Learning Minimally-Violating Continuous Control for Infeasible Linear Temporal Logic Specifications

Mingyu Cai¹, Makai Mann², Zachary Serlin², Kevin Leahy², Cristian-Ioan Vasile¹

Abstract—This paper explores continuous-time control synthesis for target-driven navigation to satisfy complex high-level tasks expressed in linear temporal logic (LTL). We propose a model-free framework using deep reinforcement learning (DRL) where the underlying dynamical system is unknown (an opaque box). Unlike prior work, we consider scenarios where the given LTL specification might be infeasible and therefore cannot be accomplished globally. Instead of modifying the given LTL formula, we provide a general DRL-based approach to satisfy it with minimal violation. To do this, we transform a previously multi-objective DRL problem, which requires simultaneous automata satisfaction and minimum violation cost, into a single objective. By guiding the DRL agent with a sampling-based path planning algorithm for the potentially infeasible LTL task, the proposed approach mitigates the myopic tendencies of DRL, which are often an issue when learning general LTL tasks that can have long or infinite horizons. This is achieved by decomposing an infeasible LTL formula into several reach-avoid sub-tasks with shorter horizons, which can be learned in a modular DRL architecture. Furthermore, we overcome the challenge of the exploration process for DRL in cluttered environments by using path planners to design rewards that are dense in the configuration space. The benefits of the presented approach are demonstrated through testing on various complex nonlinear systems and compared with state-of-the-art baselines. The video demonstration can be found here: https://youtu.be/DqesqBsjajk.

I. INTRODUCTION

Autonomous agents operating in complex environments must often accomplish high-level tasks while subject to various uncertainties. Possible uncertainties include dense obstacles or targets, and impassable or unexpected terrain. Additionally, a principled analytical model of the robot is typically (partially) unknown or hard to obtain. Motivated by these challenges, this work investigates model-free control of an autonomous system for satisfying a potentially infeasible high-level task with minimal violation while operating in a complex cluttered environment.

In the field of model-free navigation control, Reinforcement Learning (RL) is a popular unsupervised technique that optimizes long-term expected rewards to learn desired behaviors [1]. Recently, Deep Reinforcement Learning (DRL) techniques, such as the actor-critic method [2], have been shown to be capable of learning continuous controllers for unknown (opaque-box) dynamics. To investigate complex navigation tasks, Linear Temporal Logic (LTL) is a formal language that has gained attention in recent years for expressing high-level, interpretable robotics tasks. Specifications have been shown to be effective at directing RL agents to learn desired policies. Typically, a discrete robot system is abstracted as a discrete Markov Decision Process (MDP) model and composed with an automaton representing the desired LTL formula to create a product automaton for learning or planning [3]–[5]. For general continuous control subject to LTL satisfaction over infinite horizons, prior works [6], [7] proposed a modular architecture by decomposing the task into sub-tasks using automata states. Crucially, to our knowledge, no previous work has investigated infeasible LTL tasks for continuous control. In this work, we focus on the problem jointly including two aspects: infeasible LTL tasks and cluttered environments using DRL for model-free continuous control.

Related works: Minimum violation of LTL specifications over finite horizons has been considered in [8]–[13] using sampling-based methods, graph-based optimization algorithms, and dynamic programming to solve the corresponding motion planning problem. For LTL specifications over infinite horizons, general violation measurements are proposed in [14] by quantitatively revising LTL automata. The approaches are applied in [15], [16], where they build product structures and encode potentially conflicting parts of a specification as a soft constraint. However, the prior literature generally assumes known dynamical systems with abstracted navigation controllers and only focuses on motion planning problems [8]–[11], [13], [15]. Our work improves on this by directly synthesizing a low-level controller for unknown dynamics in a model-free manner that minimally violate LTL satisfaction over both finite and infinite horizons.

Previous work [17], [18] has tackled RL while considering infeasible tasks in discrete state-action spaces. In particular, authors in [17] account for the infeasible solution case by visiting the automaton accepting sets as much as possible (if at all). But many LTL formulas only have one accepting state and this strategy often fails to generalize to those cases. The learning objectives in [18] ensure satisfaction of the automata acceptance condition and, thus, LTL formulae, and reduce the violation cost, resulting in a multi-objective RL (MORL) problem. However, providing minimum-violation performance guarantees for MORL continues to be an open problem [19].

Commonly in the literature for navigation in cluttered environments, motion planning algorithms are first used to find possible paths, and then path-tracking controllers are employed to follow that path [20]–[22]. This approach decomposes into a goal-reaching control problem and can

Authorized licensed use limited to: IEEE Xplore. Downloaded on July 22, 2023 at 14:46:32 UTC from IEEE Xplore. Restrictions apply.
be easily adapted for LTL satisfaction [23], [24]. Classic approaches however require known dynamic models and may be computationally expensive for nonlinear and high-dimensional systems. This fundamentally motivates our focus on employing model-free control techniques.

**Contributions:** In this paper, we translate the MORL problem first analyzed in [18] into a standard DRL problem and apply model-free geometric motion planning to guide the learning process. The generated trajectories are shown to minimally violate a given infeasible LTL specification. By learning a controller to reach waypoints in a geometric path, our approach automatically obtains an optimal control policy for minimally-violating the infeasible LTL task specification. This paper’s main contributions are: (i) To the best of our knowledge, this is the first model-free learning-based continuous control for potentially infeasible LTL satisfaction with minimum-violation objectives. Such objectives facilitate meaningful action for infeasible LTL tasks, and the framework also generalizes discrete systems; (ii) Our framework can learn continuous control in cluttered environments and mitigates the myopic tendencies of DRL by decomposing the global task into sub-tasks, which can be trained compositionally; (iii) We demonstrate the benefits by comparing with several baselines in two nonlinear systems: a Dubins car and quadrotor.

