}_{j,2} \wedge \cdots \wedge \hat{\phi}_{j,q_j} \tag{21}$$

with the corresponding advisory selection STL formula as

$$\psi_j = \hat{\psi}_{j,1} \wedge \hat{\psi}_{j,2} \wedge \cdots \wedge \hat{\psi}_{j,q_j} \tag{22}$$

where each $(\hat{\phi}_{j,k}, \hat{\psi}_{j,k})$ pair is either a pair of primitive STL subformulas $(\phi_{j,k}, \psi_{j,k})$ or a pair of negation of primitive STL subformulas $(\neg\phi_{j,k}, \neg\psi_{j,k})$ at one node of the tree. From Algorithm 1, for every i, if $l_i = 1$, we can prove that if condition (2) and condition (3) are satisfied, then for the pair $(\phi_{j,k}, \psi_{j,k})$ associated with any node in the path from the root node to the leaf (terminal) node where the ith prefix belongs, either $((\hat{x}^i_{H_i}, 0) \models_S \phi_{j,k}) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_{j,k})$ or $((\hat{x}^i_{H_i}, 0) \models_S \neg\phi_{j,k}) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \neg\psi_{j,k})$. The reasons are as follows: (i) in Algorithm 1, if $\hat{p}(\hat{S}_{x,\top}, -1) = 0$ or $J_y(\hat{S}_{x,\top}, \psi^{\text{path}} \wedge \psi^*) > 0$, then $\psi_{j,k} = \psi^*_s$ is obtained from Algorithm 2 and perfectly classifies the prefixes of the environment with label 1 in $\hat{S}_{x,\top}$ and $\hat{S}_{x,\perp}$, respectively (as we assume that $stop_{\text{sel}}(\psi^{\text{path}}_s, h, \hat{S}')$ is achieved as all of the prefixes at the current node belong to the same class). Therefore, if $\hat{S}_{x,\emptyset} = \emptyset$ and $l_i = 1$, then either $\hat{x}^i_{H_i} \in \hat{S}_{x,\top}$, $((\hat{x}^i_{H_i}, 0) \models_S \phi_{j,k}) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_{j,k})$, or $\hat{x}^i_{H_i} \in \hat{S}_{x,\perp}$, $((\hat{x}^i_{H_i}, 0) \models_S \neg\phi_{j,k}) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \neg\psi_{j,k})$. (ii) in Algorithm 1, if $\hat{p}(\hat{S}_{x,\top}, -1) > 0$ and $J_y(\hat{S}_{x,\top}, \psi^{\text{path}} \wedge \psi^*) = 0$, then $\psi_{j,k} = \psi^* \wedge \psi^*_s$, where ψ^* perfectly classifies the prefixes of the environment with labels 1 and -1 in $\hat{S}_{x,\top}$, ψ^*_s is obtained from Algorithm 2 and perfectly classifies the prefixes of the environment with label 1 in $\hat{S}_{x,\top}$ and $\hat{S}_{x,\perp}$. Therefore, if $\hat{S}_{x,\emptyset} = \emptyset$ and $l_i = 1$, then either $\hat{x}^i_{H_i} \in \hat{S}^r_{x,\top}$ or $\hat{x}^i_{H_i} \in \hat{S}_{x,\perp}$. If $\hat{x}^i_{H_i} \in \hat{S}^r_{x,\top}$, then $(\hat{x}^i_{H_i}, 0) \models_S \phi_{j,k}$, $(\hat{y}^i_{H_i}, 0) \models_S \psi^*$, $(\hat{y}^i_{H_i}, 0) \models_S \psi^*_s$, thus as $\psi_{j,k} = \psi^* \wedge \psi^*_s$, we have $((\hat{x}^i_{H_i}, 0) \models_S \phi_{j,k}) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_{j,k})$. If $\hat{x}^i_{H_i} \in \hat{S}_{x,\perp}$, then $(\hat{x}^i_{H_i}, 0) \models_S \neg\phi_{j,k}$, $(\hat{y}^i_{H_i}, 0) \models_S \neg\psi^*_s$, thus as $\neg\psi_{j,k} = \neg\psi^* \vee \neg\psi^*_s$, we have $((\hat{x}^i_{H_i}, 0) \models_S \neg\phi_{j,k}) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \neg\psi_{j,k})$. With (i), (ii), and Algorithm 3, if $l_i = 1$, then there exists an advice (ϕ_j, ψ_j) in the form of (21) and (22) such that $((\hat{x}^i_{H_i}, 0) \models_S \phi_j) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_j)$.

