Study of Signal Temporal Logic Robustness Metrics for Robotic Tasks Optimization

Akshay Dhonthi*†, Philipp Schillinger*, Leonel Rozo* and Daniele Nardi‡
* Bosch Center for Artificial Intelligence, Renningen, Germany
†,‡ Department of Artificial Intelligence and Robotics, Sapienza University, Rome, Italy
Email: †dhonthirameshbabu.1887502@studenti.uniroma1.it, *{philipp.schillinger, leonel.rozo}@de.bosch.com,
‡nardi@diag.uniroma1.it

***Abstract*—Signal Temporal Logic (STL) is an efficient technique for describing temporal constraints. It can play a significant role in robotic manipulation, for example, to optimize the robot performance according to task-dependent metrics. In this paper, we evaluate several STL robustness metrics of interest in robotic manipulation tasks and discuss a case study showing the advantages of using STL to define complex constraints. Such constraints can be understood as cost functions in task optimization. We show how STL-based cost functions can be optimized using a variety of off-the-shelf optimizers. We report initial results of this research direction on a simulated planar environment.**

I. Introduction

Many robotic manipulation tasks are sensitive to small changes in execution parameters like target positions or stiffness. Also, applications usually benefit significantly from improved execution time or reliability. However, it usually is a challenge to optimize such tasks due to their sequential nature. To handle such a challenge, *Signal Temporal Logic* (STL) which is a formalism to describe the temporal characteristics of trajectories is well-suited. It provides a real-valued function, called *robustness metric* generated from a logical specification which can be used to evaluate the performance of the robotic tasks.

In this work, we investigate the suitability of different robustness metrics for the purpose of optimizing robotic manipulation tasks. There have been efforts towards applying STL in various robotic environments. [1], [2] develop control techniques for multi-agent systems using STL specification. [3] has used STL rewards in Learning from Demonstration (LfD) on a 2D driving scenario. However, most works focus on directly optimizing the controller, which can be limiting for complex systems. Instead, we propose to use classical black-box optimization to improve existing tasks.

II. STL Robustness Metric

Formally, an STL specification can be understood as follows. Consider a discrete time sequence $t := \{t_k | k \in \mathbb{Z}_{\geqslant 0}\}$. The STL formula φ is defined using the predicate μ that is of the form $f(x(t))$, where $x(t)$ is the state of the signal at time t and f maps each time point to the real-value $x(t_k) \in \mathbb{R}$. The STL syntax is defined as: $\varphi := \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{G}_I\varphi \mid \mathbf{F}_I\varphi \mid \varphi_1 \mathbf{U}_I \varphi_2$. Where $I = [a, b]$ is the set of all $t_k \in t$ such that $a, b \in t$; $b > a \geq 0$. The operators $\neg, \wedge, \vee$ refer to Boolean *negation*, *conjunction* and *disjunction* operators respectively. The temporal operators $\mathbf{G}, \mathbf{F}, \mathbf{U}$ refer to *globally*, *eventually* and *until* operators, respectively. A complete definition of STL can be found in [11], [10].

The *robustness metric* denoted as $r(\varphi, x, t) \in \mathbb{R}$ is the quantitative semantics of the STL formula φ that measures "how well" the signal x is fulfilled at time t. The classical way of defining this semantics is *space robustness* [8]. This measure is positive if and only if the signal satisfies the specification (*Soundness* property) and the closer the robustness is to zero, the smaller are the required changes of signal values to change the truth value. Formally, space robustness and its corresponding operators are defined as follows.

$$\begin{aligned} r(\mu, x, t) &= f(x(t)), \\ r(\neg\varphi, x, t) &= -r(\mu, x, t), \\ r(\varphi_1 \wedge \varphi_2, x, t) &= \min(r(\varphi_1, x, t), r(\varphi_2, x, t)), \\ r(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2, x, t) &= \max_{t_{k1} \in [t+a, t+b]} (\min(r(\varphi_1, x, t_{k1}), \\ &\qquad \min_{t_{k2} \in [t+a, t+t_{k1}]} r(\varphi_2, x, t_{k2}))). \end{aligned} \tag{1}$$

The remaining operators can be derived using (1). Although this is an intuitive and practical way to determine robustness, it has limitations, particularly due to the *min* and *max* functions. They are non-smooth and non-differentiable, making it more difficult for any optimizer to find a good solution. To better address this issue, we discuss other alternatives to space robustness and some of the properties.

