Poster Abstract: Signal Temporal Logic Compliant Motion Planning using Reinforcement Learning

Tushar Dilip Kurne[‡] tusharkurne@iisc.ac.in
Manas Sashank Juvvi[‡] manasjuvvi@iisc.ac.in
Vaishnavi J[‡] vaishnavij@iisc.ac.in
Pushpak Jagtap pushpak@iisc.ac.in

***Abstract*—This work proposes a novel approach for model-free motion planning in autonomous robots, with a specific emphasis on handling signal temporal logic (STL) tasks. The approach is structured into two main phases: first, we learn spatio-temporal motion primitives to encapsulate the inherent robot-specific constraints effectively. Subsequently, by utilizing these learned motion primitives, we conduct STL-compliant motion planning. To learn spatiotemporal motion primitives, we leverage reinforcement learning to construct a library of simple motion primitives. Following this, Gaussian process regression is employed to establish mappings from learned motion primitives to spatio-temporal characteristics. Subsequently, we present an STL-compliant motion planning strategy designed to meet the STL specification. Notably, the proposed framework is entirely model-free and capable of generating feasible STL-compliant motion plans across a diverse range of environments.**

***Index Terms*—Signal Temporal Logic, Path Planning, RL**

I. Introduction and Problem Definition

Signal-Temporal Logic (STL) offers a robust framework for formulating intricate tasks encompassing spatial, temporal, and logical constraints. One can formally and efficiently capture complex motion planning tasks by leveraging STL specifications. Moreover, STL representation presents a comprehensive approach for quantitatively assessing system adherence to STL formulas through various robustness metrics. Recently, researchers [1], [2] have implemented sampling-based methods for solving STL-compliant motion planning problems by considering the robustness of the STL signal in the cost function of the planner. While innovative, these methods often overlook the robot's dynamics, potentially resulting in infeasible paths. In this work, we demonstrate motion planning by developing a feasible STL-compliant planner and employing reinforcement learning (RL) policies that are effective for unknown dynamical systems to track the computed path. This work aims to provide an end-to-end solution that considers a robot's dynamical constraints (which are assumed to be unknown) through reinforcement learning (RL) and provides a feasible STL-compliant motion planning solution.

Signal Temporal Logic (STL): STL captures high-level specifications consisting of spatial, temporal, and logical constraints [1]. Consider a predicate function $h : \mathbb{R}^n \to \mathbb{R}$, $n \in \mathbb{N}$.

All authors are affiliated with the Robert-Bosch Center for Cyber-Physical Systems at the Indian Institute of Science, Bengaluru, India. This work was supported in part by the Google Research Grant, the SERB Start-up Research Grant, the ARTPARK, the Axis Bank Fellowship, and the CSR Grant by Siemens.

[‡] Authors contributed equally.

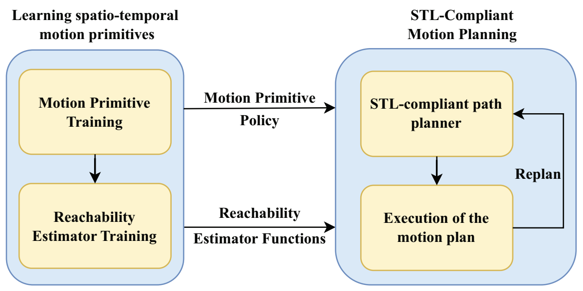
Learning spatio-temporal motion primitives
Motion Primitive Training
Reachability Estimator Training
Motion Primitive Policy
Reachability Estimator Functions
STL-Compliant Motion Planning
STL-compliant path planner
Execution of the motion plan
Replan

Fig. 1: Schematic of the proposed STL-compliant motion planning framework

The set of STL formulae we examine in this context can be recursively expressed using predicates μ. The predicate μ holds true if $h(s) \geq 0$, $s \in \mathbb{R}^n$; otherwise, it holds false.

