2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Optimizing Demonstrated Robot Manipulation Skills for Temporal Logic Constraints

Akshay Dhonthi*†§, Philipp Schillinger*, Leonel Rozo* and Daniele Nardi‡
* Bosch Center for AI, Renningen, Germany
† Formal Methods and Tools, University of Twente, Enschede, Netherlands
‡ Department of AI and Robotics, Sapienza University, Rome, Italy
Email: †a.dhonthirameshbabu@utwente.nl, *{philipp.schillinger, leonel.rozo}@de.bosch.com,
‡nardi@diag.uniroma1.it

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981384

***Abstract*—For performing robotic manipulation tasks, the core problem is determining suitable trajectories that fulfill the task requirements. Various approaches to compute such trajectories exist, being learning and optimization the main driving techniques. Our work builds on the learning-from-demonstration (LfD) paradigm, where an expert demonstrates motions, and the robot learns to imitate them. However, expert demonstrations are not sufficient to capture all sorts of task specifications, such as the timing to grasp an object. In this paper, we propose a new method that considers formal task specifications within LfD skills. Precisely, we leverage Signal Temporal Logic (STL), an expressive form of temporal properties of systems, to formulate task specifications and use black-box optimization (BBO) to adapt an LfD skill accordingly. We demonstrate our approach in simulation and on a real industrial setting using several tasks that showcase how our approach addresses the LfD limitations using STL and BBO.**

I. Introduction

Learning from demonstration (LfD) is a paradigm that uses expert demonstrations to derive robot control policies [1]. In manipulation tasks employing torque-controlled industrial robots, kinesthetic teaching is often exploited, where an expert operator physically moves the robotic arm to generate demonstrations of a task [2]. However, these demonstrations alone may not be sufficient for learning tasks that involve contacts or high accuracy and reliability [3].

In industrial settings, there are several limitations while generating demonstrations: First, it is difficult for the expert to show accurate time constraints in the task execution. For instance, reaching or avoiding a specific region during a given time interval. Second, minor additions to the task require additional full demonstrations to subsequently update the learning model. These additions may be spatial (e.g. choosing an alternative path) or temporal (e.g. completing different parts of the task at different speeds or time intervals). And third, the demonstrations do not always match the desired task performance, and therefore there may be room for improvement in task reliability, accuracy, or execution time.

Some of the aforementioned challenges may be addressed by learning reward functions from expert examples via inverse reinforcement learning [4]. However, the reward function learning heavily depends on expert data, which means that when new task requirements arise, the human often needs to provide additional demonstrations to trigger a new training process of the model or the reward function. Also, defining the specific structure of reward functions is not trivial.

Due to these difficulties, we tackle this problem from an optimization perspective, as illustrated in Fig. 1. In our method, when new task requirements arise, we use a black-box optimization strategy to refine the LfD model parameters. This optimization exploits an objective function that captures the desired spatial and temporal constraints of the new task using Signal Temporal Logic (STL) [5]. STL is a formalism capable of representing high-level task objectives as logical semantics, making their design more user-friendly. STL provides a real-valued function called *robustness degree*, generated from a logical specification, which makes the evaluation of task satisfaction appealing for black-box optimizers.

There have been several works on leveraging STL on robotic applications [6]–[9]. Aksarya *et al.* [6] proposed an optimal policy learning scheme for satisfying STL specifications. Although this method does not fully fall under our LfD-based approach, it is worth highlighting that STL specifications can be quantified and used as reward functions for learning a robot policy. Additionally, unlike [6] where the agent acts in a discrete environment, our approach focuses on continuous signals. Innes *et al.* [10] used neural networks to learn the expert demonstrations. For these networks, *Linear Temporal Logic* (LTL) specifications were transformed into a differentiable loss function. In contrast, our approach uses STL and is therefore more general because STL captures continuous propositions and robustness, as well as time intervals.

The closest approach to our work is [11], where a quantified STL is used to rank the quality of demonstrations as a function of robustness. Then, their approach learns a reward function from which an optimal policy is derived via reinforcement learning. This technique explores trajectories around the demonstrations as possible states leading to high rewards. Our approach differs from this technique in two ways: First, we exploit expert demonstrations directly to learn an LfD model without accounting for STL during this learning stage. Second, we consider scenarios where new task requirements,

§This work was carried out at Bosch Center for AI and Sapienza University.

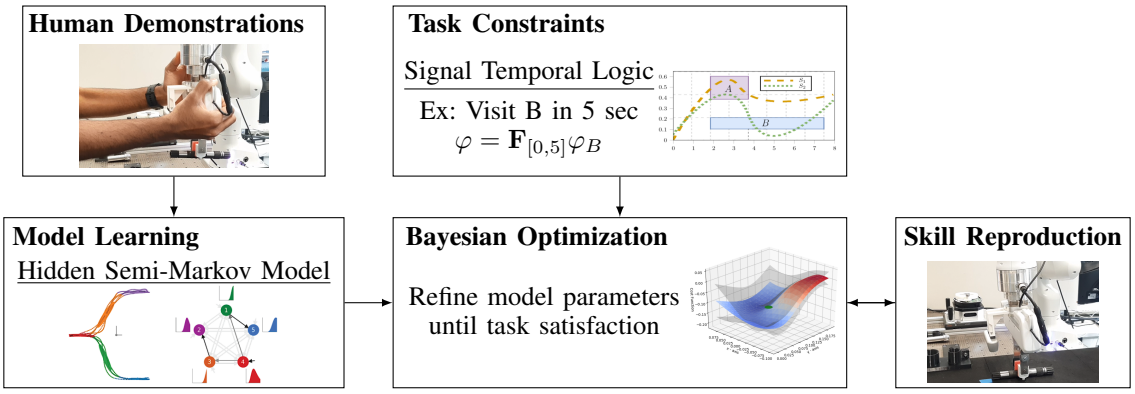

Fig. 1. Illustration of the proposed method. We collect several demonstrations of robotic manipulation skills, which are then used to learn a TP-HSMM model. This model encapsulates the observed spatial and temporal patterns of the demonstrations. We then define new task constraints based on STL specifications. Finally, we leverage Bayesian optimization to update the TP-HSMM model parameters in order to fulfill the new STL-based task requirements.

previously unseen, may be added to existing skills.