A. Robustness types

To address the above issues of space robustness, several alternative definitions have been proposed in recent years.

1) Time Robustness: Instead of looking at how well individual signal values satisfy the specification, time robustness shifts the signal in time to quantify satisfaction [6]. However, time robustness is discontinuous at rising and falling edges due to the switching from positive to negative values. This kind of robustness is useful in cases where it is important to find a signal based on how fast/slow they should satisfy the specification.

2) LSE Robustness: To deal with the smoothness problem of space robustness, [5] proposed a robustness formulation based on the *Log-Sum-Exponential* (LSE) approximation of the *max* and *min* operators. LSE corresponds to the log of a

summation of exponential terms with some scalar multiplier $k \in \mathbb{R}_{\geq 0}$ as scaling factor. This approximation is smooth due to infinitely differentiable approximation and can reduce the influence of spikes in the signal, but it comes with a cost of loosing soundness property.

3) AGM Robustness: The *Arithmetic Geometric Mean* (AGM) is an average-based robustness that modifies all STL operators at all time instances [10]. This definition captures how fast the signal $x \in [-1, 1]$ satisfies the specification by computing arithmetic and geometric means. This normalizing approach can be useful when the signals are of different units. However, AGM does not guarantee convergence with gradient-based optimization techniques [9].

4) Smooth Robustness: It can be seen as a combination of *LSE* and *AGM* robustness metrics [9] because, unlike them, it is both sound and guarantees convergence. The use case is similar to *LSE* with additional feature of giving positive outcome only if the signal satisfies the specification.

5) Avg Robustness: It is a combination of *space* and *time* robustness. It captures how soon or late a signal meets the specification [11] and therefore is suitable to optimize both accuracy and speed in achieving a task. The Avg Robustness does not support nested temporal operators, which can be a drawback.

6) NEW Robustness: It is introduced recently in [7] which is based on a scale-invariant behaviour. The metric is computed by taking weighted average of the effective measures where the weight is defined such that it becomes traditional *space robustness* when it approaches infinity. The test results from the authors show better performance comparing to *AGM robustness* in terms of finding feasible solutions.

B. Metric Properties

[7] summarizes a number of properties that help to classify robustness metrics. Since most of these properties are especially relevant for optimization, we summarize them here for all considered robustness definitions in Table I for the following property definitions. A metric is *sound* if $r(\varphi, x, t) \geq 0$ if and only if the signal x satisfies the specification at time t. A metric is *weakly smooth* if it is continuous everywhere and its gradient is continuous for all points. A metric is said to *converge* if it guaranteed to reach a local maximum with gradient-based optimization. A metric is *shadow-lifting* if the metric increases when making partial progress towards the task specification. A metric is *scale-invariant* if $\max(\alpha a_1 \wedge \alpha a_2) = \alpha \ \max(a_1 \wedge a_2)$ for any $\alpha \in \mathbb{R}$.

III. Learning from STL

In this section, we discuss on the approach of using STL for optimizing tasks and present STL specifications to obtain a desired behavior. Specifically, we focus on optimizing task execution duration and final position of the robot end-effector using CMA-ES and Bayesian optimization (BO). The optimizer gets rewards based on the STL specifications at the end of each task execution and generates different parameters for the next run to finally produce a learned trajectory that fulfills the STL constraints.

TABLE I
Comparison of properties with Robustness Metrics

Properties	Robustness Metrics					
	Space	***LSE***	***Smooth***	***AGM***	***Avg***	***NEW***
Weakly Smooth	No	Yes	Yes	Yes	No	Yes
Sound	Yes	No	Yes	Yes	Yes	Yes
Converge	No	Yes	Yes	No	No	Yes
Shadow-lifting	No	No	No	Yes	Yes	Yes
Scale-invariance	Yes	Yes	Yes	No	Yes	Yes

