Trajectory Planning with Signal Temporal Logic Costs using Deterministic Path Integral Optimization

Patrick Halder[1,3,*], Hannes Homburger[2,*], Lothar Kiltz[1], Johannes Reuter[2], and Matthias Althoff[3]

***Abstract*— Formulating the intended behavior of a dynamic system can be challenging. Signal temporal logic (STL) is frequently used for this purpose due to its suitability in formalizing comprehensible, modular, and versatile spatio-temporal specifications. Due to scaling issues with respect to the complexity of the specifications and the potential occurrence of non-differentiable terms, classical optimization methods often solve STL-based problems inefficiently. Smoothing and approximation techniques can alleviate these issues but require changing the optimization problem. This paper proposes a novel sampling-based method based on model predictive path integral control to solve optimal control problems with STL cost functions. We demonstrate the effectiveness of our method on benchmark motion planning problems and compare its performance with state-of-the-art methods. The results show that our method efficiently solves optimal control problems with STL costs.**

I. Introduction

Rich specifications are essential for delineating the desired behavior of dynamic systems [1]. Formal specification languages, such as signal temporal logic (STL), provide a precise yet interpretable way of describing the desired behavior, making them particularly attractive in robotics and autonomous driving [2]. Due to the inherent robustness measure, the level of satisfaction of an STL specification can straightforwardly be considered in optimal control problems. However, solving such problems often presents challenges due to a) scalability issues when STL specifications are deeply nested or contain long time intervals and b) potentially non-differentiable terms in the STL robustness measure. We address these issues by introducing a sequential solving method based on the path integral (PI) control framework [3]. Our approach converges quickly by sampling in the input space and gradually reducing the sampling variance, as illustrated in Fig. 1.

A. Related Work

Subsequently, we review related work on optimization with STL and model predictive path integral control.

[1]ZF Friedrichshafen AG, 88046 Friedrichshafen, Germany. {patrick.halder, lothar.kiltz}@zf.com

[2]Institute of System Dynamics, HTWG Konstanz, 78562 Konstanz, Germany. {hhomburg, jreuter}@htwg-konstanz.de

[3]School of Computation, Information and Technology, Technical University of Munich, 85748 Garching, Germany. {patrick.halder, althoff}@tum.de

*Both authors contributed equally to the paper.

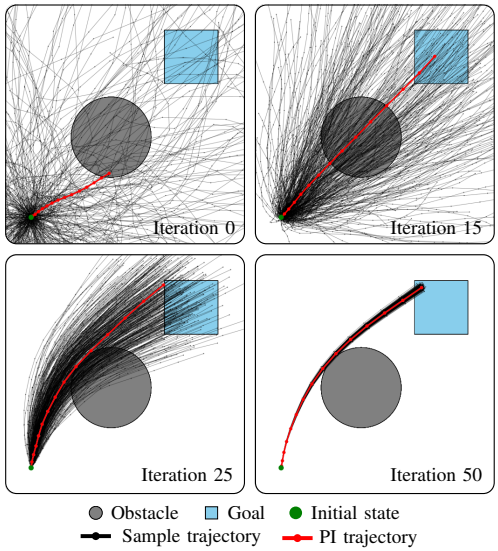

Fig. 1. Example of our PI-based solution for solving an optimal control problem with STL costs. The objective is to reach the blue box *eventually* while ensuring that the gray circular obstacle is avoided *at all times*. Four intermediate solutions are presented.

a) Optimization with STL: STL is well suited for formalizing spatio-temporal requirements in motion planning since it provides a precise, mathematical way of defining the desired behavior but remains interpretable by humans [2], [4]. STL provides a robustness semantic, quantifying the degree to which a formula is satisfied or violated [5]. A respective robustness function can be automatically derived, eliminating the need for manual design. STL specifications are usually integrated into optimization problems by including the robustness in the constraints or considering it in the objective as a penalty. However, solving these problems is challenging, e.g., due to the non-differentiable terms in the robustness function [6].

STL robustness is typically optimized using classical optimization techniques [7]. Mixed-integer encodings of the robustness function are mainly utilized (e.g., see [8]–[10]), but gradient-based methods [11]–[13] are also often applied. Recently, control barrier function approaches [14]–[17] and funnel approaches [18], [19] have also gained popularity. However, several problems arise with the aforementioned methods. Especially for mixed-integer encodings, scalability with respect to the time horizon and the nesting of the STL formula is a significant issue, as the number of constraints increases exponentially [6].

For gradient-based and control barrier function methods, the non-differentiable terms of the robustness function present another big challenge because of potentially van-

ishing gradients [20]. Thus, these approaches often require significantly modifying the original optimization problem, e.g., by restricting to a fragment of STL [19], limiting to convex predicates [6], [21], or using a modified smoothed robustness definition (see [22] for a comparison). STL is also used in learning-based optimization methods [20], which are also affected by the same problems. Sampling-based [23], [24] and evolutionary methods [25] are barely used for optimization with STL constraints, possibly because they only provide asymptotic optimality and may suffer from slow convergence rates.

b) MPPI Control: Model predictive path integral (MPPI) control [3] is a sampling-based approach that is suitable for trajectory planning and model predictive control. In contrast to most first- and second-order optimization methods employed for model predictive control [26], [27], MPPI can be straightforwardly applied to systems with non-differentiable costs and dynamics, is easy to implement, and can be inherently executed in parallel [28].