In summary, our contributions are threefold: (1) We leverage STL specifications to design explicit task objectives on top of previously-learned robotic skills; (2) We demonstrate that Bayesian optimization for these STL-based specifications overcomes the issue of suboptimal demonstrations and allows the operator to include new task requirements in the skill model; (3) We validate our approach in two experiments that exploit STL rewards to achieve task objectives given by a human operator, implemented in simulation and on a real industrial setup.

II. Background

This section provides a brief introduction to STL and computation of traditional robustness degree using an example. Further, we introduce the optimization technique and the LfD model used in this work.

A. Signal Temporal Logic (STL)

Formally, an STL specification can be understood as follows: Consider a discrete time sequence[1] $t_{0:k} \in \mathbb{R}^k$. The STL formula φ is defined using the predicate μ (an atomic proposition, i.e. a point-wise constraint using $>$ or $<$ operators) that is of the form $f(x(t_{0:k}))$, where $x(t_k)$ is the state of the signal[2] (e.g. a robot trajectory, joint velocity, joint torques, etc.) at time t_k. Moreover, the predicate function $f : \mathbb{R}^k \to \mathbb{R}$ maps each time point to a real-value [5], [6]. Then, the STL syntax is defined as,

$$\begin{aligned} \varphi \ &:= \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \\ &\mid \mathbf{G}_I\varphi \mid \mathbf{F}_I\varphi \mid \varphi_1 \mathbf{U}_I \varphi_2, \end{aligned} \tag{1}$$

where $I = [a, b]$ is the non-empty set of all $t \in t_{0:k}$ such that $a \leq t \leq b$. The operators $\neg, \wedge, \vee$ refer to Boolean *negation*, *conjunction* and *disjunction*, respectively. The temporal operators $\mathbf{G}, \mathbf{F}, \mathbf{U}$ represent *globally*, *eventually* and *until* statements, respectively. The satisfaction of μ is TRUE ($\top$) if the predicate is satisfied and FALSE ($\bot$) otherwise. The globally $\mathbf{G}_{[a,b]}$ operator states that φ must be true at all times in $[a, b]$, and the eventually $\mathbf{F}_{[a,b]}$ operator states that φ must be true at some time point in $[a, b]$. Similarly, the until $\mathbf{U}_{[a,b]}$ operator states that φ_1 must be true until φ_2 eventually becomes true in the time interval $[a, b]$.

[1] We use subscripts to denote sequences of data. i.e. $t_{0:k} = \{t_0, \cdots, t_k\}$

[2] The signal is in discrete form with k steps and it is referred to as $x(t)$ throughout the paper for simplicity.

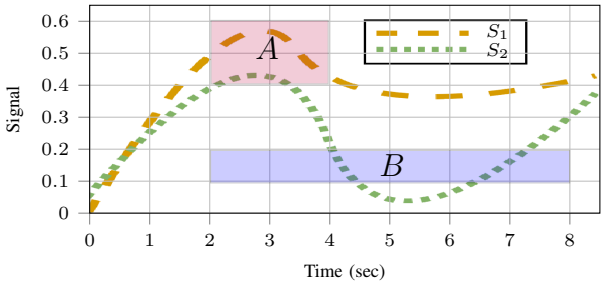

Fig. 2. For the STL specification to visit region A and avoid region B, the signal S_1 satisfies it and the signal S_2 violates it.

As an example, let us consider the two one-dimensional signals shown in Fig. 2. Also, consider the requirement: "The signal should pass through region A during 2 to 4 seconds and avoid region B during 2 to 8 seconds". This can be translated into an STL specification as follows:

$$\varphi = \mathbf{F}_{[2,4]}\varphi_A \ \wedge \ \mathbf{G}_{[2,8]}\neg\varphi_B, \tag{2}$$

where φ_A is given by the bounds $0.4 < S_i(t) < 0.6$. The proposition for this is $f_A(S_i(t)) := 0.1 - |S_i(t) - 0.5|$. Also, φ_B is defined accordingly. Figure 2 shows how the signal S_1 satisfies the specification given by (2) whereas S_2 fails due to the violated requirement of avoiding region B.

B. Robustness Degree

The robustness degree, 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 these semantics is via *space robustness* [12]. This measure is positive if and only if the signal satisfies the specification (i.e. *soundness* property). The closer the robustness is to zero, the smaller the required changes of signal

values are to change the truth value. Formally, space robustness and its corresponding operators are defined as follows

$$\begin{aligned} r(\mu, x, t) &= f(x(t)), \\ r(\neg\varphi, x, t) &= -\, r(\mu, x, t), \\ r(\varphi_1 \wedge \varphi_2, x, t) &= \min(r(\varphi_1, x, t), r(\varphi_2, x, t)), \\ r(\varphi_1 \vee \varphi_2, x, t) &= \max(r(\varphi_1, x, t), r(\varphi_2, x, t)), \\ r(\mathbf{G}_{[a,b]}\varphi, x, t) &= \min_{t_k \in [t+a, t+b]} (r(\varphi, x, t_k)), \\ r(\mathbf{F}_{[a,b]}\varphi, x, t) &= \max_{t_k \in [t+a, t+b]} (r(\varphi, x, t_k)). \end{aligned} \tag{3}$$

Concerning our example depicted in Fig. 2, the robustness value is 0.1 for the signal S_1 and -0.05 for the signal S_2. The signal S_2 gets a negative reward since it fails to satisfy the global condition to avoid region B.