2) *Soundness:* From Algorithm 1, each leaf node of the decision tree is associated with $(\top, \top)$ or $(\perp, \perp)$. From Algorithm 3, each advisory motion (selection) logic formula in each advice is constructed from the advisory motion (selection) logic subformulas or the negation of them connected with conjunction operators, and $(\perp, \perp)$ is removed from the set of advices. Therefore, each advice (ϕ_j, ψ_j) corresponds to a leaf node associated with $(\top, \top)$. As $stop(\phi^{\text{path}}, L, \hat{S})$ is achieved as all of the prefixes at the current node belong to the same class, so all the prefixes that belong to a leaf node associated with $(\top, \top)$ should have label 1. Therefore, if there exists an advice (ϕ_j, ψ_j) such that $((\hat{x}^i_{H_i}, 0) \models_S \phi_j) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_j)$, then $l_i = 1$.

3) *Uniqueness:* At each node that is associated with $(\phi_{j,k}, \psi_{j,k})$, if $\hat{S}'_{y,\emptyset} = \emptyset$ in each computation for Algorithm 2, then for any i such that $l_i = 1$, either $(\hat{y}^i_{H_i}, 0) \models_S \psi_{j,k}$ or $(\hat{y}^i_{H_i}, 0) \models_S \neg\psi_{j,k}$. Therefore, following Algorithm 3, for advisory selection STL formulas in the form of (22), there cannot exist j and j' $(j \neq j')$ such that $(\hat{y}^i_{H_i}, 0) \models_S \psi_j$ and $(\hat{y}^i_{H_i}, 0) \models_S \psi_{j'}$.

E. Proof of Proposition 3

The proof for satisfying the coverage property 1) and uniqueness property 3) of Problem 1 is the same as that in the proof of Proposition 2. For satisfying the soundness property 2) of Problem 1, as $stop(\phi^{\text{path}}, L, \hat{S})$ is achieved as no less than $(1-\zeta)$ fraction of the prefixes at the current node belong to the same class, so no less than $(1-\zeta)$ fraction of the prefixes that belong to each leaf node associated with $(\top, \top)$ should have label 1. Therefore, for no less than $(1-\zeta)$ fraction of the prefixes in the training data set, if there exists an advice (ϕ_j, ψ_j) such that $((\hat{x}^i_{H_i}, 0) \models_S \phi_j) \wedge ((\hat{y}^i_{H_i}, 0) \models_S \psi_j)$ in the training data set, then $l_i = 1$.

ACKNOWLEDGMENT

The authors would like to thank Prof. J. Wen and D. Kruse for introducing them to the simulating platform of BOW. The authors would also like to thank Prof. C. Belta and G. Bombara for introducing them to the decision tree approach for STL formula inference.

REFERENCES

[1] K.-T. Song, S.-Y. Jiang, and M.-H. Lin, "Interactive teleoperation of a mobile manipulator using a shared-control approach," *IEEE Trans. Human-Mach. Syst.*, vol. 46, no. 6, pp. 834–845, Dec. 2016.

[2] J. Jiang, P. Di Franco, and A. Astolfi, "Shared control for the kinematic and dynamic models of a mobile robot," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 6, pp. 2112–2124, Nov. 2016.

[3] J. Smisek, E. Sunil, M. M. van Paassen, D. A. Abbink, and M. Mulder, "Neuromuscular-system-based tuning of a haptic shared control interface for uav teleoperation," *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 4, pp. 449–461, Aug. 2016.

[4] X. Li, C. I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3834–3839.

[5] M. Ewerton, G. Neumann, R. Lioutikov, H. B. Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1535–1542.

[6] Z. Kong, A. Jones, and C. Belta, "Temporal logics for learning and detection of anomalous behavior," *IEEE Trans. Autom. Control*, vol. 62, no. 3, pp. 1210–1222, Mar. 2017.

[7] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A decision tree approach to data classification using signal temporal logic," in *Proc. 19th Int. Conf. Hybrid Syst., Comput. Control (HSCC)*, New York, NY, USA: ACM, 2016, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/2883817.2883843

[8] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Proc. 2nd Int. Conf. Runtime Verification*, Berlin, Germany, 2012, pp. 147–160.

[9] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *Proc. 24th Int. Conf. Test. Softw. Syst.*, Aalborg, Denmark, 2012, pp. 136–151.

[10] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," in *Proc. Int. Conf. Hybrid Syst., Comput. Control*, 2013, pp. 43–52.

[11] Z. Xu, C. Belta, and A. Julius, "Temporal logic inference with prior information: An application to robot arm movements," in *Proc. IFAC Conf. Anal. Design Hybrid Syst.*, 2015, pp. 141–146.

[12] Z. Xu and A. A. Julius, "Census signal temporal logic inference for multiagent group behavior analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 264–277, Jan. 2018.