However, similar to other sampling-based methods, like reward-weighted regression [29], or the covariance matrix adaptation evolution strategy (CMA-ES) algorithm [30], MPPI can struggle in high-dimensional problems. To overcome this, recent extensions focus on improving the sampling distribution, e.g., by improving the convergence employing gradient descent updates [31], adaptive importance sampling [32], learning input distributions to obtain low-cost samples [33], covariance steering [34], handcrafted skills [35], or ancillary controllers [36]. MPPI can be improved by combining it with iterative linear-quadratic Gaussian [37] or differential dynamic programming [38].

While MPPI was applied initially to control unmanned aerial vehicles [39] and an intelligent race car [3], [28], [40], recently, it was also intensively used by the robotics community to control systems, such as autonomous surface vessels [41], [42], autonomous underwater vehicles [43], robotic manipulators [44], [45], four-legged walking robots [46], and lab experiments like the Furuta pendulum [36], [47]. A fast GPU implementation called `MPPI-Generic` is presented in [48]. For a comprehensive overview of recent methodological contributions and applications of MPPI, the interested reader is referred to [49].

B. Contributions

This paper presents a novel PI-based approach for solving deterministic, discrete-time optimal control problems with finite horizon and STL costs. In detail, our contributions are:

- enabling a simple task formalization and planning process by combining STL and MPPI control;
- providing an algorithm to determine the solution for deterministic discrete-time finite-horizon nonlinear optimal control problems with arbitrary STL costs; and
- comparing our approach numerically with state-of-the-art methods for trajectory planning with STL costs.

The remainder of this paper is organized as follows: we formally define our system, classical MPPI, and STL in Sec. II, introduce the problem statement in Sec. III, and provide our solution in Sec. IV. Numerical experiments are presented in Sec. V, and we conclude in Sec. VI.

II. Preliminaries

A. System Definition and Projection Operator

Let $k \in \mathbb{N}_0$ be a discrete time step and $t_k := k\Delta t$ the corresponding continuous time, with $\Delta t \in \mathbb{R}_{>0}$. Without loss of generality, the initial time step is $k_0 = 0$, and the final time step is $K \in \mathbb{N}_0$. Further, let $\mathcal{K} := [0, K] \subseteq \mathbb{N}_0$ be a discrete time interval. We consider a deterministic nonlinear discrete-time system

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ represent the state and input, respectively, with $k \in \mathcal{K}$. A solution of (1) for an initial state $x_0 \in \mathbb{R}^{n_x}$ and an input trajectory $\boldsymbol{u} = [u_0, u_1, \dots, u_{K-1}]$ at time step $k \in \mathcal{K}$ is denoted by $\chi(x_0, \boldsymbol{u}, k)$ and subsequently, we use x_k as its shorthand notation. The state trajectory is denoted by $\boldsymbol{x} = [x_0, x_1, \dots, x_K]$.

Further, we introduce the projection operator $\mathrm{proj}_{\square} : \Theta \to \mathbb{R}$ which maps a vector $\varrho \in \Theta$ to its element specified by $\square$, and $\mathbf{0}_{n_0}$ is a vector of zeros of length $n_0 \in \mathbb{N}_{>0}$.

B. Signal Temporal Logic

In this work, we use the future-time fragment of STL; however, our approach is also analogously applicable to the past-time fragment. The syntax of the future-time fragment of STL over a state trajectory $\boldsymbol{x}$ is defined by the following grammar [5, Sec. 2.1]:

$$\varphi := \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbf{U}_I \varphi_2,$$

where μ is a predicate of the form $\mu := b(\boldsymbol{x}, k) \geq 0$, with $b : \mathbb{R}^{n_x \times (K+1)} \times \mathbb{N}_0 \to \mathbb{R}$. The expressions φ, φ_1, and φ_2 are STL formulas, and $\neg$ and $\wedge$ are negation and conjunction, respectively. Disjunction can be expressed as $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. The temporal operator $\mathbf{U}_I$ specifies that φ_1 holds *until* φ_2 holds in the time interval $[k_{\min}, k_{\max}]$, where $k_{\min}, k_{\max} \in \mathbb{N}_0$ and $k_{\max} \geq k_{\min}$. Further temporal operators can be derived, such as $\mathbf{F}_I\varphi := \mathrm{True}\mathbf{U}_I\varphi$ (*eventually*) and $\mathbf{G}_I\varphi := \neg\mathbf{F}_I\neg\varphi$ (*globally*). The satisfaction of an STL formula φ by a trajectory $\boldsymbol{x}$ at time step k is denoted as $(\boldsymbol{x}, k) \models \varphi$.

Subsequently, we present the definition of *space robustness* for the operators used in this paper. The space robustness $\rho^{\varphi}(\boldsymbol{x}, k) \in \mathbb{R}$ of an STL formula φ provides a measure expressing its degree of compliance or violation for a finite trajectory $\boldsymbol{x}$ at time step $k \in \mathcal{K}$. The robustness $\rho^{\varphi}(\boldsymbol{x}, k)$ is positive iff $\boldsymbol{x} \models \varphi$. The space robustness is formally defined as [5, Sec. 2.2]:

$$\begin{aligned}
\rho^{\mathrm{True}}(\boldsymbol{x}, k) &:= \infty, \\
\rho^{b(\boldsymbol{x},k)\geq 0}(\boldsymbol{x}, k) &:= b(\boldsymbol{x}, k), \\
\rho^{\neg\varphi}(\boldsymbol{x}, k) &:= -\rho^{\varphi}(\boldsymbol{x}, k), \\
\rho^{\varphi_1 \wedge \varphi_2}(\boldsymbol{x}, k) &:= \min\left(\rho^{\varphi_1}(\boldsymbol{x}, k), \rho^{\varphi_2}(\boldsymbol{x}, k)\right), \\
\rho^{\mathbf{G}_I\varphi}(\boldsymbol{x}, k) &:= \min_{k' \in [k+k_{\min}, k+k_{\max}] \cap \mathcal{K}} \left(\rho^{\varphi}(\boldsymbol{x}, k')\right), \\
\rho^{\mathbf{F}_I\varphi}(\boldsymbol{x}, k) &:= \max_{k' \in [k+k_{\min}, k+k_{\max}] \cap \mathcal{K}} \left(\rho^{\varphi}(\boldsymbol{x}, k')\right).
\end{aligned}$$

C. Classical MPPI

Let us recall the basic ideas of classical MPPI control derived from the information-theoretic framework [3]. MPPI considers the stochastic open-loop optimal control problem

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \mathbb{E}_{\mathbb{Q}_{\boldsymbol{u}}} \left[\sum_{k=0}^{K-1} L(x_k, u_k) + E(x_K) \right], \quad (2)$$

where a disturbed state trajectory is given by $\boldsymbol{x}(x_0, \boldsymbol{v}, k)$ for $k \in \mathcal{K}$, originating from the disturbed input sequence $\boldsymbol{v} = [v_0, v_1, \ldots, v_{K-1}]$, with $v_k \sim \mathcal{N}(u_k, \Sigma)$. The covariance matrix is denoted by $\Sigma \in \mathbb{R}^{n_u \times n_u}$, $\mathbb{Q}_{\boldsymbol{u}}$ is the Gaussian distribution of the sequence $\boldsymbol{v}$ with mean $\boldsymbol{u}$, the corresponding probability density function is $q(\boldsymbol{v} \,|\, \boldsymbol{u})$, and $\mathbb{E}_{\mathbb{Q}_{\boldsymbol{u}}}[\cdot]$ denotes the expectation under distribution $\mathbb{Q}_{\boldsymbol{u}}$. Further, $E : \mathbb{R}^{n_x} \to \mathbb{R}_{>0}$ is the terminal cost function and the stage cost function $L(x_k, u_k)$ is chosen as

$$L(x_k, u_k) := c(x_k) + \frac{1}{2} u_k^\top R u_k, \quad (3)$$

where $c : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$ and the system input is penalized quadratically with $R := \lambda \Sigma^{-1}$, where $\lambda \in \mathbb{R}_{>0}$ is a tuning parameter called the *inverse temperature* [3, Sec. 2].

Based on these assumptions, according to [3, Sec. 3.A], the optimal control sequence $\boldsymbol{u}^*$ that solves (2) is approximated by the so-called *information-theoretic* optimum $\tilde{\boldsymbol{u}}^*$:

$$\boldsymbol{u}^* \approx \arg\min_{\boldsymbol{u}} \mathbb{E}_{\mathbb{Q}^*} \left[\log \left(\frac{q^*(\boldsymbol{v})}{q(\boldsymbol{v} \,|\, \boldsymbol{u})} \right) \right] =: \tilde{\boldsymbol{u}}^*, \quad (4)$$

with the optimal distribution defined by its probability density function

$$q^*(\boldsymbol{v}) := \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S(\boldsymbol{v}) \right) q(\boldsymbol{v} \,|\, \boldsymbol{0}_{Kn_u}), \quad (5)$$

where $\eta \in \mathbb{R}_{>0}$ is a normalization constant, and the path costs are

$$S(\boldsymbol{v}) := E(x_K) + \sum_{k=0}^{K-1} c(x_k).$$

The equation of the information-theoretic optimum in (4) can be simplified by disregarding constant terms and using the expected value of a random variable from a distribution $\mathbb{Q}_{\hat{u}}$, where $\hat{\boldsymbol{u}}$ is a proposed control sequence, along with a correction term $w(\boldsymbol{v}, \hat{\boldsymbol{u}}) \in \mathbb{R}_{>0}$ (see [3, Sec. 3] for details). This is then given by

$$\tilde{u}_k^* = \mathbb{E}_{\mathbb{Q}^\star}[v_k] = \mathbb{E}_{\mathbb{Q}_{\hat{\boldsymbol{u}}}}[w(\boldsymbol{v}, \hat{\boldsymbol{u}}) v_k] \quad (6)$$

for $k \in \mathcal{K}$. It is common to estimate the so-called optimal *information-theoretic* control law (6) by Monte Carlo sampling of trajectories.

III. Problem Statement

Let us first define a robustness function $\mathrm{rob} : \mathbb{R}^{(K+1)n_x} \to \mathbb{R}$. Two prevalent choices are 1) to either maximize the satisfaction of φ or 2) to impose no costs if φ is satisfied and to minimize its violation otherwise:

$$\mathrm{rob}^{\varphi}(\boldsymbol{x}) := -\rho^{\varphi}(\boldsymbol{x}, 0), \quad (7a)$$

$$\mathrm{rob}^{\varphi}(\boldsymbol{x}) := -\min(0, \rho^{\varphi}(\boldsymbol{x}, 0)). \quad (7b)$$

We solve the nonlinear program typically arising from the direct single shooting discretization of an optimal control problem, with an additional STL cost term rob as

$$\min_{u_0, \ldots, u_{K-1}} \gamma \, \mathrm{rob}^{\varphi}(\boldsymbol{x}) + \sum_{k=0}^{K-1} L(x_k, u_k) + E(x_K), \quad (8)$$

where $\gamma \in \mathbb{R}_{>0}$ is a weight. We call (8) an STL-OCP. The challenge in solving this problem stems from the non-differentiable and non-convex function rob. However, many state-of-the-art methods (e.g., see [26], [27]) require differentiable and convex components for a good performance. The feasibility of (8) is always guaranteed since it is an unconstrained problem. Subsequently, we present our solution method to solve problem (8).