C. Black-box Optimization (BBO)

Black-box optimizers are widely used nowadays in several fields of machine learning [13], [14]. BBO is a sample-efficient technique to find optimal system parameters that maximize a task-specific objective function [15]. In a typical BBO setting, the objective and constraint functions are only accessible through (possibly noisy) output values, and therefore gradient-based approaches are infeasible. For this work, we use *Bayesian Optimization* (BO), which seeks an optimal set of parameters,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\operatorname{argmin}}\, g(\boldsymbol{\theta}). \tag{4}$$

BO is a class of sequential search algorithms for optimizing black-box functions [16]. BO finds a global minimizer $\boldsymbol{\theta}^*$ as in (4), where g is an unknown objective function defined over a domain of interest $\boldsymbol{\Theta}$. The function g is not observed directly, as only noisy output values are available. Using this noisy information about the objective function, BO selects a query point $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ at which to evaluate g at each optimization step. Such a selection process is carried out by optimizing an acquisition function, which resolves the explore-exploit trade-off. Several forms of acquisition function exist in the literature, and we use probability of improvement (PI) in this work [17]. PI is a greedy search algorithm that selects the most likely point to offer an improvement.

D. Learning from Demonstration (LfD)

LfD may be seen as a robot programming strategy that uses human demonstrations to build a learning model that encapsulates the main motion patterns of a skill or task. As our interest is on learning object-centric manipulation skills, we leverage *Task-Parameterized Hidden Semi-Markov Models* (TP-HSMM) [18] which have been successfully used in elaborated industrial tasks [19]. TP-HSMM is an object-centric model that can be easily used to sequence several skills to accomplish complex assembly tasks [19]. This parametric model encapsulates skill-specific spatial and temporal patterns from human demonstrations. Such encoding features are particularly useful in our setting due to their close relationship to the spatial and temporal requirements specified via STL.

Specifically, a TP-HSMM model learns a joint probability density function of the demonstrations using an object-centric formulation of hidden semi-Markov models (HSMM) [20]. The underlying process of an HSMM is a Markov chain with one extension: Each model state has variable duration, affecting how the system transits between states. The TP-HSMM parameters are estimated by following a modified version of the Expectation-Maximization algorithm [18]. Then, a linear quadratic tracker is used to synthesize a smooth trajectory following the main spatial and temporal patterns, encapsulated in the TP-HSMM model, as proposed in [18], [19]. Note that the task-parameterized formulation allows the robot to adapt its motion to the pose changes in the manipulated objects.

Here we provide a short description of the TP-HSMM parameters relevant to the subsequent STL-based optimization process. A set of K states characterizes a TP-HSMM. They are usually represented by normal distributions $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with $k = 1 \dots K$, which model the observation probabilities of the demonstrations on a set of coordinate systems (a.k.a. task parameters)[3]. The *transition probability matrix* $\mathbf{A} \in \mathbb{R}^{K \times K}$ with $A_{i,j} \triangleq P(z_t = j | z_{t-1} = i)$, defines the probability of transiting from state i to state j and encapsulates sequential information of the skill. Finally, the *duration probability* is represented by a normal distribution $\mathcal{N}(s | \mu_j^S, \Sigma_j^S)$, and indicates how long the robot stays in the model state j, therefore encapsulating temporal information of the skill. The spatial and temporal patterns of skill are then fully described by the duration probability, the transition matrix, and the observation probabilities, as detailed in [20].

III. Methodology

The core idea of our approach is to integrate LfD models, BO, and STL, to handle new spatial or temporal task requirements for fast skill adaptation. One of the first challenges in this context is to define the objective function and parameter space of interest. On the one hand, the objective function (reward function) is designed from the robustness degree $r(\varphi, x, t)$ computed via the STL specification φ, introduced in Sec. II-A. On the other hand, the parameter space can be defined as a (sub)set of the skill model parameters. Importantly, we can exploit the variance information encoded by the TP-HSMM model to design the corresponding bounds for the set of parameters as in [21]. We therefore assume that the optimal solution is safe as long as the variance-based bounds are not violated. This section covers both challenges in detail and illustrates them through several examples.

Algorithm 1 shows the STL-based Bayesian optimization procedure. The goal is to refine the model parameters $\boldsymbol{\delta}$ such that the new task requirements, defined via STL specifications, are satisfied. To begin with, we set the optimization parameters $\boldsymbol{\delta}_i$ at random for the first M iterations. Using these initial parameters set, we evaluate the objective function during the corresponding runs on the real system. Following M

[3]For the sake of simplicity, we assume that the demonstrations correspond to a single task-parameter setting.

Algorithm 1 STL-based Bayesian optimization of LfD skills

Input: Optimization iterations N, number of evaluations M to build initial observations set, STL specification φ, and TP-HSMM parameters $\boldsymbol{\delta}_0 \in \mathbb{R}^P$, where P is the total number of considered parameters.

Output: Updated model parameters $\boldsymbol{\delta}^* \in \mathbb{R}^P$

1: **for** $i = 1, 2, \cdots N$ **do**
2: **if** $(i \leq M)$ **then**
3: Randomly set $\boldsymbol{\delta}_i$ under respective upper and lower bounds $\boldsymbol{\mathcal{B}}_{1:P} \in \mathbb{R}^{2P}$
4: **else**
5: Find $\boldsymbol{\delta}_i$ under bounds $\boldsymbol{\mathcal{B}}_{1:P}$ by optimizing the PI acquisition function using the GP information: $\boldsymbol{\delta}_i = \text{argmax}_{\delta}\, u(\boldsymbol{\delta}|\mathcal{D}_{1:i-1})$
6: **end if**
7: Skill retrieval using new model parameters $\boldsymbol{\delta}_i$ and linear quadratic tracking as in [19].
8: Execute the skill and record signals of interest.
9: Calculate the robustness $r(\varphi, x, t)$ to quantify the STL specification φ for signal x.
10: Augment the data $\mathcal{D}_{1:i} = \{\mathcal{D}_{1:i-1}, (\boldsymbol{\delta}_i, r(\varphi, x, t))\}$ and update the GP model.
11: **end for**
12: **return** The model parameters with maximum reward $r(\varphi, x, t)$ from the data $\mathcal{D}_{1:N}$

evaluations, as indicated in line 5, we find the next set of parameters by maximizing the acquisition function u (in this case, *PI*), which directs the search toward the optimum.