**II. PRELIMINARIES**

**Agent:** We consider the general evolution of a continuous-time dynamical system $\mathcal{S}$ that is assumed to be unknown, starting from an initial state $s_0 \in S_0$ is given by

$$s = f(s,a),$$  \hspace{1cm} (1)

where $s \in S \subseteq \mathbb{R}^n$ is the state vector in the compact set $S$, $a \in A \subseteq \mathbb{R}^m$ is the control input. We assume the flow field $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is uniformly continuous in time and Lipschitz continuous in $s$ for fixed $a$. Under these assumptions there exists a unique solution of $\mathcal{S}$ for a given $a$, providing trajectories of the system [25].

**Definition 1.** For a robot operating in an environment $Env$, the geometric workspace can be represented by a compact subset $X \subseteq \mathbb{R}^d$, $d \in \{2,3\}$. The relation between dynamics $\mathcal{S}$ and workspace $X$ is defined by the projection $Proj : S \rightarrow X$.

The space $X$ contains regions of interest that are labeled by a set of atomic propositions $AP$, with the labeling function $L_X : X \rightarrow 2^{AP}$. Let $L : S \rightarrow 2^{AP}$ be a labeling function over $\mathcal{S}$ i.e., $L(s) = L_X(Proj(s))$.

**Reinforcement Learning:** The interactions between a robot with dynamics $\mathcal{S}$ and an environment $Env$ can be captured as a continuous-labeled Markov Decision Process (cl-MDP). A cl-MDP is a tuple $\mathcal{M} = (S, S_0, A, ps, R, \gamma, L)$, where $S \subseteq \mathbb{R}^n$ is a continuous state space, $S_0$ is a set of initial states, $A \subseteq \mathbb{R}^m$ is a continuous action space, and $ps$ captures the unknown system dynamics as a distribution. The distribution $ps : \mathcal{B}(\mathbb{R}^n) \times A \times S \rightarrow [0,1]$ is a Borel-measurable conditional transition kernel, s.t. $ps(\cdot | s, a)$ is a probability measure of the next state given current $s \in S$ and $a \in A$ over the Borel space $\mathcal{B}(\mathbb{R}^n)$, where $\mathcal{B}(\mathbb{R}^n)$ is the set of all Borel sets on $\mathbb{R}^n$. $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0,1)$ is the discount factor. $L$ is the labeling function in Def. 1.

Since the dynamics $\mathcal{S}$ are an opaque-box, the transition relation $ps(\cdot | s, a)$ for any state-action pair is unknown.

Actor-critic RL algorithms [2] have been demonstrated as promising tools to solve continuous-control problems for the cl-MDP model, where each valid transition of cl-MDP follows $f$ defined in (1) that is zero-order hold model for a continuous-time action.

**Remark 1.** Note it’s intractable to explicitly construct the cl-MDP model with continuous state and action space to apply standard model-checking algorithm [26]. Inspired by prior work [6], we track it on-the-fly via a deep neural network.

Let $\Pi$ denote the set of all policies for a cl-MDP, and $\pi \in \Pi$ denote a stochastic policy $\pi : S \times A \rightarrow [0,1]$, that maps states to distributions over actions. The objective is to learn an optimal policy $\pi^*(a|s)$ that maximizes the expected discounted return $J(\pi) = \mathbb{E}\pi\left[\sum_{k=0}^{\infty} \gamma^k \cdot R(s_k, a_k, s_{k+1})\right]$.

**Linear Temporal Logic (LTL):** LTL is a formal language to describe complex properties and high-level specifications of a system. LTL formulae are built inductively from atomic propositions by applying Boolean and temporal operators to subformulae. The syntax in Backus-Naur form is given by:

$$\phi ::= T \mid \mu \mid \phi_1 \land \phi_2 \mid \neg \phi \mid \phi_1 \lor \phi_2 \mid \phi \cdot \phi \mid \Box \phi,$$

where $\mu \in AP$ is an atomic proposition, $true \land$, negation $\neg$, and conjunction $\land$ are propositional logic operators, and $next \land until \land$ are temporal operators [26]. Alongside the standard operators introduced above, other propositional logic operators, such as $false \land disjunction \lor, and implication \rightarrow, and temporal operators, such as $always \land eventually \land$, are derived from the standard operators.

The semantics of an LTL formula are interpreted over words, where a word is an infinite (continuous) sequence $o = o_0a_1 \ldots , o_i \in 2^{AP}$ for all $i \geq 0$, where $2^{AP}$ represents the power set of $AP$. The satisfaction of an LTL formula $\phi$ by the word $o$ is denoted $o \models \phi$. The detailed semantics of LTL can be found in [26]. In this paper, we are interested in continuous-time control policies. The next operator is not always meaningful since it may require an immediate execution switch in the synthesized plans space. We can either exclude the next operator as is common in related work [23] or properly design practical LTL tasks [6] in continuous scenarios.

**III. PROBLEM FORMULATION**

Consider a cl-MDP $\mathcal{M} = (S, S_0, A, ps, R, \gamma, L)$. The induced path under a policy $\pi$ over $\mathcal{M}$ is $s_\infty = s_0 \ldots s_{i+1}$, where $ps(s_{i+1}|s_i, a_i) > 0$ if $\pi(a_i|s_i) > 0$.

We extend the labeling function to traces such that $L(s^\pi_\infty) = o_0a_1 \ldots$ is the sequence of labels associated with $s^\pi_\infty$. We denote the satisfaction relation of the induced trace with respect to $\phi$ by $L(s^\pi_\infty) \models \phi$. The probability of satisfying $\phi$ under the policy $\pi$, starting from an initial state $s_0 \in S_0$, is defined as

$$Pr^\pi_M(\phi) = Pr^\pi_M(L(s^\pi_\infty) \models \phi \mid s^\pi_\infty \in S^\pi_\infty),$$

where $S^\pi_\infty$ is the set of admissible paths from the initial state $s_0$, under the policy $\pi$ [26].

