

# Overcoming Exploration: Deep Reinforcement Learning for Continuous Control in Cluttered Environments From Temporal Logic Specifications

Mingyu Cai , *Member, IEEE*, Erfan Aasi, Calin Belta , *Fellow, IEEE*, and Cristian-Ioan Vasile 

**Abstract**—Model-free continuous control for robot navigation tasks using Deep Reinforcement Learning (DRL) that relies on noisy policies for exploration is sensitive to the density of rewards. In practice, robots are usually deployed in cluttered environments, containing many obstacles and narrow passageways. Designing dense effective rewards is challenging, resulting in exploration issues during training. Such a problem becomes even more serious when tasks are described using temporal logic specifications. This work presents a deep policy gradient algorithm for controlling a robot with unknown dynamics operating in a cluttered environment when the task is specified as a Linear Temporal Logic (LTL) formula. To overcome the environmental challenge of exploration during training, we propose a novel path planning-guided reward scheme by integrating sampling-based methods to effectively complete goal-reaching missions. To facilitate LTL satisfaction, our approach decomposes the LTL mission into sub-goal-reaching tasks that are solved in a distributed manner. Our framework is shown to significantly improve performance (effectiveness, efficiency) and exploration of robots tasked with complex missions in large-scale cluttered environments.

**Index Terms**—Formal methods in robotics and automation, deep reinforcement learning, sampling-based method.

## I. INTRODUCTION

MODEL-FREE Deep Reinforcement Learning (DRL) employs neural networks to find optimal policies for unknown dynamic systems via maximizing long-term rewards [1]. In principle, DRL offers a method to learn such policies based on the exploration vs. exploitation trade-off [2], but the efficiency of the required exploration has prohibited its usage in real-world

Manuscript received 26 September 2022; accepted 1 February 2023. Date of publication 20 February 2023; date of current version 2 March 2023. This letter was recommended for publication by Associate Editor A. Fagiolini and Editor L. Pallottino upon evaluation of the reviewers' comments. This work was supported in part by the Under Secretary of Defense for Research and Engineering under Air Force Contract FA8702-15-D-0001 at Lehigh University and in part by the NSF under Grant IIS-2024606 at Boston University. (*Corresponding author: Mingyu Cai.*)

Mingyu Cai and Cristian-Ioan Vasile are with the Mechanical Engineering, Lehigh University, Bethlehem, PA 18015 USA (e-mail: mic221@lehigh.edu; crv519@lehigh.edu).

Erfan Aasi and Calin Belta are with the Mechanical Engineering Department, Boston University, Boston, MA 02215 USA (e-mail: eaasi@bu.edu; cbelta@bu.edu).

A video demonstration can be found on YouTube Channel: [https://youtu.be/yMh\\_NUNWxho](https://youtu.be/yMh_NUNWxho).

Digital Object Identifier 10.1109/LRA.2023.3246844

robotic navigation applications due to natural sparse rewards. To effectively collect non-zero rewards, existing DRL algorithms simply explore the environments, using noisy policies and goal-oriented reward schemes. As the environment becomes cluttered and large-scale, naive exploration strategies and standard rewards become less effective resulting in local optima. This problem becomes even more challenging for complex and long-horizon robotic tasks. Consequently, the desired DRL-based approaches for robotic target-driven tasks are expected to have the capability of guiding the exploration during training toward task satisfaction.

*Related Work:* For exploration in learning processes, many prior works [3], [4], [5], [6], [7] employ noise-based exploration strategies integrated with different versions of DRLs, whereas their sampling efficiency relies mainly on the density of specified rewards. Recent works [8], [9] leverage human demonstrations to address exploration issues for robotic manipulation. On the other hand, the works in [10], [11] focus on effectively utilizing the dataset stored in the replay buffer to speed up the training. However, these advances assume a prior dataset is given beforehand, and they can not be applied to learn from scratch. In cluttered environments containing dense obstacles and narrow passageways, their natural rewards are sparse, resulting in local optimal behaviors and failure to reach destinations.

The common problem is how to generate guidance for robot navigation control in cluttered environments. Sampling-based planning methods, such as Rapidly-exploring Random Tree (RRT) [12], RRT\* [13] and Probabilistic Road Map (PRM) [14], find collision-free paths over continuous geometric spaces. In robotic navigation, they are typically integrated with path-tracking control approaches such as Model Predictive Control (MPC) [15], control contraction metrics [16] and learning Lyapunov-barrier functions [17]. While these methods are effective, their control designs are model-based. Model-free path tracking, which is the focus of this paper, is still an open problem.

Furthermore, this work also considers complex navigation tasks instead of simple goal-reaching requirements. Motivated by task-guided planning and control, formal languages are shown to be efficient tools for expressing a diverse set of high-level specifications [18]. For unknown dynamics, temporal logic-based rewards are developed and integrated with various DRL algorithms. In particular, deep Q-learning is employed in [19], [20], [21] over discrete action-space. For continuous state-action spaces, the authors in [22], [23], [24] utilize actor-critic algorithms, e.g., proximal policy optimization (PPO) [5],

validated in robotic manipulation and safety tasks, respectively. All aforementioned works only study LTL specifications over finite horizons. To facilitate defining LTL tasks over infinite horizons, recent works [25], [26] improve the results from [27] by converting LTL into a novel automaton called E-LDGBA, a variant of the Limit Deterministic Generalized Büchi Automaton (LDGBA) [28]. To improve the performance for the long-term (infinite horizon) satisfaction, the authors propose a modular architecture of Deep Deterministic Policy Gradient (DDPG) [4] to decompose the global missions into sub-tasks. However, none of the existing works can address large-scale, cluttered environments, since an LTL-based reward requires the RL-agent to visit the regions of interest towards the LTL satisfaction. Such sparse rewards can not tackle challenging environments. Sampling-based methods and reachability control synthesis for LTL satisfaction are investigated in [29], [30], [31], [32]. All assume known system dynamics. In contrast, our paper proposes a model-free approach for LTL-based navigation control in cluttered environments.

*Contributions:* Intuitively, the most effective way of addressing the environmental challenges of learning is to optimize the density of rewards such that the portion of transitions with positive rewards in the reply buffer is dramatically increased. To do so, we bridge the gap between sampling-based planning and model-free DPGs to solve standard reachability problems. In particular, we develop a novel exploration guidance technique using geometric RRT\* [13], to design the rewards. We then propose a distributed DRL framework for LTL satisfaction by decomposing a global complex and long-horizon task into individual reachability sub-tasks.

Moreover, we propose an augmentation method to address the non-Markovianity of the reward design process. Due to unknown dynamics, we overcome the infeasibility of the geometric RRT\* guidance. Our algorithm is validated through case studies to demonstrate its performance increase compared to the distance and goal-oriented baselines. We show that our method employing geometric path planning guidance achieves significant training improvements for DRL-based navigation in cluttered environments, where the tasks can be expressed using LTL formulas.