At each evaluation step i in line 7, we obtain a new skill trajectory for the respective parameter set $\boldsymbol{\delta}_i$ using the retrieval technique discussed in Sec. II-D. We run this trajectory and record the relevant multi-dimensional signal $\boldsymbol{x}(t)$ as a discretized time sequence corresponding to the specifications' propositions. Further, line 9 uses the STL specification φ and the logged signal to compute the robustness degree $r(\varphi, x, t)$ that acts as the objective function. The recorded observations and the *Gaussian Process* (GP) are updated accordingly before the next iteration. We run the optimization process for N iterations to produce an optimal trajectory that maximizes the objective function. The following subsection describes a thorough approach for calculating the robustness degree.

A. Objective Function

We tackle the robustness degree computation here, particularly the technique *New Robustness* [22]. We outline the definition of operators of *New Robustness* and refer to the original work in [22] for further details. This computation relies on the multi-dimensional signal $\boldsymbol{x}(t)$ logged at each iteration of Algorithm 1, see line 8. The signal captures relevant measures for the stated specification, such as the robot position in task space, joint torques, joint velocities, or contact forces. The STL specification can then contain several predicates μ that evaluate this signal. To recap Sec. II-A, each predicate μ is of the form $f(x_p(t))$ where f maps each point in the signal to a real value.

The robustness degree, *New Robustness* [22], was experimentally compared against alternative robustness formulations in our previous work [23]. We choose *New Robustness* here because it performed best in finding optimal solutions with a faster convergence rate for the manipulation tasks studied in [23]. The computation of *New Robustness* is based on the elementary definitions of the negation $\neg$ and conjunction $\wedge$ operators. The paper [22] introduced a structured definition of robustness degree with which it is possible to derive all the boolean and temporal operators of STL using just the $\neg$ and $\wedge$ operators. Additionally, the $\neg$ operator can also be excluded from the definition by setting it to a negative value of $\wedge$. Therefore, the computation of any STL operator boils down to the definition of the conjunction operator, which is

$$(\varphi_1 \wedge \cdots \wedge \varphi_m) := \begin{cases} \frac{\sum_i \rho_{\min} e^{\tilde{\rho}_i} e^{\nu \tilde{\rho}_i}}{\sum_i e^{\nu \tilde{\rho}_i}} & if\ \rho_{\min} < 0, \\ \frac{\sum_i \rho_i e^{-\nu \tilde{\rho}_i}}{\sum_i e^{-\nu \tilde{\rho}_i}} & if\ \rho_{\min} > 0, \\ 0 & if\ \rho_{\min} = 0, \end{cases} \quad (5)$$

with,

$$\rho_i = r(\varphi_i, x, t),\ \rho_{\min} = \min(\rho_1 \cdots \rho_m),\ \tilde{\rho}_i = \frac{\rho_i - \rho_{\min}}{\rho_{\min}}, \quad (6)$$

where $\rho_{\min}$ is the quantified conjunction operator using traditional space robustness. The parameter $\nu > 0$ tends to traditional conjunction as $\nu \to \infty$.

This robustness formalism is suitable in the robot task definition for the following reasons: It satisfies the soundness property; as for any specification, the robustness degree returns a positive value if and only if the signal satisfies φ at time t. It is achieved by preserving the sign in the definition of $\rho_{\min}$ (see [24] for the proof). Thus, the output of (5) quantifies how much a task is satisfied or violated, which is beneficial in manipulation tasks where complete satisfaction is crucial. This robustness degree also captures partial progress towards the goals and is beneficial in faster guiding towards task optimum. These properties are beneficial for *reward shaping* [25] to obtain indicative samples during the optimization procedure and for fitting the GP, although the relation between a signal x and the skill model parameters $\boldsymbol{\delta}$ remains a black-box function.

We turn the attention to two examples with different STL specifications for illustrating the approach: Consider a simple skill in which the robot end-effector visits three regions $L_{1:3}$ in Cartesian space. We first learn a TP-HSMM model with K components from human demonstrations. The new task requirements that must be satisfied using our approach are

1) While satisfying the main task of visiting three regions $L_{1:3}$, the end-effector must pass through another region L_4 in the time interval $(8, 12)$ seconds and stay inside the region L_2 during $(12, 15)$ seconds. The STL specification for these new task requirements is given as

$$\varphi_1 = \mathbf{F}\varphi_{L_1} \wedge\ \mathbf{G}_{[12,17]}\varphi_{L_2} \wedge\ \mathbf{F}\varphi_{L_3} \wedge\ \mathbf{F}_{[8,12]}\varphi_{L_4}. \quad (7)$$

2) The robot end-effector must visit regions $\{L_1, L_3\}$ at any time, avoid region L_2 during $(12, 17)$ seconds and visit a new region L_4 during the time interval $(8, 12)$ seconds. The STL specification for these new task conditions is

$$\varphi_2 = \mathbf{F}\varphi_{L_1} \wedge \mathbf{G}_{[12,17]}\neg\varphi_{L_2} \wedge \mathbf{F}\varphi_{L_3} \wedge \mathbf{F}_{[8,12]}\varphi_{L_4}. \quad (8)$$

The regions L_i with $i \in \{1, 2, 3, 4\}$ are specified using Cartesian bounds as follows

$$\begin{aligned}\varphi_{L_i} = & x_{i,lb} < x < x_{i,ub} \ \wedge \ y_{i,lb} < y < y_{i,ub} \ \wedge \\ & z_{i,lb} < z < z_{i,ub}.\end{aligned} \quad (9)$$

Note that the required signals in (9) correspond to the Cartesian coordinates x, y, z of the robot end-effector.