**Definition 2.** Given a cl-MDP $\mathcal{M}$, an LTL task $\phi$ is fully feasible if and only if there exists a policy $\pi$ s.t. $Pr^\pi_M(\phi) > 0$.

Note that according to Def. 2, an infeasible case means no policy exists that satisfies the task, which can be interpreted as $Pr^\pi_M(\phi) = 0, \forall \pi \in \Pi$.  

1447
Fig. 1. The figure shows the example described in Example 1, where the given LTL task is infeasible since some regions are surrounded by obstacles such that \( \exists \sigma \in \Sigma \). To quantify the difference between two elements in \( \Sigma \), consider \( \rho (\sigma, \sigma') = \|v - v'\|_1 = \sum_{i=1}^n |v_i - v'_i| \).

**Definition 3.** Given a cl-MDP \( \mathcal{M} \), the expected discounted violation cost with respect to a given LTL task \( \phi \) under the policy \( \pi \) is defined as

\[
J_V (\mathcal{M}^\pi, \phi) = E^\pi \left[ \sum_{k=0}^{\infty} c_V (s_k, a_k, s_{k+1}, \phi) \right],
\]

where \( c_V (s, a, s', \phi) \) is formally defined in Def. 8 as the violation cost of a transition \( (s, a, s') \) with respect to \( \phi \), and \( a_i \) is the action generated based on the policy \( \pi (s_i) \).

**Problem 1.** Given a robot with unknown dynamics \( \mathcal{S} \) in an \( \text{Env} \) with workspace \( X \) containing regions of interest, a navigation task in the form of an LTL task \( \phi \) over \( AP \), their interactions can be captured as a cl-MDP. The DRL objective is to find an optimal policy \( \pi^* \) with the following capabilities: (i) if \( \phi \) is feasible, \( \text{Pr}_{\pi^*} (\phi) > 0 \); (ii) if \( \phi \) is infeasible s.t. \( \text{Pr}_{\pi} (\phi) = 0 \), \( \forall \pi \in \Pi \), satisfy \( \phi \) with minimum violation cost via minimizing \( J_V (\mathcal{M}^\pi, \phi) \).

**Example 1.** In Fig. 1, consider an autonomous vehicle with unknown dynamics deployed in a complex, cluttered environment containing a set of labeled goal regions, \( AP_G = \{g_1, g_2, \ldots, g_5\} \), and labeled obstacles \( O \). The LTL specification is given by \( \phi = \Box (\Diamond g_1 \lor \Diamond (g_2 \land \Diamond \cdots \land \Diamond g_5)) \). We observe that some goal regions are surrounded by obstacles such that \( \phi \) is infeasible.

For LTL specifications, discrete goal-reaching may enable the automation transitions, which is one requirement to receive sparse automata-based rewards [6], [27]. However, in cluttered environments such as those depicted in Fig. 1, it can be difficult for noisy policies to explore the sparse rewards and reach regions of interest at all during learning.

**IV. SOLUTION**

In section IV-A, we develop an automaton model with relaxed constraints to address infeasible tasks. In section IV-B, we show how to synthesize a model-free plan with minimum violation cost in continuous space. Finally, in section IV-C, we propose a novel DRL design to learn optimal policies that solve Problem 1.

**A. Relaxed Product Automaton for Infeasible LTL**

Let \( \text{dist} : X \times X \rightarrow [0, \infty) \) define a metric function that computes the geometric Euclidean distance between two states. Prior work [7] introduced a transition system for an unknown dynamical system to capture the interactions between geometric space \( X \) and \( \text{Env} \).

**Definition 4.** A generalized weighted transition system (G-WTS) of \( \text{Env} \) is a tuple \( T = (X, x_0, \rightarrow_T, AP, L_X, C_T) \), where \( X \) is the configuration space of \( \text{Env} \), \( x_0 \) is the initial state of the robot; \( x \rightarrow_T x' \) if \( \text{dist}(x, x') \leq \eta \) and the straight line connecting \( x \) to \( x_{\text{new}} \) is collision-free; \( AP \) is the set of atomic propositions that label regions; \( L_X : X \rightarrow 2^{AP} \) is the labeling function that returns an atomic proposition satisfied at location \( x \); and \( C_T : (\rightarrow_T) \rightarrow \mathbb{R}^+ \) is the geometric Euclidean distance, i.e.; \( C_T (x, x') = \text{dist}(x, x'), \forall (x, x') \in \rightarrow_T \).

Let \( \tau_T = x_0 x_1 x_2 \ldots \) denote a valid run of \( T \). As opposed to the standard WTS [23] which includes dynamic state space \( S \) and explicit dynamics \( f(s, a) \) in transitions, only the geometric relations are available in our framework.

An LTL formula \( \phi \) can be converted to a Non-deterministic Büchi Automaton (NBA) to evaluate satisfaction.

**Definition 5.** [28] An NBA over \( 2^{AP} \) is a tuple \( B = (Q, q_0, \Sigma, \rightarrow_B, Q_F), \) where \( Q \) is the set of states, \( Q_0 \subseteq Q \) is the set of initial states, \( \Sigma = 2^{AP} \) is the finite alphabet, \( \rightarrow_B \subseteq Q \times \Sigma \times Q \) is the transition relation, and \( Q_F \subseteq Q \) is the set of accepting states.

A valid infinite run \( \tau_B = q_0 q_1 q_2 \ldots \) of \( B \) is called accepting, if it intersects with \( Q_F \) infinitely often. Infinite words \( \tau_B = q_0 q_1 q_2 \ldots, \forall \phi \in 2^{AP} \) generated from an accepting run satisfy the corresponding LTL formula \( \phi \). We use \( B_{\phi} \) to denote the NBA of LTL formula \( \phi \).