IV. Solution

As presented in Sec. I-A, MPPI is a prevalent sampling-based method to solve optimal control problems, which is straightforward to implement, can be executed in parallel, and can handle non-differentiable costs and dynamics. Motivated by these properties, we develop a PI-based solution method for the STL-OCP (8).

Considering the STL-OCP (8) compared to the standard MPPI optimal control problem (2), two differences occur: the additional STL cost term rob, which depends on the entire state trajectory, and the deterministic problem setting. We thus show subsequently how the STL-OCP (8) can be transformed into the required form, how the exact solution for our deterministic problem can be determined using the stochastic PI method, and finally, we present our solution algorithm.

A. Problem Transformation

The STL-OCP (8) cannot be directly solved with MPPI since the standard discrete-time Bolza form [50, Sec. 1] is required. Therefore, we introduce an augmented state $\tilde{x}_k \in \mathbb{R}^{(K+1)n_x}$ and a corresponding augmented system dynamics $\tilde{f} : \mathbb{R}^{(K+1)n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{(K+1)n_x}$, defined as

$$\tilde{x}_k := \begin{bmatrix} x_k \\ x_{k-1} \\ \vdots \\ x_0 \\ \boldsymbol{0}_{(K-k)n_x} \end{bmatrix}, \; \tilde{f}(\tilde{x}_k, u_k) := \begin{bmatrix} f(x_k, u_k) \\ x_k \\ x_{k-1} \\ \vdots \\ x_0 \\ \boldsymbol{0}_{(K-k-1)n_x} \end{bmatrix}. \quad (9)$$

The augmented state $\tilde{x}_k$ concatenates the original states for all time steps up to the current time step k. The zero vectors assure constant state dimensions. The cost terms are defined as

$$\begin{aligned} \tilde{L}(\tilde{x}_k, u_k) &:= L(x_k, u_k), \\ \tilde{c}(\tilde{x}_k) &:= c(x_k), \\ \tilde{E}(\tilde{x}_K) &:= E(x_K) + \gamma \, \mathrm{rob}^{\varphi}(\tilde{x}_K), \end{aligned}$$

where $\tilde{x}_K \equiv \boldsymbol{x}$ is exploited. Now, (8) can be expressed in standard discrete-time Bolza form as

$$\min_{u_0, \ldots, u_{K-1}} \sum_{k=0}^{K-1} \tilde{L}(\tilde{x}_k, u_k) + \tilde{E}(\tilde{x}_K) \quad (10)$$

based on the state augmentation (9). The new representation (10) has the desired form, but it comes with the drawback of a high-dimensional state space. As we will see later, this does not pose a problem for our solution approach since trajectories are only sampled in the input space.

B. Exact Solution for Deterministic Problems with MPPI

While the standard MPPI algorithm in [3] generally provides a biased solution to the open-loop stochastic optimal control problem (4) [51, Sec. 3.B], we now present a method to obtain the unbiased solution for the deterministic STL-OCP (10) by gradually reducing the uncertainty Σ and the inverse temperature λ.

Assuming there is a unique global minimizer and considering the optimal distribution (5), where we substitute the probability density function $q(\boldsymbol{v}\,|\mathbf{0}_{Kn_u})$ by its explicit expression, we obtain

$$\begin{aligned} q^*(\boldsymbol{v}) &= \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}S(\boldsymbol{v})\right)\frac{1}{Z^K}\prod_{k=0}^{K-1}\exp\left(-\frac{1}{2}v_k^\top\Sigma^{-1}v_k\right) \\ &= \frac{1}{\tilde{\eta}}\exp\left(-\frac{1}{\lambda}\left[S(\boldsymbol{v})+\frac{1}{2}\sum_{k=0}^{K-1}v_k^\top\lambda\Sigma^{-1}v_k\right]\right), \end{aligned}$$

where $Z = [(2\pi)^{n_u}\det(\Sigma)]^{\frac{1}{2}}$ is the normalization constant for the multivariate normal distribution. The second line originates from the definition of the assembled normalization constant $\tilde{\eta} = \eta Z^K$ and by applying the standard laws of exponents. Substituting λ by $\beta\lambda$ and Σ by $\beta\Sigma$, where $\beta \in (0,1)$ is a scaling factor, yields

$$q^*(\boldsymbol{v},\beta) = \frac{\exp\left(-\frac{1}{\beta\lambda}\left[S(\boldsymbol{v})+\frac{1}{2}\sum_{k=0}^{K-1}v_k^\top\lambda\Sigma^{-1}v_k\right]\right)}{\tilde{\eta}(\beta)}. \tag{11}$$

Note that the expression within the square brackets reflects the objective of (10). By definition, this expression is minimal for $\boldsymbol{v} \equiv \boldsymbol{u}^*$. By shrinking β, the effects of the suboptimal trajectories on $q^*(\boldsymbol{v},\beta)$ decrease, resulting in

$$\boldsymbol{u}^* = \lim_{\beta\to 0}\mathbb{E}_{\mathbb{Q}^*(\beta)}[\boldsymbol{v}] = \lim_{\beta\to 0}\tilde{\boldsymbol{u}}(\beta). \tag{12}$$

We refer the reader to [51] for a comprehensive analysis of the convergence properties. Subsequently, we present an algorithm to numerically compute $\boldsymbol{u}^*$ based on a progressive reduction of β.