Let us assume that we ran a trial of the experiment and recorded the aforementioned signals. We discretize them and, for simplicity, define the signals of x, y, z coordinates as $x(t), y(t), z(t)$, respectively. Each region has three propositions, one for each Cartesian bound. For instance, the proposition of the x-coordinate (similarly for y and z) for the regions L_i is written as follows,

$$f_{L_i,x}(x(t)) := \frac{x_{i,ub} - x_{i,lb}}{2} - \left| x(t) - \frac{x_{i,ub} + x_{i,lb}}{2} \right|. \quad (10)$$

At the end of these steps, we have several propositions separated by operators. We now use (5) to compute a real-value that describes how well a skill satisfied the task. Using this, the GP is updated, and we get new model parameters for the next iteration. The loop is continued until termination.

B. Parameter Space

The parameter space is defined as $\boldsymbol{\delta} = \{\boldsymbol{\mu}_{1:K}, \boldsymbol{\mu}^S_{1:K}, \boldsymbol{A}\}$, where $\boldsymbol{\mu}_{1:K}$ are the Gaussian components positions, $\boldsymbol{\mu}^S_{1:K}$ are the duration probabilities, and $\boldsymbol{A}$ is the transition matrix. Next, we discuss the role of each of these parameters in detail.

1) Component positions: As described in Sec. II-D, each TP-HSMM state is represented by a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Once a skill model is learned, these Gaussian components are used to retrieve a trajectory to be reproduced by the robot. Therefore, we can modify the TP-HSMM states to reshape the robot trajectory. For instance, modifying the mean $\boldsymbol{\mu} \in \mathbb{R}^3$ will translate the Gaussian in Cartesian space. Therefore, we adapt $\boldsymbol{\mu}$ at each iteration under certain bounds $\boldsymbol{\mathcal{B}}_\mu$ for each component. Bounds define how far the components may move in any direction. The maximum number of parameters considered for optimization is $K \times 3$ as each component $\boldsymbol{\mu}$ has three parameters to represent its position in 3D space. Shifting the components is essential to address the spatial constraints of a task. For instance, refining $\boldsymbol{\mu}$ is beneficial in the STL specification φ_1 where a new region L_4 has to be visited at a particular time interval.

2) Duration probabilities: Each TP-HSMM state is also assigned a duration probability represented by a normal distribution $\mathcal{N}(s|\mu^S, \Sigma^S)$, which defines how long to stay in that model state. Raising or lowering the value of the mean $\mu^S \in \mathbb{R}$ increases or decreases the time spent in the respective component, thus affecting the task timing. We adapt μ^S under the bounds $\mathcal{B}_{\mu^S} = [-\sigma^S, \sigma^S]$, where σ^S is the corresponding standard deviation of the duration probability. The maximum number of parameters is K when considering to optimize duration probabilities of all components. Modifying duration probabilities allows us to optimize the temporal constraints of a task. For instance, adapting μ^S is beneficial in φ_1, where the condition to globally stay in the region L_2 for the time interval $(12, 17)$ is possible by reducing the duration of all the components except the one associated with L_2.

3) Transition matrix: The matrix $\boldsymbol{A}$ defines the probability of transiting to the different model states. Modifying the transition probabilities is carried out by updating the elements of the matrix $\boldsymbol{A} \in \mathbb{R}^{K \times K}$. The bounds of the element $\mathbf{A}_{ij}$ are naturally $\mathcal{B}_{A_{ij}} = [0, 1]$ as $\boldsymbol{A}$ encodes probability measures. We may skip a component j if the adjacent element $A_{i,j+1}$ in the row $\boldsymbol{A}_i$ has a higher value when compared to the element A_{ij}. Therefore, the algorithm modifies the transition matrix during optimization to get an optimal states sequence for the new task requirements. To keep the sum of each row $\boldsymbol{A}_{i,:}$ equal to 1, we normalize the matrix before using it in the retrieval phase. The maximum number of parameters considered for optimization is $K(K-1)/2$. Each model state covers a specific region in space, and therefore changing the transition matrix may skip a region that was part of the demonstrated trajectories, which can be helpful to satisfy the specification φ_2.

IV. Experiments and Results

We here show several experiments to test our approach on a variety of robot skills and using several STL specifications. We evaluate our approach in simulation and on a real-robot task. In the former, we show how our approach can adjust the skill to changing task needs over time and space. In the latter, we test our approach on industrial assembly tasks to generate a robust trajectory that fulfils the task requirements. In both settings, we use the Franka-Emika Panda 7-DOF robotic arm.

A. Simulation experiments

Our goal is to show that our approach handles spatial and temporal task constraints like staying or avoiding a region during specific time intervals. We also show that it is possible to modify a skill by either adding new unseen goals or removing a part of the skill. The number of iterations N for these experiments is set to 32. Each iteration required around a minute to execute the skill on the simulator and less than a second to compute the rewards and obtain new parameters. We recorded a couple of demonstrations as in Fig. 3 on a real robot by moving the end-effector to three regions $L_{1:3}$ in task space in the same order. For simplicity, we kept the initial and final positions of the end-effector unchanged in all the provided demonstrations. Therefore, the problem is no more task parameterized as we do not observe the demonstrations from different perspectives, and hence the TP-HSMM model boils down to a simple HSMM model. We train this model by setting $K = 6$ experimentally.

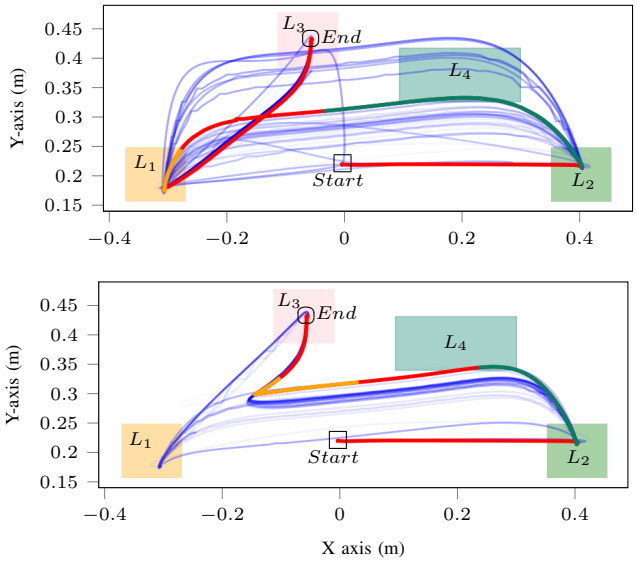

Fig. 3. End-effector trajectories on (x, y)-plane (blue solid lines) at each BO iteration for φ_1 (top) and φ_2 (bottom). The color intensity depicts the optimization evolution and the red line represents the highest reward execution. The lines have colored patches representing the time interval when the end-effector has to reach the respective regions.