$$\psi ::= \mathsf{true}|\mu|\neg\psi|\psi_1 \vee \psi_2|\psi_1 \wedge \psi_2 \quad (1)$$
$$\varphi ::= \Diamond_{[t_1,t_2]}\psi|\Box_{[t_1,t_2]}\psi|\neg\varphi|\varphi_1 \wedge \varphi_2|\varphi_1 \vee \varphi_2, \quad (2)$$

where ψ_1 and ψ_2 are STL formulae of form ψ, and φ_1 and φ_2 are STL formulae of the form ψ. Boolean operators for negation, disjunction and conjunction are denoted by $\neg, \vee$ and $\wedge$, respectively. Symbols $\Diamond$, and $\Box$ represent the temporal operators - eventually and always, respectively. $[t_1, t_2] \subset \mathbb{R}_{\geq 0}$ is the time interval with $t_1 \leq t_2$. Nested temporal operators are excluded due to potential complications during the path-planning phase, but will be considered in future work.

Robustness Metric for STL [1]: For a given signal $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and STL formulae ϕ of form (1)-(2), a robustness metric $\rho^\phi(x,t) \in \mathbb{R}$ gives a quantitative indication of how well a task is satisfied. Specifically, the sign of the robustness metric gives the indication of satisfaction ($\rho^\phi(x,0) > 0$) and the magnitude gives robustness of STL formulae. The robustness semantics for the mentioned STL formulae are given as: $\rho^\mu(x,t) = h(x(t))$, $\rho^{\neg\phi}(x,t) = -\rho^\phi(x,t)$, $\rho^{\phi_1\wedge\phi_2}(x,t) = \min(\rho^{\phi_1}(x,t), \rho^{\phi_2}(x,t))$, $\rho^{\phi_1\vee\phi_2}(x,t) = \max(\rho^{\phi_1}(x,t), \rho^{\phi_2}(x,t))$, $\rho^{\Box_{[t_1,t_2]}\phi}(x,t) = \min_{t' \in [t_1,t_2]} \rho^\phi(x,t)$, $\rho^{\Diamond_{[t_1,t_2]}\phi}(x,t) = \max_{t' \in [t_1,t_2]} \rho^\phi(x,t)$.

Problem Definition: Given a robotic system with unknown dynamics and an STL specification of the form mentioned in (1)-(2), design a control strategy to navigate in diverse environments that enforces the STL specification.

II. Proposed Framework

A. Learning Spatio-temporal Motion Primitives

1) Generating Motion Primitive Library: We introduce a methodology centred on learning motion primitives using Reinforcement learning algorithms to generate multiple policies to perform tasks representing each motion primitive. The library of motion primitives allows the sequential construction of a complex trajectory that can be utilized without the need for re-training in diverse environments.

2) Training Reachability Estimator: To apply appropriate policies to enforce STL requirements, it is crucial to determine the duration for which a policy should be applied to generate a desired motion and a desired spatial position. The relationship between distance travelled and time is non-linear due to the presence of non-linearities in dynamics and the environment. Therefore, we propose training reachability estimators utilizing the Gaussian process regression method, which is very efficient in modelling complex, non-linear relationships [3]. We apply each policy to the robot for a monotonically increasing sequence of time intervals, and then record the robot's relative change in position at the end of these intervals. These data samples are then used as training data for the GP model, with displacement as input and the time horizon as output.

B. STL-compliant Path Planner

Given an STL specification (1)-(2), spatiotemporal motion primitive library(obtained using motion primitive and the reachability estimator), we present a modified sampling-based path planner. The nodes are sampled in space and time, represented as (x, y, t), where x, y are the 2-D coordinates of the node, and t is the time required to reach the node. The constraints over the sampling intervals of x, y and t are given by the reachability estimator, thus ensuring that reaching the sampled point is feasible in accordance with the learnt motion primitives. Steering of the tree is done such that robustness measure $\rho^{\phi}(p)$ is maximized, where p is a path representing a sequence of nodes (x, y, t) extended by the planner. This encourages the exploration of tree branches that exhibit positive robustness until the tree is fully developed, while less promising paths with negative robustness (which violate STL specifications) are pruned. The proposed solution is probabilistically complete and asymptotically optimal as the number of nodes sampled tends to infinity.