C. Algorithm

Alg. 1 solves problem (8). In contrast to the standard MPPI algorithm [28], the covariance Σ and the inverse temperature λ gradually decrease within $J \in \mathbb{N}_{>0}$ iterations by multiplying with the shrinking factor $\nu \in (0,1)$ (see Lines 1, 16 and 17). This approximates the weak limit of $\beta \to 0$ in (12). In each iteration, $M \in \mathbb{N}_{>0}$ trajectories are simulated, and their corresponding costs are evaluated (see Lines 2 to 9). The correction term $\lambda\epsilon_k^{m\top}\Sigma^{-1}\hat{u}_k$ in Line 8 corresponds to the importance weighting of MPPI (cf. (6)). The solution trajectory for one iteration is determined using the weighted sum of the sampled input trajectories (see Lines 10 to 15). The output of the algorithm is the solution of the optimal control problem for a reduced covariance $\Sigma\nu^J \approx 0$.

Alg. 1 is characterized by gradually reducing the input uncertainty, allowing for a good tradeoff between an intense exploration of the solution space and converging fast towards a local minimum. The algorithm scales linearly with the product of the number of shrinking iterations J, the number of trajectory samples M, and the problem horizon K, where M is typically large. By using a parallel implementation of the loop in Lines 1 to 9, we can alleviate the runtime dependency on M, which usually dominates the computational cost of Alg. 1. The algorithm can be terminated early for real-time applications with limited runtime since intermediate solutions are computed.

While the standard MPPI algorithm [3] or the deterministic `MPPI-Generic` algorithm [48] provide suboptimal solutions, Alg. 1 solves the deterministic STL-OCP (8) up to an arbitrary accuracy [51, Thm. 1]. The convergence of Alg. 1 as $M \to \infty$ and $J \to \infty$ is proven in [51, Thm. 1]. Contrary to the standard MPPI, Σ and λ are hyper-parameters without physical meaning. In contrast to other Monte-Carlo methods like, e.g., reward-weighted regression [29] and CMA-ES [30], our approach only adapts the mean of the sample distribution, whereas the exponential shrinking of the covariance is predefined.

V. Numerical Experiments

In this section, STL motion planning problems are solved numerically to evaluate the performance of the PI approach

Algorithm 1 Deterministic STL-OCP Path Integral Solver

Input: Initial system state x_0, initial control sequence $\hat{\boldsymbol{u}}$, number J of iterations, number M of trajectory samples, shrinking factor ν, initial covariance Σ, initial inverse temperature λ
Output: Optimal input trajectory $\boldsymbol{u}^*$, optimal state trajectory $\boldsymbol{x}^*$

1: **for** $j \in \{1,2,\dots,J\}$ **do**
2: **for** $m \in \{1,2,\dots,M\}$ **do** ▷ *parallelizable*
3: $\tilde{x}_0^m \leftarrow x_0$
4: $S^m \leftarrow 0$
5: **for** $k \in \{0,1,\dots,K-1\}$ **do**
6: $\epsilon_k^m \sim \mathcal{N}(0,\Sigma)$
7: $\tilde{x}_{k+1}^m \leftarrow \tilde{f}(\tilde{x}_k^m, \hat{u}_k + \epsilon_k^m)$
8: $S^m \leftarrow S^m + \tilde{c}(\tilde{x}_k^m) + \lambda\epsilon_k^{m\top}\Sigma^{-1}\hat{u}_k$
9: $S^m \leftarrow S^m + \tilde{E}(\tilde{x}_K^m)$
10: $\psi \leftarrow \min_{m\in\{1,2,\dots,M\}} S^m$
11: $\eta \leftarrow \sum_{m=1}^M \exp(-\frac{1}{\lambda}(S^m - \psi))$
12: **for** $m \in \{1,2,\dots,M\}$ **do**
13: $\omega^m \leftarrow \frac{1}{\eta}\exp(-\frac{1}{\lambda}(S^m - \psi))$
14: **for** $k \in \{0,1,\dots,K-1\}$ **do**
15: $\hat{u}_k \leftarrow \hat{u}_k + \sum_{m=1}^M \omega^m\epsilon_k^m$
16: $\lambda \leftarrow \nu\lambda$
17: $\Sigma \leftarrow \nu\Sigma$
18: $\boldsymbol{u}^* \leftarrow \hat{\boldsymbol{u}}, x_0^* \leftarrow x_0$
19: **for** $k \in \{0,1,\dots,K-1\}$ **do**
20: $x_{k+1}^* \leftarrow f(x_k^*, u_k^*)$
21: **return** $\boldsymbol{u}^*, \boldsymbol{x}^*$

TABLE I

OPTIMIZED HYPERPARAMETERS OF THE USED SOLVERS.