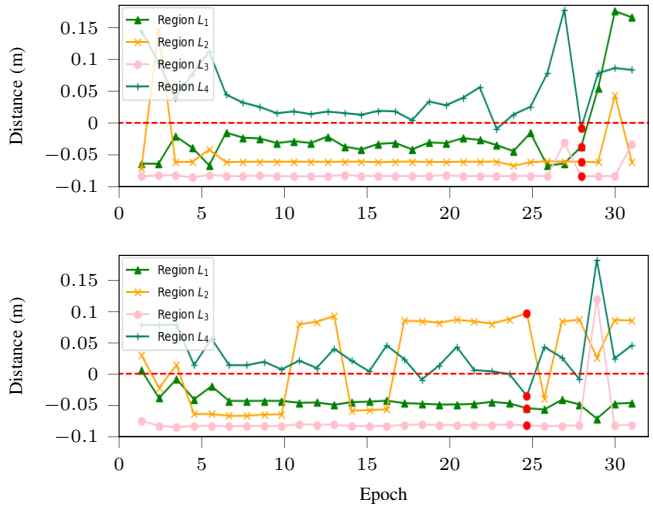

Fig. 4. Signed distance function for each region under respective time constraints for φ_1 (top) and φ_2 (bottom). The horizontal dotted line shows the region's boundary in space. The positive and negative values denote the distances to the regions and boundaries when inside the regions respectively. The red dot depicts the iteration with the maximum reward.

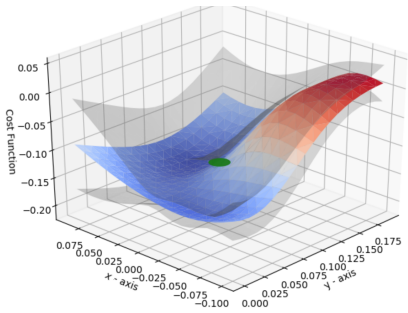

Fig. 5. BO surrogate Gaussian Process model projected on the (x, y) positions of component 3 of the skill. The cost denotes negative STL Robustness. The coloured surface shows the mean of the model. The grey surface is the $\pm$ variance, and the green point is the location of the minimum underlying cost.

For the STL specification φ_1 defined in (7), to accommodate the task of visiting a new region and staying inside L_2 for several seconds, we adjusted the Gaussian component positions $\boldsymbol{\mu}_{1:6}$ and the duration probabilities $\boldsymbol{\mu}^S_{1:6}$, respectively. Since this yields a total of 24 parameters, which is a challenging dimensionality for BO, we further reduced the parameter domain by considering only the most relevant component positions for this task. Specifically, we only optimized $\boldsymbol{\mu}_{1:2}$ as we observed that the addition of L_4 was required during the first few seconds of the trajectory. Further, we used all the duration probabilities $\boldsymbol{\mu}^S_{1:6}$ as they all influence on obtaining the trajectory. Finally, the reduced number of parameters is 12. Thus, the resulting parameter space is $\boldsymbol{\delta}_{\varphi_1} = \{\boldsymbol{\mu}_{1:2}, \boldsymbol{\mu}^S_{1:6}\}$.

The trajectory of the end-effector in Fig. 3-top shows that the additional time constraint on L_2 to stay inside the region for the entire 5 seconds, depicted by the yellow trajectory segment, is satisfied. Moreover, the trace also shows the inclusion of L_4 into the trajectory. We can observe in Fig. 4-top that all the regions were visited under the respective time bounds after several iterations. Figure 5 shows the low-dimensional BO surrogate model representation, i.e. the Gaussian Process of the third component position along (x, y) axes, after evaluating 32 iterations. We can see that by translating the third component around $(0.025, 0.5)$ in Fig. 5, the cost is minimized (i.e. task satisfaction is maximized).

For the second task specification φ_2 , along with $\boldsymbol{\mu}_{1:2}$ and $\boldsymbol{\mu}^S_{1:6}$, we also considered the transition matrix $\mathbf{A}$. This allows us to remove regions visited in initial demonstrations by pruning transitions to irrelevant Gaussian components. For this task, we reduce the number of parameters by restricting the transition matrix to the entries

$$\mathbf{A}_{red} = \begin{pmatrix} 0 & A_{01} & A_{02} & 0 & 0 & 0 \\ 0 & 0 & A_{12} & A_{13} & 0 & 0 \\ 0 & 0 & 0 & A_{23} & A_{24} & 0 \\ 0 & 0 & 0 & 0 & A_{34} & A_{35} \\ 0 & 0 & 0 & 0 & 0 & A_{45} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{11}$$

effectively preventing to skip multiple components in immediate sequence to limit severe deviations from the original model. Therefore, the number of parameters reduces to 21 from the initial 27 and the final parameter space is given by $\boldsymbol{\delta}_{\varphi_2} = \{\boldsymbol{\mu}_{1:2}, \boldsymbol{\mu}^S_{1:6}, \boldsymbol{A}_{red}\}$.