A common approach for synthesizing a geometric plan satisfying LTL formula \( \phi \) is to construct the product Büchi automaton (PBA) between the G-WTS \( T \) and the NBA \( B [7], [23] \). This approach assumes the given \( \phi \) is feasible. Inspired by [14], [15] to handle infeasible cases, we introduce a relaxed PBA and define corresponding violation costs.

**Definition 6.** Given the G-WTS \( T \) and the NBA \( B_{\phi} \), the relaxed PBA is a tuple \( P = T \times B = (Q_P, Q_{F_P}, \rightarrow_P, Q_{F_P}, \rightarrow_P, \Sigma_L, \Sigma_R, L_P, \Sigma) \), where

- \( Q_P = X \times Q \) is the set of infinite product states, \( Q_{F_P} = x_0 \times Q \) is the set of initial states, \( Q_{F} = X \times Q_F \) is the set of accepting states, and \( \Sigma = 2^{AP} \).
- \( \rightarrow_P \subseteq Q \times Q_P \) is the transition function such that \( q_P = (x, q) \rightarrow_P q'_{P} = (x', q') \) is a valid transition if and only if the following two conditions hold: \( x \rightarrow_T x' \) and \( \exists q_{F_P} \in \Sigma_L \text{ s.t. } q 
\rightarrow_{B_{\phi}} q' \).
- \( c_P : (\rightarrow_P) \rightarrow \mathbb{R}^+ \) is the geometric cost function defined as the cost in the configuration space, e.g., \( c_P (q_P) = (x, q), q'_{P} = (x', q') = C_T (x, x'), \forall (q_P, q'_{P}) \in \rightarrow_P \).
- \( L_P : Q_F \rightarrow \Sigma \) is the labeling function s.t. \( L_P (q_P) = L_X (x), \forall q_{F_P} = (x, q) \).
- \( c_V : (\rightarrow_P) \rightarrow \mathbb{R}^+ \) is the violation cost function for each transition with respect to the associated LTL formula \( \phi \).

The major novelty of the relaxed PBA is replacing the transition constraint \( q 
\rightarrow_{B_{\phi}} q' \) of a standard PBA [26] with \( \exists q_{F_P} \in \Sigma \text{ s.t. } q 
\rightarrow_{B_{\phi}} q' \) to construct a more connected product graph. Next, we define the computation of the violation function \( c_{V_P} \).

Suppose \( AP = \{\mu_1, \mu_2, \ldots, \mu_M\} \) and consider an evaluation function \( \text{Eval} : \Sigma \rightarrow \{0, 1\}^M \), where \( \Sigma = 2^{AP} \) and \( M = |AP| \). \( \text{Eval} [\sigma] = [v_i]_{i=1}^M \) and \( v_i \) is a binary variable s.t. \( v_i = 1 \) if \( \mu_i \in \sigma \) and \( v_i = 0 \) if \( \mu_i \notin \sigma \), where \( i = 1, 2, \ldots, M \) and \( \sigma \in \Sigma \). To quantify the difference between two elements in \( \Sigma \), consider \( \rho (\sigma, \sigma') = \|v - v'\|_1 = \sum_{i=1}^M |v_i - v'_i| \).
where \( v = \text{Eval}(\sigma), v' = \text{Eval}(\sigma'), \sigma, \sigma' \in \Sigma, \) and \( \| \cdot \|_1 \) is the \( L_1 \) norm. The distance from \( \sigma \in \Sigma \) to a set \( X \subseteq 2^\Sigma \) is then defined as 
\[
D_V(\sigma, X) = \min_{\sigma' \in X} \rho(\sigma, \sigma') \quad \text{if} \quad \sigma \notin X,
\]
and 
\[
D_V(\sigma, X) = 0
\]
Def. 7. In a relaxed PBA, the violation cost of the transition \( q_p = (x, q) \rightarrow_p q'_p = (x', q') \) imposed by \( \phi \) can be computed with 
\[
c_V(q_p, q'_p) = D_V(L_p(q_p), X(q', q'))
\]
where \( X(q, q') = \{ \sigma_{AP} \in \Sigma \mid q \rightarrow_B \sigma \} \) is the set of input symbols enabling the transition from \( q \) to \( q' \) in \( B_0 \).

Based on that, we can also measure the violation cost imposed on the dynamic system \( \Sigma \) as follows.

Def. 8. Given a transition \((s, a, s')\) of dynamics and current automata state \( q \), the next automata state \( q' \) can be generated on-the-fly as \( q \rightarrow_B \phi q' \). The violation cost of the transition with respect to \( \phi \) can be obtained as 
\[
c_V(s, a, s', \phi) = D_V(L(s), X(q', q'))
\]

Remark 2. Note that directly applying \( c_V(s, a, s', \phi) \) into reward design for LTL satisfaction formulates a multi-objective DRL problem. T The optimality performance of multi-objective DRL as a nonlinear regression process is hard to control. Instead, we use path planning workspace \( X \) for DRL guidance.