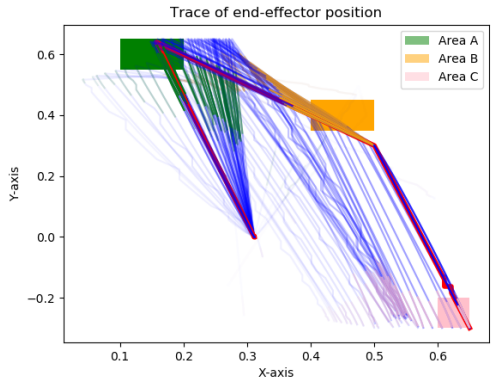

Fig. 1. End-effector trajectories (blue solid lines) at each BO iteration using the *NEW* robustness metric, where color intensity depicts the optimization evolution. The red line represents the highest reward execution. The lines has colored patches which are the time domain when the end-effector has to reach the respective regions.

Let us consider the following desired behavior: "The end-effector has to *eventually* visit three regions $j \in \{A, B, C\}$ within the time intervals 3 to 4, 8 to 10, and 13 to 15 seconds, respectively". Equation (2) represents this as an STL specification.

$$\begin{aligned} \varphi =& \mathbf{F}_{[3,4]}\varphi_A \ \wedge \ \mathbf{F}_{[8,10]}\varphi_B \ \wedge \ \mathbf{F}_{[13,15]}\varphi_C, \\ \varphi_j =& x_{j,lb} < x_j < x_{j,ub} \ \wedge \ y_{j,lb} < y_j < y_{j,ub}. \end{aligned} \tag{2}$$

These specifications show the potential of STL for defining temporal constraints for task optimization. In general, we obtain a lower reward when the behavior fails the constraints soon. The rewards obtained using such specifications can push the optimizer to satisfy all the conditions even when those constraints are in different time instants. This specification is simple to change to achieve different desired outcomes.

IV. Experiments and Results

We conduct experiments on the 7-DOF Panda Robot in a simulated environment. The number of evaluations per experiment is 60, each one taking 25 minutes approximately. Time taken for reward computation and optimization is fast. The robot motion profile has three point to point trajectories. The objective here is to optimize duration parameters and (x, y) end-effector position parameters at each trajectory which totals to 9 parameters. The robot resets to its initial position at the end of each evaluation.

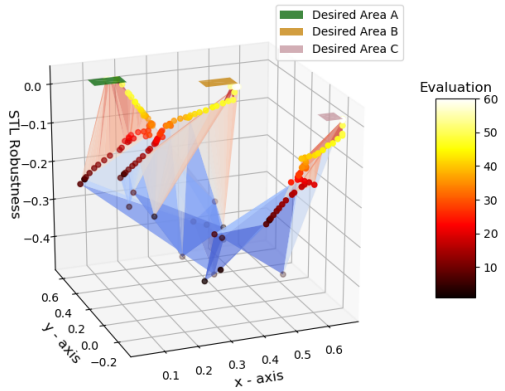

Fig. 2. STL Robustness values as a function of the (x, y) values provided by BO during task optimization considering the three desired regions A, B, C (depicted by colored planes). The used metric is *NEW* robustness.

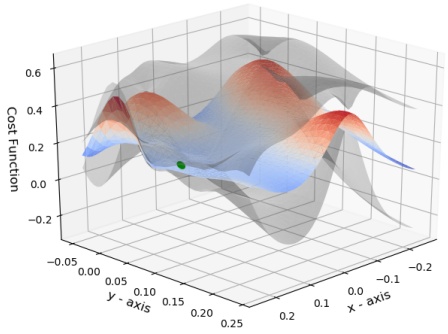

Fig. 3. The cost of Gaussian process as a function of (x, y) positions of the region B at the end of the experiment with *Smooth* robustness. Cost here denotes negative *STL Robustness*. The mean of the GP is shown as colored surface. The $\pm$ variance is given as grey surface. The green dot is the location of the minimum underlying cost.