Solver	Parameters	Pr. I	Pr. II	Pr. III
MIP	all parameters		- defaults -	
GRAD	`maxiter`	100	100	1e5
	`ftol`	7.0e−5	2.2e−7	1e−6
	`eps`	4.7e−5	1.6e−7	1.49e−8
SGRAD	`maxiter`	100	30	1e5
	`ftol`	3.4e−6	6.7e−7	1e−6
	`eps`	1.5e−5	4.8e−7	1.49e−8
	`k_1, k_2`	186	490	400
CMA-ES	`sigma`	0.03	0.037	0.022
	`maxfevals`	4060	9300	1e6
	`noise_change_sigma_exponent`	0.887	0.555	0.644
	`pop_size`	17	13	35
	`n_iter`	2140	773	11240
PI	J	19	75	40
	M	955	1140	81650
	Σ	5.6	$\begin{bmatrix} 3.4 & 0.0 \\ 0.0 & 3.4 \end{bmatrix}$	$\begin{bmatrix} 0.002 & 0.0 \\ 0.0 & 0.002 \end{bmatrix}$
	λ	11.2	60.8	0.2
	ν	0.3	0.8	0.8

and to compare it with state-of-the-art solvers. In particular, we compare the following solvers:

- **MIP**: Mixed-integer encoding of robustness, typically used for linear systems [7], but also applicable to nonlinear systems, solved with `Gurobi` [52];
- **GRAD**: Gradient-based solver using the `SLSQP` method from `SciPy` [53];
- **SGRAD**: Smooth approximation of robustness [54]; solved with `SLSQP` from `SciPy` [53];
- **CMA-ES**: Adaptive evolutionary solver from [55];
- **PI**: The approach of this paper; see Alg. 1.

Our algorithm is implemented in C++, and the experiment setup is based on the STL-benchmark tool `stlpy` [6]. For comparability, we extensively optimize the hyperparameters of each solver with `Optuna` [56], by minimizing the objective $C^* + \vartheta t_\mathrm{c}^*$, where C^* is the optimized cost value of (8), t_c^* is the convergence time, and ϑ is a tuning parameter, with $\vartheta_{\text{Prob. I}} = 0.1$, $\vartheta_{\text{Prob. II}} = 1.0$, and $\vartheta_{\text{Prob. III}} = 0.1$. We terminate after `n_trials` $= 1000$. The hyperparameters of the **MIP** solver are optimized using the tuning tool of `Gurobi`, specifically designed for this purpose, and no improvements are detected compared to the default values. The optimized hyperparameters are listed in Tab. I.

For problem I and problem III, (7b) is the robustness cost function and (7a) is selected for problem II. For all experiments, we use $c(x_k) := 0$. The experiments are executed on an `AMD Ryzen™ 5 PRO 7540U CPU`. The code can be accessed at https://github.com/TUMcps/STL-PI-Planner.

A. Problem I: Scalar Integrator

Let us consider the following linear scalar discrete-time integrator with the dynamics $x_{k+1} = x_k + u_k$. We aim to maximize x_K at the end of the planning time horizon $K = 10$. Additionally, the state shall be below 1 for at least two not necessarily consecutive time steps. This problem could be formalized as a mixed-integer linear program;

TABLE II

OPTIMAL COSTS, ROBUSTNESS, AND CONVERGENCE TIMES.

Pr.		MIP	GRAD	SGRAD	CMA-ES	PI
I	C^*	-3.0	-0.25	-2.85	-2.99	-2.98
	ρ^{φ_1}	0.0	-2.5	0.0	0.0	0.0
	t_c^*	0.008 s	0.039 s	0.104 s	0.397 s	0.024 s
II	C^*	13.32	13.33	13.58	13.4	13.48
	ρ^{φ_2}	0.2	0.2	0.015	0.2	0.17
	t_c^*	1.16 s	1.27 s	0.34 s	1.62 s	0.12 s

however, specifying the required constraints is cumbersome, and solving mixed-integer problems is hard.

Hence, we formalize an STL formula and consider it in the cost function. We define $E(x_K) := -x_K$, and the STL formula is $\varphi_1 := F_{[0,K]}(\mu_\text{gate} \wedge F_{[1,K]}(\mu_\text{gate}))$, with $\mu_\text{gate} := 1 - x_k \geq 0$. The corresponding robustness function can be directly derived from the grammar shown in Sec. II-B as:

$$\rho^{\varphi_1}(\boldsymbol{x}, k) = \max_{k' \in [k, k+K] \cap \mathcal{K}} \Big(\min \big(1 - x_{k'}, \max_{k'' \in [k'+1, k'+K] \cap \mathcal{K}} (1 - x_{k''})\big)\Big).$$

This non-differentiable function is costly to evaluate and hard to maximize.

The solution of our **PI** solver is presented in Fig. 2. It satisfies φ_1 and x_K reaches a maximum. Fig. 2 visualizes the sampled trajectories for four iterations, illustrating the shrinking variance of the samples over the iterations. Table II presents the optimal costs C^*, the robustness ρ^φ, and the convergence time t_c^* for the solutions of the different solvers. All solvers, except the **GRAD** solver, provide similar optimal costs and do not violate φ_1.