Given a valid trace \( \tau_P = q^0_P q^1_P q^2_P \ldots \) of a PBA, the total violation cost can be computed with 
\[
W_V(\tau_P) = \sum_{i=0}^{\infty} c_V(q^i_P, q^{i+1}_P)
\]
The trace \( \tau_P = q^0_P q^1_P q^2_P \ldots \) satisfies the acceptance conditions if it intersects with \( Q_F \) infinitely often. Its corresponding words are obtained as \( \tau_0 = q_0 q_1 \ldots \), \( \forall q_i = L_P(q^i_P) \). Now, the LTL satisfaction according to the accumulated violation cost can be measured as follows:

Lemma 1. [14] The words \( \tau^*_P \) of an optimal trace \( \tau^*_P \) satisfy the corresponding potentially infeasible LTL specification \( \phi \) with minimum-violation guarantees if and only the following two conditions hold: (i) \( \tau^*_P \) satisfies the acceptance condition and (ii) \( W_V(\tau^*_P) \) has the minimum total violation cost, i.e.,
\[
\tau^*_P = \min_{\tau \in \tau^P} W_V(\tau_P), \quad \tau_{\infty} \text{ denotes the set of all valid traces in the relaxed PBA } P.
\]
The word \( \tau_0^* \) of \( \tau^*_P \) satisfies \( \phi \) exactly if \( W_V(\tau^*_P) = 0 \). Let \( proj|_X : Q_F \rightarrow X \) denote a projection s.t. \( proj|_X(q_p) = x, \forall q_p = (x, q) \). We use this operator to extract the geometric trajectory \( \tau^*_G = proj|_X(\tau^*_P) \) for minimally-violating LTL satisfaction from the optimal trace \( \tau^*_P \).

Remark 3. This work considers unknown dynamic systems so that the generated path \( \tau^*_G \) is only optimal in the sense of geometric relations e.g., shortest euclidean distance.

B. Minimum-violation Synthesis and Decomposition

Minimum-violation Synthesis: Since this work considers continuous control and the geometric space \( X \) of \( T \) is also continuous, it’s impossible to explicitly construct the relaxed PBA, \( P \), and find the optimal trace \( \tau^*_P \). Instead of discretizing space, we apply Temporal Logic Rapidly-Exploring Random Trees (TL-RRT*) [23], which is abstraction-free and builds a tree in the configuration space incrementally. This technique has the same properties as RRT* [24], [29]; it can generate probabilistically complete and asymptotically optimal solutions.

Formally, let \( \tau_F \) denote any accepting run of PBA \( P \). TL-RRT* leverages the fact that an accepting run \( \tau_F \) is a lasso-type sequence with a prefix-suffix structure, i.e., \( \tau_F = \tau_F^{pre}[\tau_F^{suf}]^\omega \), where the prefix part \( \tau_F^{pre} = q^0_P q^1_P \ldots q^k_P \) is only executed once, and the suffix part \( \tau_F^{suf} = q^{k+1}_P \ldots q_P^\infty \) loops infinitely. Following this idea, we build the prefix and suffix optimal trees, respectively. To ensure the acceptance condition, the set of goal states of the prefix tree \( G^{pre} = (V^{pre}, E^{pre}) \) is defined as \( V^{pre} = \{ q_p = (x, q) \in X_{free} \times Q \subseteq Q_P | q \in Q_F \} \), where \( X_{free} \) is the collision-free configuration space. The optimal goal states of the prefix tree are \( Q_F = V^{pre} \cap Q^{goal} \) that is regarded as the roots of suffix tree \( G^{suf} = (V^{suf}, E^{suf}) \).

The destination states of the suffix tree are 
\[
Q^{suf} = \{ q_p = (x, q) \in X_{free} \times Q \subseteq Q_P | q_p \text{ reaches } \}
\]
We refer readers to [23] for more details.

Unlike [23], we consider infeasible cases where rapidly-exploring random trees should be built incrementally based on the relaxed PBA and our optimization objective includes both violation and geometric cost. To guarantee total-minimum-violation, let the weight between any two product states in the tree be \( w(q_p, q'_p) = c_p(q_p, q'_p) + \beta c_V(q_p, q'_p) \), where we select \( \beta \gg 1 \) to ensure the violation cost \( c_V \) has higher priority than geometric distance \( c_p \). Finally, we can find the minimum-violation path in the form of \( \tau^*_P = \tau^*_F[\tau^*_G]^\omega \), which can be projected into the configuration space \( \tau^*_T = proj|_X(\tau^*_P) = \tau^*_F[\tau^*_G]^\omega \).

Optimal decomposition: Since directly applying the whole geometric path \( \tau^*_T = proj|_X(\tau^*_P) \) with long or infinite horizons as guidance for DRL suffers from myopic tendencies (gathering intermediate rewards rather than reaching the desired region), we can decompose \( \tau^*_T \) into several sub-paths. To do so, we rewrite \( \tau^*_P = \tau^*_F[\tau^*_G]^\omega \) such that \( \forall i \in \{0, 1, \ldots, K+1, \ldots \} \), \( \tau^*_i \) is a sub-trajectory and the product states of each \( \tau^*_i \) have the same automaton components i.e., \( q = q_i, \forall q_p = (x, q) \in \tau^*_G \). Each segment \( \tau^*_i \) can be projected into workspace as \( \tau^*_T, i \), and we have 
\[
\tau^*_T = proj|_X(\tau^*_P) = \tau^*_F[\tau^*_G]^\omega.
\]

Let \( proj|_Q : Q_P \rightarrow Q \) denote a projection s.t. \( proj|_Q(q_p) = q, \forall q = (x, q) \) to extract the automaton components, and we have 
\[
\tau^*_Q = proj|_Q(\tau^*_P) = q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9 q_{10} q_{11} \ldots.
\]

As a result, we can define each segment \( \tau^*_T, i \) as a reach-avoid path \( R_i(\tilde{G}, \tilde{O}) \), where \( \tilde{G} \) is its goal region and \( \tilde{O} \) is a set of all obstacles. The lasso-type geometric reach-avoid path can be written
\[
R_T = (R_0 \ldots R_{K})(R_{K+1} \ldots R_{K+L})\ldots
\]
Lemma 2 states that we can measure the total violation cost of an infinite continuous path using a finite discrete form. The structure of RT can enable a modular structure of distributively solving each $R_i$ with shorter horizons. Note that, the above process generates the asymptotically minimally-violating plan that is probabilistically complete.