The trace of the end-effector in Fig. 1 shows trajectory positions converging to the goal regions. Figure 2 displays the obtained rewards, which increase when the trajectories get closer to the desired regions. Figure 3 shows the BO surrogate model, i.e. a *Gaussian Process* (GP), after evaluating 60 iterations. It is clear from Fig. 4 that the optimization is heavily affected by the robustness metric used. Table II shows the performance of all the metrics using *BO* and *CMA-ES* optimizers. The *Success Rate* (SR) is computed as the number of times the robot satisfies all the constraints over 60 epochs. The *Task Satisfaction* (TS) is the first evaluation step when all the STL constraints were satisfied. Some experiments are run until convergence. *Smooth Robustness* performs better with *BO* while *NEW robustness* performs better using *CMA-ES*.

Based on observations from the experiments, optimization with the *space robustness* metric does not always satisfy the constraints when there are more temporal operators. *LSE* and *Smooth robustness* performance is influenced by their respective scaling factors and has to be tuned for different problems. The convergence with *AGM* metric is not guaranteed unless signals are normalised, but this is not straightforward in manipulators due to the complexity of obtaining workspace

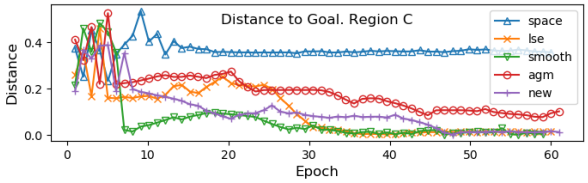

Fig. 4. The figure shows at each epoch, how close the end-effector is to the goal during respective time frames for regions C. The optimizer used is BO.

boundaries. The *NEW* robustness metric is suitable for defining maniputator tasks as they are robust in finding a solution all the time while *Smooth robustness* can converge faster given the scaling factors are tuned.

TABLE II
OVERVIEW TABLE FOR ROBUSTNESS METRIC PROPERTIES

	Type	Robustness Metrics					
		Space	***LSE***	***Smooth***	***Avg***	***AGM***	***NEW***
BO	SR	12.71%	10.0%	**29.9%**	9.92%	20.0%	27.41%
	TS	75	68	**33**	85	39	51
CMA	SR	1.67%	5.0%	3.51%	0.0%	3.7%	**6.67%**
	TS	58	48	32	Fail	50	**29**

V. CONCLUSION

In this paper, we exploited STL-based constraints as cost functions to optimize simple robotic manipulation tasks. We analyzed several STL-based cost functions and showed their influence on optimizing simple robot trajectories in multiple-target reaching tasks. With several experiments, it is possible to see *Smooth* and *NEW robustness* are suitable with classical black-box optimizers such as *BO* and *CMA-ES*. Further, this work can be extended by considering orientation parameters and nested temporal STL specifications on other optimizers.

REFERENCES

[1] Lindemann, L et al., (2020). Barrier function based collaborative control of multiple robots under signal temporal logic tasks. IEEE TCNS, 7(4), 1916–1928.

[2] Gundana, D., and Kress-Gazit, H. (2021). Event-based signal temporal logic synthesis for single and multi-robot tasks. IEEE RAL, 6(2), 3687–3694.

[3] Puranic et al., (2021). Learning from Demonstrations Using Signal Temporal Logic in Stochastic and Continuous Domains. IEEE RAL.

[4] Brochu et al., (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. ArXiv Preprint:1012.2599.

[5] Pant, Y et al., Smooth operator: Control using the smooth robustness of temporal logic. *2017 IEEE CCTA*. pp. 1235-1240 (2017)

[6] Donzé, A. & Maler, O. Robust satisfaction of temporal logic over real-valued signals. *ICFMATS*. pp. 92-106 (2010)

[7] Varnai, P. & Dimarogonas, D. On robustness metrics for learning STL tasks. *2020 ACC*. pp. 5394-5399 (2020)

[8] Belta, C. & Sadraddini, S. Formal methods for control synthesis: An optimization perspective. *Annual Review Of CRAS*. **2** pp. 115-140 (2019)

[9] Gilpin et al., A smooth robustness measure of signal temporal logic for symbolic control. *IEEE CSL*. **5**, 241-246 (2020)

[10] Mehdipour et al., Arithmetic-geometric mean robustness for control from signal temporal logic specifications. *2019 ACC*. pp. 1690-1695 (2019)

[11] Aksaray et al., Q-learning for robust satisfaction of signal temporal logic specifications. *2016 IEEE CDC*. pp. 6565-6570 (2016)