B. Problem II: Simple Motion Planning

We now revise the motion planning problem from Fig. 1, where the goal is to eventually reach the blue rectangular area while avoiding a collision with the gray circular obstacle in the planning time horizon $K = 15$. A state is defined as $x := [\mathtt{p}_x, \mathtt{p}_y, \mathtt{v}_x, \mathtt{v}_y]$, where $\mathtt{p}_x, \mathtt{v}_x \in \mathbb{R}$ are the position and the velocity in x-direction, respectively, and $\mathtt{p}_y, \mathtt{v}_y \in \mathbb{R}$ are the position and the velocity in y-direction, respectively. The input is $u := [\mathtt{a}_x, \mathtt{a}_y]$, where $\mathtt{a}_x, \mathtt{a}_y \in \mathbb{R}$ are the accelerations

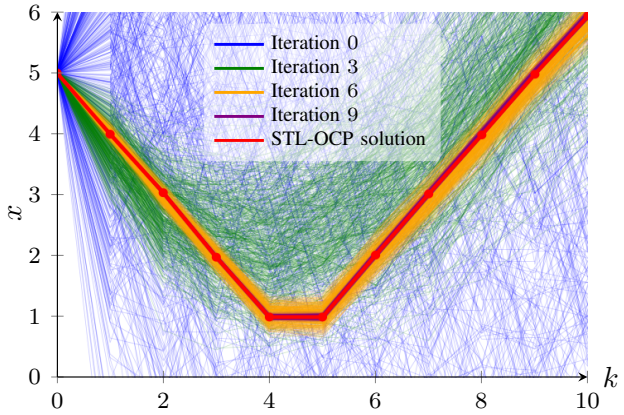

Fig. 2. STL-OCP solution of our PI solver for problem I. Also, the sampled trajectories of four iterations are presented.

in x-direction and y-direction. We use standard discrete-time double integrator dynamics. Using STL, the objective is formalized as follows:

$$\varphi_2 := G_{[0,K]}(\neg \mu_{\text{in\_circle}}) \wedge F_{[0,K]}(\varphi_{\text{in\_box}}), \text{ with}$$
$$\varphi_{\text{in\_box}} := \mu_{\text{above\_of}} \wedge \mu_{\text{below\_of}} \wedge \mu_{\text{right\_of}} \wedge \mu_{\text{left\_of}},$$

and the predicates are:

$$\begin{aligned}
\mu_{\text{in_circle}} &:= r^2 - (\text{proj}_{\mathtt{p}_x}(x_k) - m_x)^2 \\
&\quad - (\text{proj}_{\mathtt{p}_y}(x_k) - m_y)^2 \geq 0, \\
\mu_{\text{above_of}} &:= \text{proj}_{\mathtt{p}_y}(x_k) - \underline{\mathtt{p}}_y \geq 0, \\
\mu_{\text{below_of}} &:= \overline{\mathtt{p}}_y - \text{proj}_{\mathtt{p}_y}(x_k) \geq 0, \\
\mu_{\text{right_of}} &:= \text{proj}_{\mathtt{p}_x}(x_k) - \underline{\mathtt{p}}_x \geq 0, \\
\mu_{\text{left_of}} &:= \overline{\mathtt{p}}_x - \text{proj}_{\mathtt{p}_x}(x_k) \geq 0,
\end{aligned}$$

where $r, m_x, m_y, \underline{\mathtt{p}}_x, \overline{\mathtt{p}}_x, \underline{\mathtt{p}}_y, \overline{\mathtt{p}}_y \in \mathbb{R}$ are parameters. Due to $\mu_{\text{in\_circle}}$, problem II is nonlinear and has two distinct optimal solutions. The terminal costs are $E(x_K) := \text{proj}_{\mathtt{p}_x}(x_K) + \text{proj}_{\mathtt{p}_y}(x_K)$.

Again, a numerical comparison is provided in Tab II. All solvers find a solution that satisfies φ_2, and they achieve similar optimal costs. The increased complexity of problem II, compared to problem I, leads to a higher convergence time for all solvers besides our **PI** solver. The increased computation time for the **CMA-ES** solver likely indicates that this covariance-adaption approach has more difficulty deciding on one of the two possible solutions. Using a smooth STL cost term, the **SGRAD** solver converges faster than the **GRAD** solver.

C. Problem III: Complex Motion Planning

Finally, we present a challenging motion planning problem. The state is defined as $x := [\mathtt{p}_x, \mathtt{p}_y, \theta, \mathtt{v}, \Psi]^\top$, where $\mathtt{p}_x, \mathtt{p}_y$ are the positions in x-direction and y-direction, respectively, θ is the steering angle, $\mathtt{v}$ is the velocity, and Ψ is the orientation. The input is $u := \left[\mathtt{v}^\theta, \mathtt{a}\right]^\top$, where $\mathtt{v}^\theta$ is the steering velocity and $\mathtt{a}$ is the longitudinal acceleration. We consider a kinematic single-track model with an explicit Euler integration as: $x_{k+1} = x_k + \check{f}(x_k, u_k)\Delta t$, with

$$\check{f}(x_k, u_k) = \left[\mathtt{v}_k \cos(\Psi_k), \mathtt{v}_k \sin(\Psi_k), \mathtt{v}_k^\theta, \mathtt{a}_k, \frac{\mathtt{v}_k}{\ell} \tan(\theta_k)\right]^\top,$$

where $\ell \in \mathbb{R}$ is the wheelbase of the vehicle. The planning time horizon is $K = 50$.