**Example 2.** Fig. 2 shows an example of the optimal decomposition for the inflexible case. The LTL task is $\phi = [\square \neg O \land \phi_g^1] = [\square \triangledown G_1 \land \square \triangledown G_2 \land \square \triangledown G_3]$, where $\phi_g^1$ is not accessible. The resulting NBA and decomposed trajectories of TL-RRT* minimally-violating $\phi$ are shown in Fig. 2 (left) and right), respectively, where decomposed reach-avoid paths (sub-tasks) are expressed as $R_T = R_{red}(R_{blue}, R_{brown})^\omega$.

**C. Reward Design and Distributed DRL**

**Reward design:** According to the optimal compositional plan $R_T$, this section designs a reward function for each $R_i$ that is dense in the configuration space to overcome the challenge of the complex environment. We can train each reach-avoid $R_i$ navigation control in the same way distributively. Even if we can straightforwardly use the geometric sub-path for reward design, there exist three issues: (a) without considering actual dynamical systems, controllers cannot strictly track the geometric path; (b) the robot may linger around previous waypoints resulting in sub-optimal solutions; (c) the waypoints might overlap with obstacles. We address them in the following descriptions, and the intuition of the reward design is shown in Fig. 1 (b), where the robot just needs to move towards the goal and is not required to strictly follow the geometric path.

(a): Each geometric path of $R_i$ is a sequence of waypoints $\tau^i T = (x_1, x_2, \ldots x_N)$. We construct an r-norm ball for each state $x_{t,i} \in \tau^i T$, $\forall j$, as $Ball_r(x_{t,i}) = \{ x \in X | dist(x, x_{t,i}) \leq r \}$. These balls allow the robot to pass them and move towards the goal as guidance, which we use to design a reward function. The consecutive balls can be regarded as a tube which is popular in the robust control community [30] to handle path tracking insensitivity.

(b): The intuitive approach is to track the minimum distance of the visited balls to the destination so far and utilize it as a constraint in the reward design. Since TL-RRT* is an extension of RRT* [29], it also provides the optimal distance function in the tree from each state to the global final destination denoted as $Cost(x)$. For a sub-path $\tau^i T = (x_1, x_2, \ldots x_N)$, we compute the distance from each state $x_{i,j} \in \tau^i T$ to the destination $x_i,N$ as $Dist(x_{i,j}) = Cost(x_{i,j}) - Cost(x_{i,N})$. We return the necessary distance information as:

$$D(x) = \begin{cases} Dist(x_{i,j}|x^*) & \text{if } x \in Ball_r(x_{i,j}|x^*) \\ \infty & \text{otherwise} \end{cases}$$ (5)

During each episode of training, a state-action sequence $s_0 \pi_0 a_1 \ldots s_t$ up to current time $t$ is split into the state and action sequences $s_t = s_0 s_1 \ldots s_t$ and $a_t = a_0 a_1 \ldots a_{t-1}$, respectively. We develop a progression function $\Delta_{min}$ to identify the next state is getting closer to the goal region as $\Delta_{min}(s_t) = \min \{D(Proj(s_t))\}$.

The learning objective is to push $\Delta_{min}$ to decrease at each time-step. We design a constrained reward for each time $t$ of an episode as $R(s_t^\pi) = r_+ > 0$ if $D(Proj(s_t)) < \Delta_{min}(s_t)$, Such a design also alleviates the first issue (a), since the RL agent is not required to visit every waypoint.

(c): We design the reward in different priorities such that if the RL agent collides with obstacles, it will always return the negative reward and ignore other conditions. In summary, our reward design is the following with decreasing priorities:

$$R(s_t) = \begin{cases} -r_- & \text{if } \text{Proj}(s_t) < X_0 \\ +r_+ & \text{if } \text{Proj}(s_t) > 0 \\ 0 & \text{if } \text{Proj}(s_t) < \Delta_{min}(s_t) \end{cases}$$ (6)

**Theorem 1.** If there exists at least one policy $\pi^*_{R,i}$ satisfying the reach-avoid task $R^i$, by selecting $r_+ > 0$ to be sufficiently larger than $r_-$, i.e., $r_+ \gg r_-$, any algorithm that optimizes the expected return $J(\pi)$ is guaranteed to find such an optimal policy $\pi^*_{R,i}$.

**Proof.** The detailed proof can be found in [31].

Note that even though training deep neural networks to optimize policies is a nonlinear regression process, we can use neural network verification techniques to certify and improve the learned policy $\pi^*_{R,i}$ for a reach-avoid task.

**Remark 4.** Employing $\Delta_{min}(s_t)$ to design a reward function results in a non-Markovian property, since it relies on the past history $s_t$, whereas the reward function (6) only depends on current state. To address this, we can leverage the structure of the product MDP [7], [26], and augment the current state $s_t$ with the index of the closest visited waypoint, to construct a product state to recover the Markovian property.

**Distributed DRL:** For each $R_i$, we can apply the above approach to an existing off-the-shelf DRL algorithms, e.g., SAC [2], to train the optimal policy $\pi^*_{R,i}$. The process can be repeated for all $R_i, \forall i \in \{0, 1, \ldots K + 1\}$ of $R_T = (R_0 \ldots R_K)(R_{K+1} \ldots R_{K+l})^\omega$ in a distributed manner. We then concatenate them as the globally optimal policy

$$\pi^*_{\phi} = (\pi^*_{R_0, i} \pi^*_{R_1, i} \ldots \pi^*_{R_K, i})(\pi^*_{R_{K+1}, i} \ldots \pi^*_{R_{K+l}, i})^\omega$$ (7)

where $\pi^*_{\phi}$ is the global policy satisfying LTL task $\phi$ and containing a set of neural network parameters. Note that if we train each reach-avoid task $R_i$ individually, we need to randomize the initial velocity and acceleration conditions of state $S$, since the ending condition of completing a policy $\pi^*_{R,i}$ results in different starting states for the next policy $\pi^*_{R,i+1}$. This enables a smooth concatenation instead of retraining the whole global policy $\pi^*_{\phi}$.