C. Execution of the Motion Plan

To travel between two nodes of the planned trajectory, we apply appropriate policies from the primitives library in an open loop for a time horizon predicted by the reachability estimator. However, if the robot deviates higher than a certain tolerance from the preplanned node, the trajectory is re-planned to account for unforeseen uncertainties such as changes in the environment, disturbances, etc.

III. Case Study

We apply the proposed STL-compliant framework to a differential drive robot 'Turtlebot3' in a workspace $\mathcal{X} = [0, 3] \times [0, 3]$, for the following specifications: The robot position indicated by (x, y) starts at any start location inside $\mathcal{X}$ at time $t = 0$, and it should navigate to a goal G at $(2.5, 2.5)$ between 22 to 25 seconds while staying in $\mathcal{X}$, and avoiding the static obstacles SO1 and SO2, centred at $(o_{1_x}, o_{1_y}) = (0.75, 2.25)$ and $(o_{2_x}, o_{2_y}) = (2.25, 0.75)$, the timed obstacle TO that appears between 0 to 10-second interval at $(1.5, 1.5)$. $\phi = \Box_{[0,30]}(0 \le x \le 3, 0 \le y \le 3) \wedge \Diamond_{[22,25]}(\|(x, y) - (2.5, 2.5)\|_2 \le 0.25) \wedge \Box_{[5,10]}(\|(x, y) - (1.5, 1.5)\|_2 > 0.5) \wedge \Box_{[0,30]}(\|(x, y) - (o_{i_x}, o_{i_y})\|_2 > 0.4), \forall i = \{1, 2\}$. For Turtlebot3, the generated motion primitive library consists of linear motions (forward and backward) and rotational motions (clockwise and anticlockwise) and RL-based policies for each primitive are constructed, dependent on speed ranges that the robot can follow. We trained the reachability estimators for each policy. Given the STL specification, we formulated the cost functions based on the robustness metric and developed an STL-compliant path plan. This path is then tracked by sequentially applying appropriate policies. It can be observed from Fig 2 that the generated path passes through the obstacle region after 5 seconds. Any other path that avoids the obstacle during the entire duration of planning would violate the STL specification due to the robot's dynamic constraints. We have also experimentally verified the framework on hardware setup of Turtlebot3 and Unitree Go1 quadruped robot with different specifications and initial configuration of the robot. The videos of these demonstrations are available here.

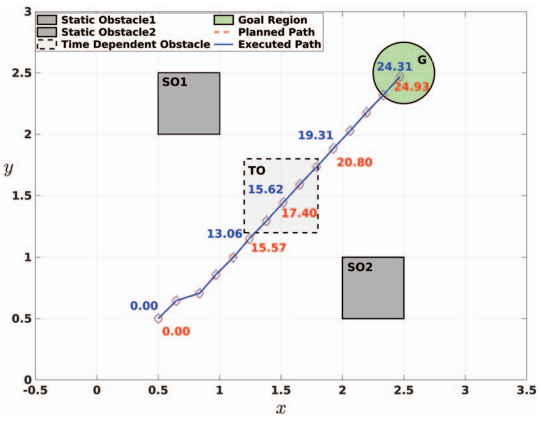

Fig. 2: Results of STL-compliant framework - Red and blue data points indicate the planned reach time (in seconds) and the actual time taken to reach the node during execution, respectively.

References

[1] A. Linard, I. Torre, E. Bartoli, A. Sleat, I. Leite, and J. Tumova, "Real-time RRT* with signal temporal logic preferences," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8621–8627, 2023.

[2] J. Karlsson, F. S. Barbosa, and J. Tumova, "Sampling-based motion planning with temporal logic missions and spatial preferences," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 537–15 543, 2020.

[3] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.