*Organization:* Section II introduces basic concepts of robot dynamics, Markov Decision Processes (MDP) that capture the interactions between robot and environment, RL for solving learning-based control for the MDP model, and LTL for defining robot navigation specifications. In Section III, we define the problem and emphasize the challenges. Section IV presents our approach for addressing simple goal-reaching tasks via reward design using sampling-based methods over the workspace. In Section V, we show how to use and extend the proposed approach to general LTL mission specifications. The performance of the proposed method is shown in Section VI.

## II. PRELIMINARIES

The evolution of a continuous-time dynamic system  $\mathcal{S}$  starting from an initial state  $s_0 \in S_0 \subset S$  is given by

$$\dot{s} = f(s, a), \quad (1)$$

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, and  $S_0$  is the initial set. The function  $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$  is locally Lipschitz continuous and unknown.

Consider a robot with the unknown dynamics (1), operating in an environment  $Env$  that is represented by a compact subset  $X \subset \mathbb{R}^d$ ,  $d \in \{2, 3\}$  as a workspace of the robot. The relation between  $S$  and  $X$  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  $AP$ . We use  $2^{AP}$  to represent the power set of  $AP$ . We denote  $L_X : X \rightarrow 2^{AP}$  to label regions in the workspace. Let  $L : S \rightarrow 2^{AP}$  be the induced labeling function over  $S$  and we have  $L(s) = L_X(Proj(s))$ . Note that  $S$  represents the robot state space that can be high-dimensional while  $X$  is the workspace it is deployed in, i.e., two or three-dimensional Euclidean space.

*Reinforcement Learning:* The interactions between environment  $Env$  and the unknown dynamic system  $\mathcal{S}$  with the state-space  $S$  can be captured by a continuous labeled-MDP (cl-MDP) [33]. A cl-MDP is a tuple  $\mathcal{M} = (S, S_0, A, p_S, AP, L, R, \gamma)$ , where  $S \subseteq \mathbb{R}^n$  is the continuous state space,  $S_0$  is the set of initial states,  $A \subseteq \mathbb{R}^m$  is the continuous action space,  $p_S$  represents the unknown system dynamics as a distribution,  $AP$  is the set of atomic propositions,  $L : S \rightarrow 2^{AP}$  is the labeling function,  $R : S \times A \times S \rightarrow \mathbb{R}$  is the reward function, and  $\gamma \in (0, 1)$  is the discount factor. The distribution  $p_S : \mathfrak{B}(\mathbb{R}^n) \times A \times S \rightarrow [0, 1]$  is a Borel-measurable conditional transition kernel, s.t.  $p_S(\cdot | s, a)$  is the probability measure of the next state given current  $s \in S$  and  $a \in A$  over the Borel space  $(\mathbb{R}^n, \mathfrak{B}(\mathbb{R}^n))$ , where  $\mathfrak{B}(\mathbb{R}^n)$  is the set of all Borel sets on  $\mathbb{R}^n$ .

Let  $\pi(a|s)$  denote a policy that is either deterministic, i.e.,  $\pi : S \rightarrow A$ , or stochastic, i.e.,  $\pi : S \times A \rightarrow [0, 1]$ , which maps states to distributions over actions. At each episode, the initial state of the robot in  $Env$  is denoted by  $s_0 \in S_0$ . At each time step  $t$ , the agent observes the state  $s_t \in S$  and executes an action  $a_t \in A$ , according to the policy  $\pi(a_t | s_t)$ , and  $Env$  returns the next state  $s_{t+1}$  sampled from  $p_S(s_{t+1} | s_t, a_t)$ . The process is repeated until the episode is terminated. The objective of the robot is to learn an optimal policy  $\pi^*(a|s)$  that maximizes the expected discounted return  $J(\pi) = \mathbb{E}^\pi \left[ \sum_{k=0}^{\infty} \gamma^k \cdot R(s_k, a_k, s_{k+1}) \right]$  under the policy  $\pi$ .

*Linear Temporal Logic (LTL):* An LTL formula is built to describe high-level specifications of a system. Its ingredients are a set of atomic propositions, and combinations of Boolean and temporal operators. The syntax of LTL formulas is defined:

$$\phi := \text{true} \mid a \mid \phi_1 \wedge \phi_2 \mid \neg \phi_1 \mid \phi \mid \phi_1 \mathcal{U} \phi_2,$$

where  $a \in AP$  is an atomic proposition, *true*, *negation*  $\neg$ , and *conjunction*  $\wedge$  are propositional logic operators, and *next*  $\bigcirc$  and *until*  $\mathcal{U}$  are temporal operators. Alongside the standard operators introduced above, other propositional logic operators, such as *false*, *disjunction*  $\vee$ , and *implication*  $\rightarrow$ , and temporal operators, such as *always*  $\square$  and *eventually*  $\diamond$ , are derived from the standard operators.

For a infinite word  $o$  starting from the step 0, let  $o_t, t \in \mathbb{N}$  denotes the value at step  $t$ . The semantics of an LTL formula are interpreted over words, where a word is an infinite sequence  $o = o_0 o_1 \dots$ , with  $o_i \in 2^{AP}$  for all  $i \geq 0$ . The satisfaction of an LTL formula  $\phi$  by the word  $o$  is denote by  $o \models \phi$ . More details about LTL formulas can be found in [18].

In this work, we restrict our attention to LTL formulas that exclude the *next* temporal operator, which is not meaningful for continuous state-action space [32], [34].

### III. PROBLEM FORMULATION

Consider a cl-MDP  $\mathcal{M} = (S, S_0, A, p_S, AP, L, R, \gamma)$ . The induced path under a policy  $\pi = \pi_0\pi_1 \dots$  over  $\mathcal{M}$  is  $\mathbf{s}_\infty^\pi = s_0 \dots s_i s_{i+1} \dots$ , where  $p_S(s_{i+1}|s_i, a_i) > 0$  if  $\pi_i(a_i|s_i) > 0$ . Let  $L(\mathbf{s}_\infty^\pi) = o_1 o_2 \dots$  be the sequence of labels associated with  $\mathbf{s}_\infty^\pi$ , such that  $o_i = L(s_i), \forall i \in \{0, 1, 2, \dots\}$ . We denote the satisfaction relation of the induced trace with  $\phi$  by  $L(\mathbf{s}_\infty^\pi) \models \phi$ . The probability of satisfying  $\phi$  under the policy  $\pi$ , starting from an initial state  $s_0 \in S_0$ , is defined as

$$\Pr_M^\pi(\phi) = \Pr_M^\pi(L(\mathbf{s}_\infty^\pi) \models \phi \mid \mathbf{s}_\infty^\pi \in \mathcal{S}_\infty^\pi),$$

where  $\mathcal{S}_\infty^\pi$  is the set of admissible paths from the initial state  $s_0$ , under the policy  $\pi$ , and the detailed computation of  $\Pr_M^\pi(\phi)$  can be found in [18]. The transition distributions  $p_S$  of  $\mathcal{M}$  are unknown due to the unknown dynamic  $\mathcal{S}$ , and DRL algorithms are employed to learn the optimal control policies.