**Theorem 2.** If for all $i \in \{0, 1, \ldots K + l\}$, $\pi^*_{R,i}$ satisfies its corresponding reach-avoid task $R_i$ of $R_T$ in (4) under...
any initial states (velocities, accelerations), then their concatenation \( \pi^*_s \) in (7) satisfies the LTL task \( \phi \) with minimum-violation guarantees.

Proof: The intuition is to prove that every admissible path \( s^*_{\infty} \in S^*_{\infty} \) under the optimal policy \( \pi^*_s \) satisfies the LTL specification \( \phi \) with minimum total violation cost. The proof is based on the decomposition property in Section IV-B.

First, the geometric projection of every admissible path \( s^*_{\infty} \in S^*_{\infty} \) satisfies the lasso-type reach-avoid \( R_T \) in (4), which is associated with a lasso-type automaton path \( \tau^*_Q \) as (3). From Lemma 2 and the optimality of TL-RRT* [23], \( \tau^*_Q \) satisfies the acceptance condition of the NFA of LTL formula \( \phi \) and has the minimum total violation cost. \( \square \)

Remark 5. Due to the conclusion of minimally-violating the LTL task from Theorem 2, our framework generalizes to both feasible and infeasible cases for a given LTL task.

V. EXPERIMENTAL RESULTS

We implement the framework on two different nonlinear dynamical systems for various LTL tasks over both finite and infinite horizons. The algorithm test focuses on infeasible cases and cluttered environments, where dense cluttered obstacles are randomly sampled. We show that the framework can complete feasible LTL tasks exactly. We apply SAC [2] as a state-of-art DRL algorithm for all baselines.

**Baseline Approaches:** From the learning perspective, we refer to our distributed framework as “RRT*” or “D-RRT*”, and compare it against three baselines: (i) The relaxed TL-based multi-objective rewards in [6], [18] referred to as “TL”, for the single LTL task; (ii) For the goal-reaching task \( \phi \), the baseline referred to as “NED” designs the reward based on the negative Euclidean distance between the robot and destination; (iii) For a complex LTL task, instead of decomposition, this baseline directly applies the reward scheme (6) for the global trajectory \( \tau^*_g = \tau^*_{pre}[\tau^*_{suf}] \) referred to as “G-RRT*”.

From the perspective of infeasible LTL tasks, we compare with the work [17] of visiting as many automaton accepting sets as possible, and we empirically show the improvement over our prior work [7] that assumes the feasible cases.

**Autonomous Vehicle** We first implement the Dubins car model in the Pybullet physics engine shown in Fig. 3. We consider the surveillance LTL task \( \phi_{1,inf} = \square \neg O \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \) that requires sequentially visiting regions labeled as \( G_1, G_2, G_3 \), and the robot’s initial position infinitely often. Its finite-horizon version can be expressed as \( \phi_{1,f} = \square \neg O \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \) by removing the *always* operator \( \square \). Both tasks are infeasible since the region \( G_2 \) is surrounded by obstacles. Fig. 3 shows the learning curves of task \( \phi_{1,inf} \) compared with different baselines, and \( \phi_{1,f} \), has the same comparison results. We can observe that our framework can provide better performance than other baselines under the challenge of complex cluttered environments.

**Quadrotor Model** We test our algorithms in a 3D environment with Quadrotor dynamics shown in Fig. 4. It demonstrates that our model-free framework is capable of handling complex cluttered environments with high dimensions. The LTL specification is given as \( \phi_{2,inf} = \square \neg O \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \), which requires the quadrotor to navigate regions of \( G_1, G_2, G_3 \) infinitely often without specific orders. Its finite version can also be defined as \( \phi_{2,f} = \square \neg O \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \). These two tasks are also infeasible and, their comparisons of learning performances are the same. Due to page limitation, we only show the learning results for the task \( \phi_{2,inf} \) in Fig 4.

**Feasibility Generalization** According to theorem 2, the advantage of minimally violating a given task allows generalization of feasible cases. To show the metric, we remove the highlighted obstacles for the region \( G_2 \) in both environments, and implement our algorithms. The feasible experimental results shown in the video demo demonstrate that our framework can satisfy feasible tasks exactly.

5.5 Infeasibility Analysis For each environment and dynamical system, we increase the task complexity by randomly sampling 12 obstacle-free goal regions in both environments and set the specifications as \( \phi_{3,inf} = \square \neg O \land (\Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) \land \Box (G_1 \land \Box (G_2 \land \Box (G_3 \land \Box (G_{init})))) ) \). We repeat the random sampling for 50 trials and record the success rates of generating valid plans for two environments. The results for these task \( \phi_{3,inf} \) compared with our prior work [7] referred to as ”feasible” are shown in Fig. 6 (a). Due to the random process, the target regions may be surrounded by obstacles, and the baseline returns no solutions, whereas our framework always finds minimally-violating trajectories.

From the aspect of automaton-based rewards, the work [18]
directly applies a linear combination of violation cost and automaton acceptance reward to formulate a MORL process, with no guarantees on satisfaction and violation. Our proposed framework leveraging planning methods as guidance improves the performance. This seen in the learning comparisons in Fig. 3 and Fig. 4. Moreover, we also compare the work [17] that addresses infeasible cases for discrete MDP model. It finds the policy that satisfies the given LTL tasks as much as possible by intersecting with the maximum number of automaton accepting sets. However, the LDGBAs of some LTL formulas only have one accepting set. As an example, Fig. 5 shows the LDGBA of $\phi_{1, \text{fin}} = \square \tau_1 \lor \Diamond (G_1 \lor \Diamond (G_2 \lor \Diamond (G_3)))$ for an autonomous vehicle that has only one accepting set $\{q_1\}$. The same case also applies to the LTL formula $\phi_{1, \text{fin}}$, $\phi_{3, \text{fin}}$. In such cases, the work [17] returns no solution since the only accepting set is not accessible.