After optimizing the reduced parameter set, Fig. 3-bottom depicts the trace of the end-effector at each iteration. As shown in Fig. 4-bottom, the region L_2 is avoided and the region L_4 is added as specified in the STL constraint. Figure 6 shows the GP for duration probability factors after 32 iterations. It shows that to minimize the cost, the duration probability μ^S_1 has to be moved to the right, therefore increasing the time to stay in

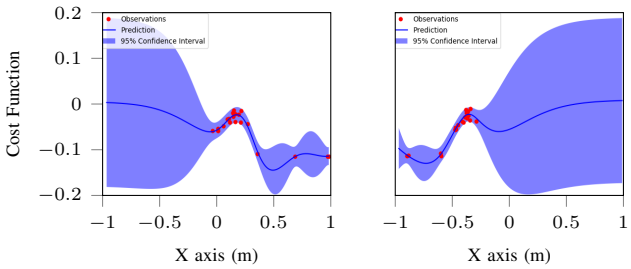

Fig. 6. Gaussian Process of the duration probabilities for component 1 (left) and component 3 (right) at the end of experiment for the test φ_2. The solid blue line represents the predictions, and the red dots depict the observations. The coloured area corresponds to the 95% confidence interval.

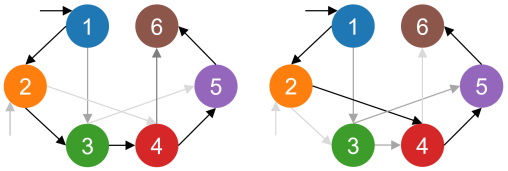

Fig. 7. Transition graph extracted from the transition matrix for the STL specification φ_2 before (left) and after (right) optimization. The black arrows depict the final transition sequence. Note that the components shown here do not associate to specific task requirements. The LfD model parameters are exclusively capturing the spatial and temporal patterns of the demonstrations, but later adapted according to new task requirements.

the Gaussian component 1. Similarly, μ_3^S has to be moved to the left, thus reducing the time to stay in the corresponding component. Investigating the skill model more closely, the aforementioned results are reasonable as the component 3 is located near L_2, so reducing μ_3^S and increasing A_{24} transition probability will skip that region as depicted in Fig. 7.

The foregoing results show that optimizing LfD parameters using STL specifications can accommodate temporal constraints like staying in the region L_2 for 5 seconds. Similarly, satisfying spatial properties like visiting a new region L_4 by changing the trajectory can be achieved.

B. Robot Experiment

We designed a robot experiment to test our approach on an industrial assembly task as shown in Fig. 8. The robot has to pick up a shaft, re-orient it, and insert it into a specific location. We show that our approach not only satisfies the task but also attempts to find the most robust solution. We defined two different skills: One to pick up the object at a fixed position, and the other to re-orient and insert it into the desired location. We trained two HSMM models (by setting $K = 6$) with one demonstration for each skill. Finally, we created a behaviour by sequentially combining the aforementioned skills with end-effector grasp and release actions. The BO optimization was carried out for 16 iterations.

Let us consider the task: Pick up the object within 20 seconds and reduce the contact forces to less than 2N. The

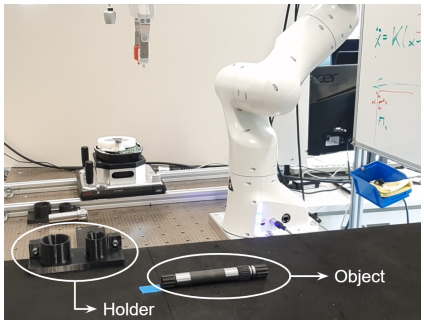

Fig. 8. Robot setup with a 7-DOF robot manipulator on an industrial assembly setup. The table has an object (a shaft) and a holder to place it.

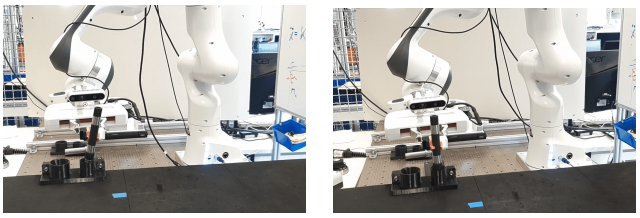

Fig. 9. Object insertion before (left) and after (right) optimization.

STL specification φ_3 for this task is,

$$\begin{aligned} \varphi_3 =& \mathbf{F}_{[0,20]}\varphi_{obj} \wedge \mathbf{F}\varphi_{force}, \\ \varphi_{obj} =& x_{obj,lb} < x < x_{obj,ub} \ \wedge \ y_{obj,lb} < y < y_{obj,ub} \ \wedge \\ & z_{obj,lb} < z < z_{obj,ub}, \\ \varphi_{force} =& |F_{contact}| < 2.0. \end{aligned} \tag{12}$$

The optimization parameters are the transition matrix $\mathbf{A}_{red}$ and duration probabilities $\boldsymbol{\mu}_{1:6}^S$ of the picking skill and the component position μ_6 of the insertion skill. We considered the norm of the contact forces as another signal along with the end-effector trajectory. We got access to contact forces using a 6D force-torque sensor mounted at the robot end-effector.

Figure 9-left shows the initially learned LfD skill, which is sub-optimal due to the shift in the end-effector position and the holder. However, insertion after refinement, as shown in Fig. 9-right overcomes such shift so that the object does not hit the edges of the holder. This in turn reduces the contact forces as shown in Fig. 10. Note that there is no time constraint for φ_{force} and in Fig. 10. The contact force is less than 2N at the beginning, which means, the Eventually operator is already positive. Albeit, the optimization finds a robust trajectory due to the property of robustness degree to guide towards task satisfaction. Note that to achieve similar results without our approach, we may need a force-sensitive LfD framework to specifically consider the required force patterns during the demonstration phase of the insertion task.

Figure 11 depicts the change in duration probabilities at each iteration. It can be observed that the number of components approaches $K = 3$ instead of $K = 6$ during optimization, which means several components have been skipped due to the modifications of the transition matrix $\boldsymbol{A}$. This can be interpreted as the skill being simple enough to be

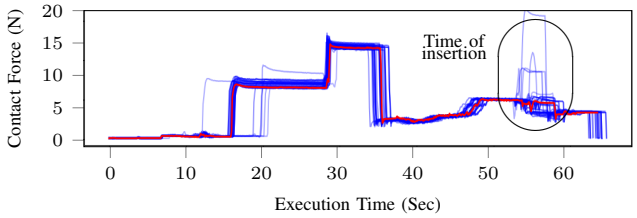

Fig. 10. Time series of the norm of the contact forces during the entire trajectory. The color intensity represents the optimization evolution. The circled time period ($52 - 58$ sec) corresponds to the image frames in Fig 9.