In this paper, the cl-MDP  $\mathcal{M}$  captures the interactions between a cluttered environment  $Env$  with geometric space  $X$ , and an unknown dynamic system  $\mathcal{S}$ . Note that explicitly constructing a cl-MDP  $\mathcal{M}$  is impossible, due to the continuous state-action space. We track any cl-MDP  $\mathcal{M}$  on-the-fly (abstraction-free) using deep neural network, according to the evolution of the dynamic system  $\mathcal{S}$  operating in  $Env$ .

*Problem 1:* Consider a set of labeled goal regions in  $Env$  i.e.,  $AP_G = \{\mathcal{G}_1, \mathcal{G}_2, \dots\}$ . The safety-critical specification is expressed as  $\phi = \square \neg \mathcal{O} \wedge \phi_g$ , where  $\mathcal{O}$  denotes the atomic proposition for obstacles. The expression  $\phi = \square \neg \mathcal{O} \wedge \phi_g$  requires the robot satisfying a general navigation task  $\phi_g$ , e.g., goal-reaching, while avoiding obstacles. The objective is to synthesize the optimal policy  $\pi^*$  of  $\mathcal{M}$  satisfying the task  $\phi$ , i.e.,  $\Pr_M^\pi(\phi) > 0$ .

*Assumption 1:* Let  $X_{free}$  denote the obstacle-free space. We assume that there exists at least one policy that drives the robot from initial states toward the regions of interest while always operating in  $X_{free}$ . This is reasonable since the assumption ensures the existence of policies satisfying a given valid LTL specification.

For Problem 1, typical learning-based algorithms for target-driven problems only assign positive rewards when the robot reaches any goal region  $X_{G_i}$  toward the LTL satisfaction, resulting exploration issues of DRL rendered from the environmental challenge. This point is obvious even when considering the special case of goal-reaching tasks, i.e.,  $\phi_p = \square \neg \mathcal{O} \wedge \diamond \mathcal{G}_i$ , where the sub-task  $\diamond \mathcal{G}_i$  requires to eventually visit the goal region  $X_{G_i}$  labeled as  $\mathcal{G}_i$ .

*Example 1:* Consider an autonomous vehicle as an RL-agent with unknown dynamics that is tasked with specification  $\phi$ , shown in Fig. 1(a). For a goal-reaching task as a special LTL formula  $\phi_p = \square \neg \mathcal{O} \wedge \diamond \mathcal{G}_1$ , if the RL-agent only receives a reward after reaching the goal region  $\mathcal{G}_1$ , it will be hard to effectively explore using data with positive rewards task and noisy policies. The problem becomes more challenging for specifications such as  $\phi_{ex} = \square \neg \mathcal{O} \wedge \phi_{g,ex} = \square \neg \mathcal{O} \wedge \square(\diamond \mathcal{G}_1 \wedge \diamond(\mathcal{G}_2 \wedge \diamond \dots \wedge \diamond \mathcal{G}_4))$  that require the robot to visit regions  $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4$  sequentially infinitely many times.

In Section IV, we show how to learn the control policy of completing a standard goal-reaching mission  $\phi_p = \square \neg \mathcal{O} \wedge \diamond \mathcal{G}_i$  in cluttered environments. Then, Section V builds upon and extends the approach to solve a general safety-critical navigation task  $\phi = \square \neg \mathcal{O} \wedge \phi_g$  in a distributed manner.

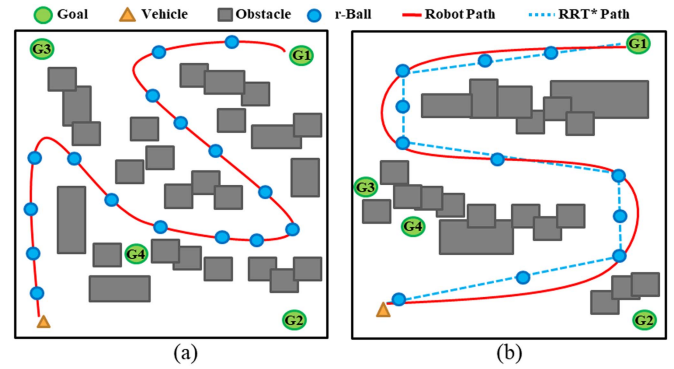


Fig. 1. Consider an autonomous vehicle operating in a cluttered and large-scale environment. It is challenging to learn optimal policies as described in Example 1. The trajectories provide insights of reward design in Section IV.

### IV. OVERCOMING EXPLORATION

In Section IV-A, we briefly introduce the geometric sampling-based algorithm to generate an optimal path for the standard goal-reaching tasks  $\phi_p = \square \neg \mathcal{O} \wedge \diamond \mathcal{G}_i$ . Subsequently, Section IV-B develops a novel dense reward to overcome the exploration challenges and provides rigorous analysis for learning performance.

#### A. Geometric RRT\*

The standard optimal RRT\* method [13] is a variant of RRT [12]. Both RRT and RRT\* are able to handle the path planning in cluttered and large-scale environments. Due to its optimality, we choose RRT\* over RRT, to improve the learned performance of the optimal policies. Since the dynamic system  $\mathcal{S}$  is unknown, we use the geometric RRT\* method that builds a tree  $G = (V, E)$  incrementally in  $X$ , where  $V$  is the set of vertices and  $E$  is the set of edges. If  $V$  intersects the goal region, we find a geometric trajectory to complete the task  $\phi$ . The detailed procedure of geometric RRT\* is described in the Appendix A of [35]. Here, we briefly introduce two of the functions in the geometric RRT\* method, which are used in explaining our method in the next sections.

*Distance and cost:* The function  $dist : X \times X \rightarrow [0, \infty)$  is the metric that computes the geometric Euclidean distance between two states. The function  $Cost : X \rightarrow [0, \infty)$  returns the length of the path in the tree between two input states.

*Steering:* Given two states  $x$  and  $x'$ , the function *Steer* returns a state  $x_{new}$  such that  $x_{new}$  lies on the geometric line connecting  $x$  to  $x'$ , and its distance from  $x$  is at most  $\eta$ , i.e.,  $dist(x, x') \leq \eta$ , where  $\eta$  is a user-specified parameter. In addition, the state  $x_{new}$  must satisfy  $dist(x_{new}, x') \leq dist(x, x')$ .

Having the tree  $G = (V, E)$  generated by the RRT\* method, if there exists at least one node  $x \in V$  that is located within the goal regions, we find the optimal state trajectory satisfying the task  $\phi$  as a sequence of geometric states  $\mathbf{x}^* = x_0 x_1 \dots x_{N_p}$ , where  $x_{N_p} \in X_G$  and  $x_i \in V, \forall i = 0, 1, \dots, N_p$ .