[13] Z. Xu, M. Birtwistle, C. Belta, and A. Julius, "A temporal logic inference approach for model discrimination," *IEEE Life Sci. Lett.*, vol. 2, no. 3, pp. 19–22, Sep. 2016.

[14] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.

[15] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. 8th Int. Conf. Formal Modeling Anal. Timed Syst.*, Berlin, Germany, 2010, pp. 92–106.

[16] C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. Van Campenhout, *Reasoning With Temporal Logic on Truncated Paths*. Berlin, Germany: Springer, 2003, pp. 27–39.

[17] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods Syst. Des.*, vol. 19, no. 3, pp. 291–314, Oct. 2001.

[18] H.-M. Ho, J. Ouaknine, and J. Worrell, *Online Monitoring of Metric Temporal Logic*. Cham, Switzerland: Springer, 2014, pp. 178–192.

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

[20] A. Donzé and V. Raman, "BluSTL: Controller synthesis from signal temporal logic specifications," in *Proc. 1st, 2nd Int. Workshop Appl. Verification Continuous Hybrid Syst. (ARCH)*, in EPiC Series in Computing, vol. 34, G. Frehse and M. Althoff, Eds. EasyChair, 2015, pp. 160–168.

[21] S. Saha and A. A. Julius, "An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proc. IEEE Amer. Control Conf.*, Jul. 2016, pp. 1105–1110.

[22] J. Křetínský and J. Esparza, *Deterministic Automata for the (F,G)-Fragment of LTL*. Berlin, Germany: Springer, 2012, pp. 7–22. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31424-7_7

[23] O. Maler and D. Nickovic, *Monitoring Temporal Properties of Continuous Signals*. Berlin, Germany: Springer, 2004, pp. 152–166. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30206-3_12

[24] L. Lu and J. T. Wen, "Human-directed coordinated control of an assistive mobile manipulator," *Int. J. Intell. Robot. Appl.*, vol. 1, no. 1, pp. 104–120, 2017.

[25] A. Cunningham *et al.*, "Jamster: A mobile dual-arm assistive robot with jamboxx control," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2014, pp. 509–514.

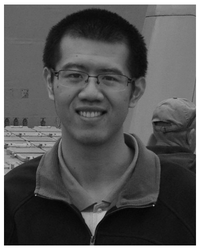
[26] W. Zeng and R. L. Church. (Apr. 2009). *Finding Shortest Paths on Real Road Networks: The Case for A**. [Online]. Available: https://doi.org/10.1080/13658810801949850

Zhe Xu (S'16) received the B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, China, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with Rensselaer Polytechnic Institute, Troy, NY, USA.

His research interests include temporal logic, systems and control, hybrid systems, and power systems.

Sayan Saha (S'13) received the B.E. degree in instrumentation and electronics from Jadavpur University, Kolkata, India, in 2012, and the M.S. degree in applied mathematics from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2016, where he is currently pursuing the Ph.D. degree in electrical engineering.

His research interests include robotics, control theory, temporal logic, and optimization algorithms.



Botao Hu (S'13) received the B.S. degree in industrial engineering and the M.S. degree in mechantronics engineering from the School of Mechantronics Engineering, Harbin Institute of Technology, Harbin, Heilongjiang, China, in 2011 and 2013, respectively. He is currently pursuing the Ph.D. degree in mechanical engineering with the Mechanical, Aerospace, and Nuclear Engineering Department, Rensselaer Polytechnic Institute, Troy, NY, USA.

His research interests include control, robotics, and UAV design.

Sandipan Mishra (M'05) received the B.Tech. degree in mechanical engineering from IIT Madras, Chennai, India, in 2002, and the Ph.D. degree in mechanical engineering from the University of California, Berkeley, Berkeley, CA, USA, in 2008.

In 2010, he was a Faculty Member with the Rensselaer Polytechnic Institute, Troy, NY, USA, where he is currently an Associate Professor with the Department of Mechanical, Aerospace, and Nuclear Engineering. His current research interests include general area of systems and control theory, iterative learning control, optimal control, and precision mechatronics, as applied to autonomous aerial vehicles, additive manufacturing, and smart building systems.

A. Agung Julius (M'06) received the Ph.D. degree in applied mathematics from the University of Twente, Enschede, The Netherlands, in 2005.

From 2005 to 2008, he was a Post-Doctoral Researcher with the University of Pennsylvania, Philadelphia, PA, USA. Since 2008, he has been with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA, where he is currently an Associate Professor. His research interests include systems and control, systems biology, stochastic models in systems biology, control of biological systems, hybrid systems, and mathematical systems theory.