5.6 Training efficiency We record the training time for each reach-avoid sub-task for the finite-horizon LTL tasks i.e., $\phi_{1, \text{fin}}$, $\phi_{2, \text{fin}}$, $\phi_{3, \text{fin}}$ in two environments. We compared with the baseline "NED" that distributes solved each reach-avoid sub-task using distance based rewards. The results in Fig. 6 (b) shows that our framework is more efficient.

VI. CONCLUSION

This article proposes a model-free framework using DRL to learn continuous controllers for completing complex LTL navigation tasks. We overcome the challenge of infeasible LTL specifications in complex cluttered environments. To minimally violate the LTL task, we apply geometric path planning methods as guidance of DRL that can guarantee minimal total violation cost, which can decompose the task into sub-reach-avoid missions. Based on the advantage of minimal violation, our work generalizes both feasible and infeasible cases. Future work will explore safety-critical exploration and bridge the gap of sim-to-real to deploy the framework in real-world robots.

REFERENCES

[1] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.
[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in Intl Conference on Machine Learning, 2018, pp. 1861–1870.
[3] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications," in IEEE Conference on Decision and Control, 2014, pp. 1091–1096.
[4] Y. Kantaros, "Accelerated reinforcement learning for temporal logic control objectives," Intl Conf on Intelligent Robots and Systems, 2022.
[5] M. Cai, S. Xiao, B. Li, Z. Li, and Z. Kan, "Reinforcement learning based temporal logic control with maximum probabilistic satisfaction," in IEEE Intl Conf on Robotics and Automation, 2021, pp. 806–812.
[6] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 7973–7980, 2021.
[7] M. Cai, E. Aasi, C. Belta, and C.-I. Vasile, "Overcoming exploration: Deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications," IEEE Robotics and Automation Letters, 2023.
[8] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in Proc. Int. Conf. Hybrid syst., Comput. Control, 2013, pp. 1–10.
[9] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," IEEE Transactions on Robotics, vol. 32, no. 3, pp. 583–599, 2016.
[10] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in IEEE Intl Conference on Robotics and Automation (ICRA), 2017, pp. 1481–1488.
[11] B. Lacerda, F. Farouk, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," Intl Journal of Robotics Research, vol. 38, no. 9, pp. 1098–1112, 2019.
[12] H. Rahmani and J. M. O’Kane, "What to do when you can’t do it all: Temporal logic planning with soft temporal logic constraints," in Intl Conference on Intelligent Robots and Systems, 2020, pp. 6619–6626.
[13] T. Wongpiromsarn, K. Slutsksy, E. Frazzoli, and U. Topcu, "Minimum-violation planning for autonomous systems: Theoretical and practical considerations," in American Control Conference, 2021, pp. 4866–4872.
[14] K. Kim, G. Fainekos, and S. Sankaranarayanan, "On the minimal revision problem of specification automata," The International Journal of Robotics Research, vol. 34, no. 12, pp. 1515–1535, 2015.
[15] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," International Journal of Robotics Research, vol. 34, no. 2, pp. 218–235, 2015.
[16] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible ltl constraints," IEEE Transactions on Automatic Control, 2021.
[17] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in IEEE Conference on Decision and Control (CDC), 2019, pp. 5338–5343.
[18] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic ltl motion planning with environment and motion uncertainties," IEEE Tran on Automatic Control, vol. 66, no. 5, pp. 2386–2392, 2021.
[19] K. Van Muffert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 3483–3512, 2014.
[20] S. J. Qin and T. A. Badgewell, "A survey of industrial model predictive control technology," Control engineering practice, vol. 11, no. 7, pp. 733–764, 2003.
[21] I. R. Manchester and J.-J. E. Slotine, "Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design," IEEE Tran on Automatic Control, vol. 62, no. 6, pp. 3046–3053, 2017.
[22] M. Chen, S. L. Herbert, H. Hu, Y. Pu, J. F. Fisac, S. Bansal, S. Han, and C. J. Tomlin, "Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking," IEEE Transactions on Automatic Control, vol. 62, no. 12, pp. 5861–5873, 2017.
[23] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multirobot temporal logic optimal control synthesis," IEEE Transactions on Robotics, 2021.
[24] C. I. Vasile, X. Li, and C. Belta, "Reactive Sampling-Based Path Planning with Temporal Logic Specifications," International Journal of Robotics Research, vol. 39, no. 8, pp. 1002–1028, June 2020.
[25] E. A. Coddington and N. Levinson, "Theory of ordinary differential equations." McGraw-Hill, pp. 1–42, 1955.
[26] C. Bailer and J.-P. Katoen, Principles of model checking. MIT press, 2008.
[27] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," Science Robotics, vol. 4, no. 37, p. eazy2767, 2019.
[28] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in 1st Symposium on Logic in Computer Science. IEEE Computer Society, 1986, pp. 322–331.
[29] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The international journal of robotics research, vol. 30, no. 7, pp. 846–894, 2011.
[30] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," International Journal of Adaptive Control and Signal Processing, vol. 21, no. 11, pp. 1341–1351, 2007.
[31] M. Cai, M. Mann, Z. Serlin, K. Leahy, and C.-I. Vasile, "Learning minimal-violating continuous control for infeasible temporal logic specifications," arXiv preprint arXiv:2210.01162, 2022.