#### B. Sampling-Based Reward

Here we use the optimal geometric trajectory  $\mathbf{x}^*$  and the properties of the generated tree  $G = (V, E)$  to synthesize the reward scheme. First, let  $x|\mathbf{x}^*$  denote a state of  $\mathbf{x}^*$ , and the total length of  $\mathbf{x}^*$  in the tree  $G$  be equal to  $Cost(x_{N_p}|\mathbf{x}^*)$ . We



define the distance from each state  $x \in \mathbf{x}^*$  to the destination  $x_{N_p}$  as  $Dist(x|\mathbf{x}^*) = Cost(x_{N_p}|\mathbf{x}^*) - Cost(x|\mathbf{x}^*)$ . Based on the distance, we design the RRT\* reward scheme to guide the learning progress towards the satisfaction of  $\phi$ . Reaching an exact state in the continuous state space is challenging for robots. Thus, we define the norm  $r$ -ball for each state  $x|\mathbf{x}^*$  to allow the robot to visit the neighboring area of the state as  $Ball_r(x|\mathbf{x}^*) = \{x' \in X \mid dist(x|\mathbf{x}^*, x') \leq r\}$ , where  $x|\mathbf{x}^*$  is the center and  $r$  is the radius. For simplicity, we select  $r \leq \frac{\eta}{2}$  based on the steering function of the geometric RRT\*, such that the adjacent  $r$ -balls along the optimal trajectory  $\mathbf{x}^*$  do not intersect with each other.

We develop a progression function  $D : X \rightarrow [0, \infty)$  to track whether the current state is getting closer to the goal region, by following the sequence of balls along  $\mathbf{x}^*$  as:

$$D(x) = \begin{cases} Dist(x|\mathbf{x}^*) & \text{if } x \in Ball_r(x_i|\mathbf{x}^*) \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

For the cl-MDP  $\mathcal{M}$  capturing the interactions between  $\mathcal{S}$  and  $Env$ , the intuition behind the sampling-based reward design is to assign a positive reward whenever the robot gets geometrically closer to the goal region, along the optimal path obtained by the RRT\* (Alg.2 of [35]).

During each episode of learning, a state-action sequence  $s_0 a_0 s_1 a_1 \dots s_t$  up to current time  $t$  is projected into the state and action sequences  $\mathbf{s}_t = s_0 s_1 \dots s_t$  and  $\mathbf{a}_t = a_0 a_1 \dots a_{t-1}$ , respectively. We define

$$D_{min}(\mathbf{s}_t) = \min_{s \in \mathbf{s}_t} \{D(Proj(s))\}$$

as the progression energy that is equal to the minimum distance along the optimal path towards the destination, up to step  $t$ . The objective of the reward is to drive the robot such that  $D_{min}(\mathbf{s}_t)$  decreases. However, employing the function  $D_{min}(\mathbf{s}_t)$  for reward design that depends on the history of the trajectory results in a non-Markovian reward function [19], while the policy  $\pi(s)$  only takes the current state as input and can not distinguish the progress achieved by the histories  $\mathbf{s}_t$ .

To address the issue, inspired by the product MPD [18], given the history  $\mathbf{s}_t$ , we keep tracking the index  $i_t \in \{0, 1, \dots, N_p\}$  of the center state of the visited  $r$ -ball regions  $Ball_r(x_{i_t}|\mathbf{x}^*)$  with minimum distance  $Dist(x_{i_t}|\mathbf{x}^*)$  deterministically, i.e.,

$$x_{i_t} = Proj(s_{i_t}), \text{ where } s_{i_t} = \arg \min_{s \in \mathbf{s}_t} \{D(Proj(s))\}$$

If none of the  $r$ -balls are visited up to  $t$ , we set  $i_t = 0$ . Then, the current state  $s_t$  is embedded with the index  $i_t$  as a product state  $s_t^\times = (s_t, i_t)$ , which is considered as the input of the policy, i.e.,  $\pi(s_t^\times)$ . Note that we treat the embedded component  $i_t$  as the state of a deterministic automaton [18]. The relevant analysis can be found in Appendix B of [35].

Let  $R : s^\times \rightarrow \mathbb{R}$  denote the episodic reward function. We propose a novel scheme to assign the Markovian reward with respect to the product state  $s_t^\times$  as:

$$R(s_t^\times) = \begin{cases} R_-, & \text{if } Proj(s_t) \in X_{\mathcal{O}}, \\ R_{++}, & \text{if } D(Proj(s_t)) = 0, \\ R_+, & \text{if } D(Proj(s_t)) < D_{min}(\mathbf{s}_{t-1}), \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

where  $R_+$  is a positive constant reward,  $R_{++}$  is a boosted positive constant that is awarded when the robot reaches the destination, and  $R_-$  is the negative constant reward that is

assigned when the robot violates the safety task of  $\phi$ , i.e.,  $\phi_{safe} = \square \neg \mathcal{O}$ . Note that if the robot crosses both obstacles and  $r$ -balls, it receives the negative reward  $R_-$ , which has first priority. This setting does not restrict selections of the parameter  $r$  (radius of  $r$ -balls) for implementations.

*Example 2:* As shown in Fig. 1(a), we apply the RRT\* method to generate the optimal trajectory (shown in red) in a challenging environment. Then, we construct the sequence of  $r$ -balls along it and apply the reward design (3) to guide the learning and overcome exploration difficulties.

*Remark 1:* Since geometric RRT\* does not consider the dynamic system, the optimal path  $\mathbf{x}^*$  may be infeasible for the robot to follow exactly, with respect to any policy. As a running example in Fig. 1(b), our RRT\* reward is robust such that the robot is not required to strictly follow all  $r$ -balls of the optimal path. Instead, in order to receive the positive reward, the robot only needs to move towards the destination and pass through the partial  $r$ -balls  $Ball_r(x_i|\mathbf{x}^*)$ ,  $i \in \{0, 1, \dots, N_p\}$  along the optimal path. If we want the robot to reach the goal with desired orientations, we can add another reward that measures the errors between the robot's actual and desired orientations.

By applying the reward design (3), we formally verify the performance of the reward (3) for the reach-avoid task  $\phi_p$ .

*Theorem 1:* If Assumption 1 holds, by selecting  $R_{++}$  to be sufficiently larger than  $R_+$ , i.e.,  $R_{++} \gg R_+$ , any algorithm that optimizes the expected return  $J(\pi)$  is guaranteed to find the optimal policy  $\pi^*$  satisfying the goal-reaching task  $\phi_p$ , i.e.,  $\Pr_M^{\pi^*}(\phi_p) > 0$ .

The proof is presented in Appendix C of [35]. Theorem 1 provides a theoretical guarantee for the optimization performance, allowing us to apply practical algorithms to find the approximated optimal policy in continuous space.

Based on Theorem 1 and regarding the continuous control task, we apply advanced DRL methods, e.g., actor-critic algorithms [4], [5], [7], to find the optimal policy  $\pi^*$ . Consider a policy  $\pi_\theta(a|s^\times)$ , parameterized by  $\theta$ , the learning objective aims to find the optimal policy via optimizing the parameters  $\theta$  and maximizing the expected discount return  $J(\theta) = \mathbb{E}^{\pi_\theta} \left[ \sum_{[k=0]}^{\infty} \gamma^k \cdot R(s_t^\times) \right]$ , which minimizes the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s^\times, a, r, s'^\times) \sim \mathcal{D}} \left[ (Q(s, a|\omega) - y)^2 \right], \quad (4)$$

where  $\mathcal{D}$  is the replay buffer that stores experience tuples  $(s^\times, a, R, s'^\times)$ ,  $Q(s, a|\omega)$  is the state-action valuation function parameterized by  $\omega$ , and  $y = r + \gamma Q(s'^\times, a'|\omega)$ . As observed in (4), actor-critics rely on effective data in the replay buffer, or sample efficiency of the state distribution to minimize the loss function. Due to its high reward density over geometric space, the sampling-based reward is easy to explore and improves the training performance using noisy policies.