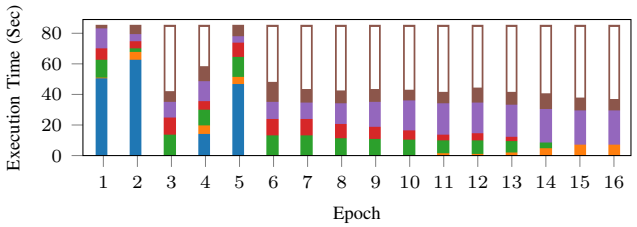

Fig. 11. Stacked bar char depicting the duration mean at each iteration. Each color represents a component of the insertion skill. The y-axis shows the duration of each component's trajectory, and the white stack represents the reduced time.

defined with fewer components. Note that the approach does not modify the total duration of the skill but instead changes the duration to stay in each component. Accordingly, the white bars in Fig. 11 represent the time at the end during which the end-effector does not move anymore, and we can remove that time from the trajectory if needed. Overall, this shows the capability of our optimization approach to ensure more reliable and faster satisfaction of the stated STL specification.

V. Conclusion

We presented an algorithm to include formal task requirements in an LfD model which are hard to specify implicitly by demonstration. Instead, they are defined formally as an STL specification and then, the model parameters are optimized to accommodate the additional requirements. Robot Experiments indicated that our approach of combining STL and BO could capture spatial and temporal constraints and find optimum trajectories for the task. Some benefits of our approach are due to BO: we do not need to explicitly model the environment nor differentiable objectives to adapt the skill. Regarding STL, it allows us to define a broad variety of spatial and temporal task requirements in a user-friendly manner. Future work will leverage nested STL operators and address two well-known problems in BO: the curse of dimensionality [16] and the geometry of the parameter space, the latter naturally arising in several robotic applications [26].

References

[1] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *International Conference on Machine Learning*, pp. 12–20, 1997. 1

[2] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 297–330, 2020. 1

[3] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, 2018. 1

[4] S. Adams, T. Cody, and P. A. Beling, "A survey of inverse reinforcement learning," *Artificial Intelligence Review volume*, vol. 55, pp. 4307–4346, 2022. 1

[5] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *American Control Conference (ACC)*, pp. 1690–1695, 2019. 1, 2

[6] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *IEEE Conference on Decision and Control*, pp. 6565–6570, 2016. 1, 2

[7] B. Araki, K. Vodrahalli, T. Leech, C.-I. Vasile, M. D. Donahue, and D. L. Rus, "Learning to plan with logical automata," in *Robotics: Science and Systems (R:SS)*, 2019. 1

[8] Z. Xu and U. Topcu, "Transfer of temporal logic formulas in reinforcement learning," in *IJCAI*, vol. 28, p. 4010, 2019. 1

[9] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839, 2017. 1

[10] C. Innes and S. Ramamoorthy, "Elaborating on learned demonstrations with temporal logic specifications," in *Robotics: Science and System (R:SS)*, 2020. 1

[11] A. G. Puranic, J. V. Deshmukh, and S. Nikolaidis, "Learning from demonstrations using signal temporal logic in stochastic and continuous domains," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 4, pp. 6250–6257, 2021. 1

[12] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019. 2

[13] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, "Two decades of blackbox optimization applications," *EURO Journal on Computational Optimization*, vol. 9, 2021. 3

[14] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon, "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020," in *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, vol. 133, pp. 3–26, 2021. 3

[15] S. Watanabe and J. Le Roux, "Black box optimization for automatic speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3256–3260, 2014. 3

[16] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010. 3, 8

[17] H. J. Kushner, "A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise," in *Basic Engineering*, vol. 86, pp. 97–106, 1964. 3

[18] A. Kumar Tanwani, J. Lee, B. Thananjeyan, M. Laskey, S. Krishnan, R. Fox, K. Goldberg, and S. Calinon, "Generalizing robot imitation learning with invariant hidden semi-Markov models," in *Workshop on the Algorithmic Foundations of Robotics*, 2018. 3

[19] L. Rozo, M. Guo, A. G. Kupcsik, M. Todescato, P. Schillinger, M. Giftthaler, M. Ochs, M. Spies, N. Waniek, P. Kesper, *et al.*, "Learning and sequencing of object-centric manipulation skills for industrial tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9072–9079, 2020. 3, 4

[20] S.-Z. Yu, "Hidden semi-Markov models," *Artificial intelligence, Special Review Issue*, vol. 174, no. 2, pp. 215–243, 2010. 3

[21] L. Rozo, "Interactive trajectory adaptation through force-guided Bayesian optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7596–7603, 2019. 3

[22] P. Varnai and D. V. Dimarogonas, "On robustness metrics for learning STL tasks," in *American Control Conference (ACC)*, pp. 5394–5399, 2020. 4

[23] A. Dhonthi, P. Schillinger, L. Rozo, and D. Nardi, "Study of signal temporal logic robustness metrics for robotic tasks optimization," *arXiv preprint arXiv:2110.00339*, 2021. 4

[24] P. Varnai and D. V. Dimarogonas, "On robustness metrics for learning stl tasks," *arXiv preprint arXiv:2003.06041*, 2020. 4

[25] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning*, pp. 278–287, 1999. 4

[26] N. Jaquier, V. Borovitskiy, A. Smolensky, A. Terenin, T. Asfour, and L. Rozo, "Geometry-aware Bayesian optimization in robotics using Riemannian Matn kernels," in *Conference on Robot Learning (CoRL)*, pp. 794–805, 2021. 8