*Theorem 2:* If Assumption 1 holds, by selecting  $R_{++}$  to be sufficiently larger than  $R_+$ , i.e.,  $R_{++} \gg R_+$ , a suitable DPG algorithm that optimizes the expected return  $J(\theta)$ , finds the optimal parameterized policy  $\pi_\theta^*$  satisfying the LTL tasks  $\phi_p$ , i.e.,  $\Pr_M^{\pi_\theta^*}(\phi_p) > 0$  in the limit.

Theorem 2 is an immediate result of Theorem 1 and the nature of nonlinear regressions in deep neural networks. In practice, the number of episodes and steps are limited and training has to be stopped eventually.

## V. LTL TASK SATISFACTION

Section V-A describes how to generate and decompose the optimal path of satisfying general LTL task  $\phi$  in a sequence of paths of completing goal-reaching missions  $\phi_p$ , and Section V-B explains how to integrate distributed DPGs with the novel exploration guidance of Section IV, to learn the optimal policy.

### A. Geometric TL-RRT\*

Due to the unknown dynamic system, we define the transition system over the geometric space  $X$ , referred as Geometric-Weighted Transition System (G-WTS).

*Definition 1:* A G-WTS of  $Env$  is a tuple  $\mathcal{T} = (X, x_0, \rightarrow_{\mathcal{T}}, AP, L_X, C_{\mathcal{T}})$ , where  $X$  is the geometric space of  $Env$ ,  $x_0$  is the initial state of robot;  $\rightarrow_{\mathcal{T}} \subseteq X \times X$  is the geometric transition relation s.t.  $x \rightarrow_{\mathcal{T}} x'$  if  $dist(x, x') \leq \eta$  and the straight line  $\sigma$  connecting  $x$  to  $x_{new}$  is collision-free,  $AP$  is the set of atomic propositions as the labels of regions,  $L_X : X \rightarrow AP$  is the labelling function that returns an atomic proposition satisfied at a location  $x$ , and  $C_{\mathcal{T}} : (\rightarrow_{\mathcal{T}}) \rightarrow \mathbb{R}^+$  is the geometric Euclidean distance, i.e.,  $C_{\mathcal{T}}(x, x') = dist(x, x'), \forall (x, x') \in \rightarrow_{\mathcal{T}}$ .

The standard WTS [32], [34] defines the transition relations  $x \rightarrow_{\mathcal{T}} x'$  according to the existence of model-based controllers that drive the robot between neighbor regions  $x, x'$ . Differently, we only consider the geometric connection among states in a model-free manner.

Let  $\tau_{\mathcal{T}} = x_0 x_1 x_2 \dots$  denote a valid run of  $\mathcal{T}$ . An LTL formula  $\phi$  can be converted into a Non-deterministic Büchi Automata (NBA) to verify its satisfaction.

*Definition 2:* [36] An NBA over  $2^{AP}$  is a tuple  $\mathcal{B} = (Q, Q_0, \Sigma, \rightarrow_{\mathcal{B}}, Q_F)$ , where  $Q$  is the set of states,  $Q_0 \subseteq Q$  is the set of initial states,  $\Sigma = 2^{AP}$  is the finite alphabet,  $\rightarrow_{\mathcal{B}} \subseteq Q \times \Sigma \times Q$  is the transition relation, and  $Q_F \subseteq Q$  is the set of accepting states.

A valid infinite run  $\tau_{\mathcal{B}} = q_0 q_1 q_2 \dots$  of  $\mathcal{B}$  is called accepting, if it intersects with  $Q_F$  infinite often. Infinite words  $\tau_o = o_0 o_1 o_2 \dots, \forall o \in 2^{AP}$  generated from an accepting run satisfy the corresponding LTL formula  $\phi$ . An LTL formula is converted into NBA using the tool [37]. As in [30], we prune the infeasible transitions of the resulting NBA to obtain the truncated NBA.

*Definition 3:* Given the G-WTS  $\mathcal{T}$  and the NBA  $\mathcal{B}$ , the product Büchi automaton (PBA) is a tuple  $P = \mathcal{T} \times \mathcal{B} = (Q_P, Q_P^0, \rightarrow_P, Q_P^F, C_P, L_P)$ , where  $Q_P = X \times Q$  is the set of infinite product states,  $Q_P^0 = x_0 \times Q_0$  is the set of initial states;  $\rightarrow_P \subseteq Q_P \times 2^{AP} \times Q_P$  is the transition relation defined by the rule:  $\frac{x \rightarrow_{\mathcal{T}} x' \wedge q \xrightarrow{L_X(x)} q'}{q_P = (x, q) \rightarrow_P q'_P = (x', q')}$ , where  $q_P \rightarrow_P q'_P$  denotes the transition  $(q_P, q'_P) \in \rightarrow_P$ ,  $Q_P^F = X \times Q_F$  is the set of accepting states,  $C_P : (\rightarrow_P) \rightarrow \mathbb{R}^+$  is the cost function defined as the cost in the geometric space, e.g.,  $C_P(q_P = (x, q), q'_P = (x', q')) = C_{\mathcal{T}}(x, x'), \forall (q_P, q'_P) \in \rightarrow_P$ , and  $L_P : Q_P \rightarrow AP$  is the labelling function s.t.  $L_P(q_P) = L_X(x), \forall q_P = (x, q)$ .

A valid trace  $\tau_P = q_P^0 q_P^1 q_P^2 \dots$  of a PBA is called accepting, if it visits  $Q_P^F$  infinitely often, referred as the acceptance condition. Its accepting words  $\tau_o = o_0 o_1 o_2 \dots, \forall o_i = L_P(q_P^i)$  satisfy the corresponding LTL formula  $\phi$ . Let  $\tau_F$  denote an accepting trace, and  $proj|_X : Q_P \rightarrow X$  is a function that projects product state space into the workspace, i.e.,  $proj|_X(q_P) = x, \forall q_P = (x, q)$ . Using the projection, we extract the geometric trajectory  $\tau_{\mathcal{T}} = proj|_X(\tau_F)$  that satisfies the LTL formula. More details are presented in [18]. Therefore, the planning objective is to

---

### Algorithm 1: LTL-RRT\*-Distributed DPGs.

---

- 1: **Input:**  $Env, \phi = \square \neg \mathcal{O} \wedge \phi_g$ , Black-box  $\mathcal{S}$ ;
  - 2: **Initialize:** Geometric space  $X$ , Primitives of TL-RRT\*;
  - 3: Convert  $\phi$  into NBA  $\mathcal{B}$
  - 4: Build the incremental trees for PBA geometrically, based on definition 1 and definition 3
  - 5: Generate the optimal trajectory  $\tau_F^* = \tau_{pre}^* [\tau_{suf}^*]^{\omega}$
  - 6: Reformulate the trajectory into the modular form
- $$\mathcal{R}_F = (\mathcal{R}_0 \mathcal{R}_1 \dots \mathcal{R}_K) (\mathcal{R}_{K+1} \dots \mathcal{R}_{K+l})^{\omega}$$
- 7: **for**  $i = 1, \dots, K + l$  **do**
  - 8: Construct the RRT\* reward based on (3) for  $\mathcal{R}_i$
  - 9: Assign an actor-critic DPG e.g., DDPG, PPO, for  $\mathcal{R}_i$
  - 10: **end for**
  - 11: Assign the rewards (3) and DPGs for each  $\mathcal{R}_i$ .
  - 12: Train the distributed DPGs in parallel
  - 13: Extract the optimal policy  $\pi_i^*$  from each DPG  $\mathcal{R}_i$
  - 14: Concatenate all optimal policies in the form

$$\pi_{\theta}^* = (\pi_0^* \pi_1^* \dots \pi_K^*) (\pi_{K+1}^* \dots \pi_{K+l}^*)^{\omega}$$


---

find an acceptable path  $\tau_P$  of PBA, with minimum cumulative geometric cost  $C_P$ .

However, the state space of G-WTS and PBA are both infinite. Consequently, we are not able to apply a graph search method to a PBA with infinite states. Thanks to the TL-RRT\* algorithm [32] for providing an abstraction-free method, it allows us to incrementally build trees that explore the product state-space and find the feasible optimal accepting path. The procedure applies the sampling-based method over the PBA, and is inspired by the fact that the accepting run  $\tau_F$  admits a lasso-type sequence in the form of prefix-suffix structure, i.e.,  $\tau_F = \tau_P^{pre} [\tau_P^{suf}]^{\omega}$ , where the prefix part  $\tau_P^{pre} = q_P^0 q_P^1 \dots q_P^K$  is only executed once, and the suffix part  $\tau_P^{suf} = q_P^K q_P^{K+1} \dots q_P^{K+M}$  with  $q_P^K = q_P^{K+M}$  is executed infinitely often.

Following this idea, we build the trees for the prefix and suffix paths, respectively. To satisfy the acceptance condition, the set of goal states of the prefix tree  $G_P^{pre} = (V_P^{pre}, E_P^{pre})$  is defined as  $Q_{goal}^{pre} = \{q_P = (x, q) \in X_{free} \times Q \subseteq Q_P \mid q \in Q_F\}$ , where  $X_{free}$  is the collision-free geometric space. After obtaining the prefix tree, we construct the set  $Q_{goal}^* = V_P^{pre} \cap Q_{goal}^{pre}$ , and compute the optimal prefix path  $\tau_{pre}^*$  reaching a state  $q_P^* \in Q_{goal}^*$  from the root  $q_P^0$ . The suffix tree  $G_P^{suf} = (V_P^{suf}, E_P^{suf})$  is built by treating  $q_P^* = (x^*, q^*)$  as the root, and its goal states are:

$$Q_{goal}^{suf}(q_P^*) = \left\{ q_P = (x, q) \in X_{free} \times Q \subseteq Q_P \mid x \rightarrow_{\mathcal{T}} x^* \wedge q \xrightarrow{L_X(x)} q^* \right\}.$$

$Q_{goal}^{suf}(q_P^*)$  collects all states that can reach the state  $q_P^*$  via one transition, and this way it ensures the feasible cyclic path matching the suffix structure. Finally, we search the optimal suffix path  $\tau_{suf}^*$ , by constructing  $V_P^{suf} \cap Q_{goal}^{suf}$ .

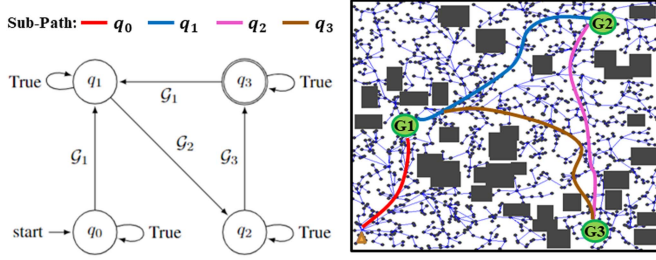


Fig. 2. Decomposition example. (Left) Truncated NBA  $\mathcal{B}$  of the LTL formula  $\phi_{g_1} = \square \diamond \mathcal{G}_1 \wedge \square \diamond \mathcal{G}_2 \wedge \square \diamond \mathcal{G}_3$  for  $\phi_{1,inf} = \square -\mathcal{O} \wedge \phi_{g_1}$ ; (Right) Decomposed sub-goal-reaching tasks.

## B. Distributed DPGs

In this section, we first employ the optimal geometric path  $\tau_F^* = \tau_{pre}^*[\tau_{suf}^*]^\omega$  from Section IV-A, to propose a distributed reward scheme. Since the policy gradient strategy suffers from the variance issue and only finds the optimal policy in the limit (see Theorem 2), instead of directly applying the reward design (3) for the whole path  $\tau_F^*$ , we decompose it into sub-tasks. To do so, we divide  $\tau_F^*$  into separated consecutive segments, each of which shares the same automaton components, i.e.,  $\tau_F^* = \tau_0^* \tau_1^* \dots \tau_K^* [\tau_{K+1}^* \dots \tau_{K+l}^*]^\omega$  such that all states of each sub trajectory  $\tau_i^*$  have the same automaton components. Each segment can be treated as a collision-free goal-reaching problem, denoted as  $\mathcal{R}_i(\mathcal{G}_i, \mathcal{O})$ , where  $\mathcal{G}_i$  is label of the  $i^{th}$  goal region. Specifically, suppose the state trajectory of each  $\mathcal{R}_i(\mathcal{G}_i, \mathcal{O})$  is  $\tau_i^* = q_{P,i}^0 q_{P,i}^1 \dots q_{P,i}^{N_i}$ , we select the region labeled as  $L_P(q_{P,i}^{N_i})$  containing the geometric state  $proj|_X(q_{P,i}^{N_i})$ .

We show an example of the optimal decomposition in Fig. 2, where the LTL task  $\phi_{1,inf} = \square -\mathcal{O} \wedge \phi_{g_1}$  over infinite horizons with  $\phi_{g_1} = \square \diamond \mathcal{G}_1 \wedge \square \diamond \mathcal{G}_2 \wedge \square \diamond \mathcal{G}_3$  that requires to infinitely visit goal regions labeled as  $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ . The resulting truncated NBA and decomposed trajectories of TL-RRT\* are shown in Fig. 2(a) and 2(b), respectively, where decomposed sub-paths are expressed as  $\mathcal{R}_F = \mathcal{R}_{red}(\mathcal{R}_{blue} \mathcal{R}_{pink} \mathcal{R}_{brown})^\omega$ , such that the distributed DPGs are applied to train the optimal sub-policies for each one in parallel.

The lasso-type optimal trajectory is reformulated as:  $\mathcal{R}_F = (\mathcal{R}_0 \mathcal{R}_1 \dots \mathcal{R}_K)(\mathcal{R}_{K+1} \dots \mathcal{R}_{K+l})^\omega$ . For the cl-MDP  $\mathcal{M}$ , we treat each  $\mathcal{R}_i$  as a task  $\phi_{\mathcal{R}_i} = \square -\mathcal{O} \wedge \diamond \mathcal{G}_i$  solved in Section IV-B. In particular, we propose collaborative team of RRT\* rewards in (3) for each sub-task and assign distributed DPGs for each  $\mathcal{R}_i$  that are trained in parallel. After training, we extract the concatenate policy  $\pi_i^*$  of each  $\mathcal{R}_i$  to obtain the global optimal policy as  $\pi_\theta^* = (\pi_0^* \pi_1^* \dots \pi_K^*)(\pi_{K+1}^* \dots \pi_{K+l}^*)^\omega$ . The overall procedure is summarized in Algorithm 1, and a detailed diagram with rigorous analysis is presented in Appendix C of [35]. Based on the decomposition properties and Theorem 2, we can conclude that the concatenated optimal policy of Algorithm 1 satisfies the global LTL specification.

## VI. EXPERIMENTAL RESULTS

We evaluate the framework on different nonlinear dynamic systems tasked to satisfy various LTL specifications. The tests focus on large-scale cluttered environments that generalize the simple ones to demonstrate the method's performance. Obstacles are randomly sampled. We integrate all baselines with

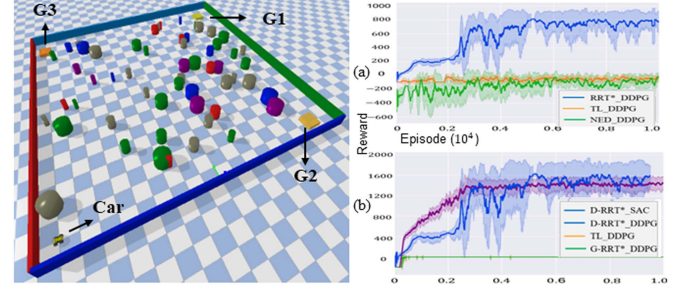


Fig. 3. Baselines comparison of tasks  $\phi_{2,fin}$  (a) and  $\phi_{2,inf}$  (b) in the Pybullet environment.

either DDPG or SAC as DPG algorithms. Finally, we show that our algorithm improves the success rates of task satisfaction over both infinite and finite horizons in cluttered environments, and significantly reduces training time for the task over finite horizons. Detailed descriptions of environments and LTL tasks will be introduced.

Recall that  $\diamond$  denotes the *eventually* operator used to specify feasibility properties (e.g., goal-reaching), while  $\square$  stand for the *always* operator used for safety (e.g., collision avoidance) and invariance (e.g., geo-fencing) properties.

**Baseline Approaches:** We refer to our distributed framework as “RRT\*” or “D-RRT\*,” and compare it against three baselines: (i) The TL-based rewards in [25], [27] referred as “TL,” for the single LTL task, have shown excellent performance in non-cluttered environments, which generalizes the cases of finite horizons in existing literature [22], [23], [24]; (ii) Similar as [38], [39], 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 apply the reward scheme (3) for the global trajectory  $\tau_F^* = \tau_{pre}^*[\tau_{suf}^*]^\omega$  referred as “G-RRT\*.” Note that we focus on comparing the baseline “NED” for finite-horizon tasks and the baseline “G-RRT\*” for infinite-horizon and complex tasks.

**6.1 Autonomous Vehicle:** We first implement the car-like model of Pybullet<sup>1</sup> physical engine shown in Fig. 3. We consider the sequential LTL task and surveillance LTL task over both finite and infinite horizons as  $\phi_{2,fin} = \square -\mathcal{O} \wedge \diamond (\mathcal{G}_1 \wedge \diamond (\mathcal{G}_2 \wedge \diamond (\mathcal{G}_3 \wedge \diamond \mathcal{G}_{init})))$  and  $\phi_{2,inf} = \square -\mathcal{O} \wedge \square \diamond (\mathcal{G}_1 \wedge \diamond (\mathcal{G}_2 \wedge \diamond (\mathcal{G}_3)))$ , respectively, where  $\phi_{2,fin}$  requires the vehicle to visit goal regions labeled as  $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$  and initial position sequentially, and  $\phi_{1,inf}$  requires to visit initial positions and other goal regions infinitely often. Fig. 3(a) and (b) show the learning curves of task  $\phi_{2,fin}$  and  $\phi_{2,inf}$ , respectively, compared with different baselines. We can observe that our framework can be adopted in both DDPG and SAC to provide better performance than other baselines in cluttered environments.

**6.2 Quadrotor Model:** We implement our algorithms in a 3D environment with Quadrotor<sup>2</sup> dynamics shown in Fig. 4, which shows the capability of handling cluttered environments and high dimensional systems. We also test two types of LTL specifications as  $\phi_{1,fin} = \square -\mathcal{O} \wedge \diamond \mathcal{G}_1 \wedge \diamond \mathcal{G}_2 \wedge \diamond \mathcal{G}_3$  and  $\phi_{1,inf} =$

<sup>1</sup>[Online]. Available: <https://pybullet.org/wordpress/>

<sup>2</sup>[Online]. Available: <https://github.com/Bharath2/Quadrotor-Simulation/tree/main/PathPlanning>



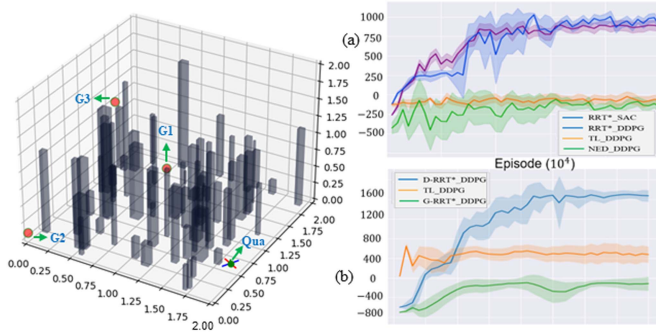


Fig. 4. Baselines comparison of tasks  $\phi_{1,fin}$  in (a) and  $\phi_{1,inf}$  in (b) of the cluttered 3D quadrotor environment.

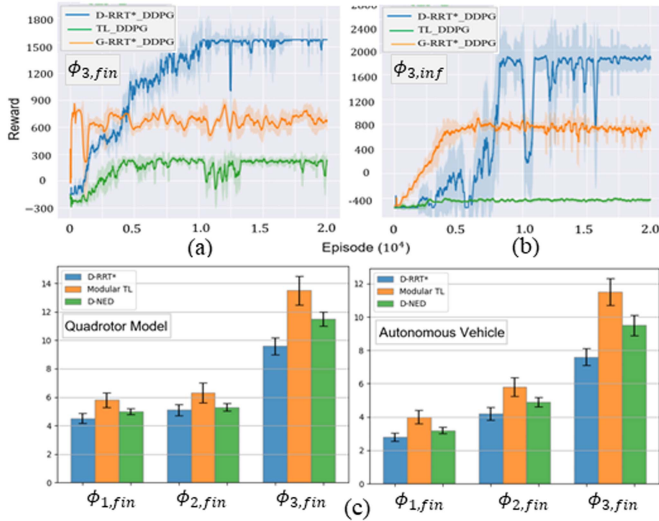


Fig. 5. (a), (b) Results of baselines for more complex task  $\phi_{3,fin}$  and  $\phi_{3,inf}$  in cluttered 3D environments; (c) Training time comparison for all tasks over finite horizons in both environments and dynamic systems.

$\square \rightarrow \mathcal{O} \wedge \square \diamond \mathcal{G}_1 \wedge \square \diamond \mathcal{G}_2 \wedge \square \diamond \mathcal{G}_3$ . The learning results for these tasks are shown in Fig 4(a) and (b), respectively.

Then, we increase the complexity by random sampling 12 obstacle-free goal regions in the 3D environment and set the rich specifications as  $\phi_{3,fin} = \square \rightarrow \mathcal{O} \wedge ((\diamond \mathcal{G}_1 \wedge \diamond (\mathcal{G}_2 \wedge \diamond \dots \wedge \diamond \mathcal{G}_{12}))$ , and  $\phi_{3,inf} = \square \rightarrow \mathcal{O} \wedge \square \diamond \mathcal{G}_1 \wedge \square \diamond \mathcal{G}_2 \dots \wedge \square \diamond \mathcal{G}_{12}$ . The results are shown in Fig. 5(a) and (b), and we observe that the “TL” baseline is sensitive to the environments and has poor performances, and when the optimal trajectories become complicated in the sense of the complexity of LTL tasks, “G-RRT\*” easily converges to the sub-optimal solutions.

**6.3 Performance Evaluation:** Since our algorithm learns to complete the task faster, it terminates each episode during learning earlier for tasks over finite horizons. To illustrate the efficiency, we implement the training process 10 times for tasks  $\phi$ ,  $\phi_{1,fin}$ ,  $\phi_{2,fin}$ ,  $\phi_{3,fin}$  in both cluttered environments and dynamics, and record the average training time compared with the baseline modular “TL” [25] and distributed “NED” (D-NED). The results in Fig. 5(d) show that we have optimized the learning efficiency and the training time is reduced. In practice, we can apply distributed computing to train each local DPG of sub-tasks

TABLE I  
ANALYSIS OF SUCCESS RATES AND TRAINING TIME

LTL Task	Dynamic model	Baseline rate	Success rate
$\phi_{1,fin}$	Quadrotor	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{2,fin}$	Vehicle	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{3,fin}$	Quadrotor	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{3,fin}$	Vehicle	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{1,inf}$	Quadrotor	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{2,inf}$	Vehicle	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{3,inf}$	Quadrotor	D-RRT*	100%
		TL, G-RRT*	0%
$\phi_{3,inf}$	Vehicle	D-RRT*	100%
		TL, G-RRT*	0%

simultaneously for complicated global tasks to alleviate the training burden.

We statistically run 200 trials of the learned policies for each sub-task and record the average success rates and training time of both models, i.e., autonomous vehicle and quadrotor. The results are shown in Table I. We see that in cluttered environments, success rates of other all baselines are 0, and our method achieves success rates of near 100%. As a result, the effectiveness of the performance has been significantly improved under environmental challenges.

## VII. CONCLUSION

Applying DPG algorithms to cluttered environments produces vastly different behaviors and results in failure to complete complex tasks. A persistent problem is the exploration phase of the learning process and the density of reward designs that limit its applications to real-world robotic systems. This paper provides a novel path-planning-based reward scheme to alleviate this problem, enabling significant improvement of reward performance and generating optimal policies satisfying complex specifications in cluttered environments. To facilitate rich high-level specification, we develop an optimal decomposition strategy for the global LTL task, allowing to train all sub-tasks in parallel and optimize the efficiency. The main limitation of our approach is to generalize various environments from a distribution. Future works aim at shrinking the gap of sim-to-real. We will also consider safety-critical exploration during learning and investigate multi-agent systems.

## REFERENCES

- [1] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
- [3] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
- [4] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in *Proc. Int. Conf. Learn. Representation*, 2016.
- [5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *OpenAI*, 2017.

- [6] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.
- [7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
- [8] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 3223–3230.
- [9] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6292–6299.
- [10] M. Vecerik et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, *arXiv:1707.08817*.
- [11] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2052–2062.
- [12] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
- [13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
- [14] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23, no. 7/8, pp. 673–692, 2004.
- [15] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, 2003.
- [16] I. R. Manchester and J.-J. E. Slotine, "Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 3046–3053, Jun. 2017.
- [17] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Proc. Conf. Robot Learn.*, 2022, pp. 1724–1735.
- [18] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
- [19] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2107–2116.
- [20] M. Hasanbeig, N. Y. Jeppu, A. Abate, T. Melham, and D. Kroening, "DeepSynth: Program synthesis for automatic task segmentation in deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019.
- [21] Z. Xu et al., "Joint inference of reward machines and policies for reinforcement learning," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2020, vol. 30, pp. 590–598.
- [22] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Sci. Robot.*, vol. 4, no. 37, 2019.
- [23] P. Vaezipoor, A. Li, R. T. Icarte, and S. McIlraith, "LTL2Action: Generalizing LTL instructions for multi-task RL," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10497–10508.
- [24] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Reward machines: Exploiting reward function structure in reinforcement learning," *J. Artif. Intell. Res.*, vol. 73, pp. 173–208, 2022.
- [25] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7973–7980, Oct. 2021.
- [26] M. Cai and C.-I. Vasile, "Safety-critical modular deep reinforcement learning with temporal logic through Gaussian processes and control barrier functions," 2021, *arXiv:2109.02791*.
- [27] M. Hasanbeig, D. Kroening, and A. Abate, "Deep reinforcement learning with temporal logics," in *Proc. Int. Conf. Formal Model. Anal. Timed Syst.*, Springer, 2020, pp. 1–22.
- [28] S. Sickert, J. Esparza, S. Jaax, and J. Křetínský, "Limit-deterministic Büchi automata for linear temporal logic," in *Proc. Int. Conf. Comput. Aided Verif.*, Springer, 2016, pp. 312–332.
- [29] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Robot. Res.*, vol. 39, no. 8, pp. 1002–1028, 2020.
- [30] Y. Kantaros and M. M. Zavlanos, "STyLuS\*: A Temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 812–836, 2020.
- [31] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 363–374, Apr. 2021.
- [32] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multirobot temporal logic optimal control synthesis," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1487–1507, Oct. 2021.
- [33] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.
- [34] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, Feb. 2008.
- [35] 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," 2022, *arXiv:2201.12231*.
- [36] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *Proc. First Symp. Log. Comput. Sci.* 1986, pp. 322–331.
- [37] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *Proc. Int. Conf. Comput. Aided Verification*, Springer, 2001, pp. 53–65.
- [38] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.
- [39] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 1904–1911, Apr. 2022.