We intend the system to a) remain in the area α, to b) not collide with any of the five obstacles $\mathcal{O}$, to c) stay at least for two consecutive time steps in each of the task areas $\tau \in \mathcal{T} := \{\tau_1, \tau_2, \tau_3\}$ (while they can be visited in arbitrary order), and to d) not be simultaneously at task area τ_1 and τ_2. In STL, this is formalized as follows:

$$\varphi_3 := \varphi_{\text{in\_area}} \wedge \varphi_{\text{no\_collision}} \wedge \varphi_{\text{do\_tasks}} \wedge \varphi_{\text{avoid\_overlap}}, \text{ with}$$
$$\begin{aligned}
\varphi_{\text{in_area}} &:= G_{[0,K]}\left(\varphi^\alpha_{\text{in_box}}\right), \\
\varphi_{\text{no_collision}} &:= G_{[0,K]}\left(\textstyle\bigwedge_{o \in \mathcal{O}} (\neg \varphi^o_{\text{in_box}})\right), \\
\varphi_{\text{do_tasks}} &:= \textstyle\bigwedge_{\tau \in \mathcal{T}} \left(F_{[0,K]}\left(G_{[0,1]}(\mu^\tau_{\text{in_circle}})\right)\right), \\
\varphi_{\text{avoid_overlap}} &:= G_{[0,K]}(\neg(\mu^{\tau_1}_{\text{in_circle}} \wedge \mu^{\tau_2}_{\text{in_circle}})).
\end{aligned}$$

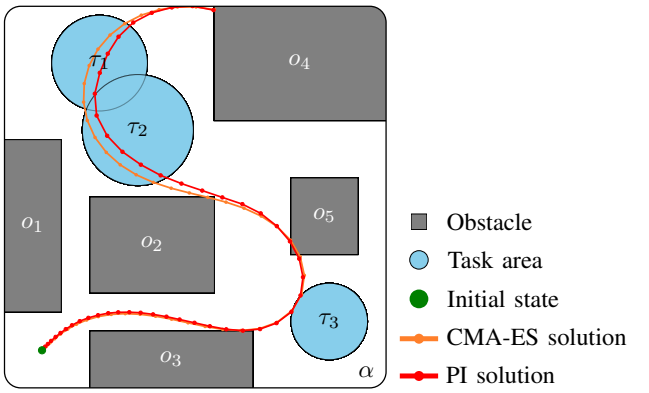

Fig. 3. Solutions for problem III. Note that we evaluate the robustness function only at discrete time steps.

The superscripts α, o, and τ indicate that the required parameters are provided to the respective subformulas and predicates. The terminal costs are $E(x_K) := \text{proj}_{\mathtt{p}_x}(x_K) + \text{proj}_{\mathtt{p}_y}(x_K)$.

The **PI** solver converged in 25.07 s, and it took 79.82 s for the **CMA-ES** solver. Their solutions satisfy φ_3 and are presented in Fig. 3. However, the remaining solvers did not converge within a limit of 3600 s. This indicates the capability of the sampling-based methods, especially of our PI approach, to solve complex STL-OCPs in adequate time without the need to reformulate the robustness term. The performance of the presented **PI** solver can be even more improved by utilizing a GPU. However, a comprehensive numerical benchmark with many problems and further state-of-the-art solvers is essential for future work.

VI. Conclusions

This paper introduces a novel sampling-based solution method for deterministic STL-OCPs within the PI framework. A key feature of this sampling-based approach is the gradual reduction of the problem variance, which enables the handling of non-differentiable problem formulations typically associated with STL. Additionally, the method maintains MPPI's inherent advantage of a straightforward implementation and parallel execution, making it suitable for real-time applications. Through numerical experiments, the performance of the method is demonstrated and compared to other state-of-the-art methods for solving STL-OCPs. The findings suggest that our approach holds promise for advancing the design of autonomous systems by efficiently addressing complex planning problems.

Future work will focus on extensively comparing the performance of our method with other state-of-the-art solvers and enhancing it to perform model predictive control with STL specifications. Additionally, we intend to extend the method to handle prioritized STL specifications.

Acknowledgment

The authors gratefully acknowledge financial support from the German Research Foundation (DFG) under grant AL 1185/19-1 and thank Moritz Diehl, Samuel Henkel, and Florian Messerer for the valuable discussions that contributed strongly to this paper.

REFERENCES

[1] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, 2023.

[2] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[3] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[4] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *Proc. of the ACM-IEEE Int. Conf. on Formal Methods and Models for Sys. Design*, 2019, pp. 1–11.

[5] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Springer, 2018, pp. 135–175.

[6] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Sys. Letters*, vol. 6, pp. 2635–2640, 2022.

[7] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Sys.*, vol. 2, no. 1, pp. 115–140, 2019.

[8] S. Sadraddini and C. Belta, "Formal synthesis of control strategies for positive monotone systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 2, pp. 480–495, 2019.

[9] Y. E. Sahin, R. Quirynen, and S. D. Cairano, "Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming," in *Proc. of the American Control Conf.*, 2020, pp. 454–459.

[10] G. A. Cardona and C.-I. Vasile, "Partial satisfaction of signal temporal logic specifications for coordination of multi-robot systems," in *Proc. of the Workshop on the Algorithmic Foundations of Robotics*, 2023, pp. 223–238.

[11] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Proc. of the IEEE Conf. on Control Technology and Applications*, 2017, pp. 1235–1240.

[12] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *Proc. of the American Control Conf.*, 2019, pp. 1690–1695.

[13] Y. Mao, B. Acikmese, P.-L. Garoche, and A. Chapoutot, "Successive convexification for optimal control with signal temporal logic specifications," in *Proc. of the ACM Int. Conf. on Hybrid Sys.: Computation and Control*, 2022, pp. 1–7.

[14] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Sys. Letters*, vol. 3, no. 1, pp. 96–101, 2019.

[15] ——, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Transactions on Control of Network Sys.*, vol. 7, no. 4, pp. 1916–1928, 2020.

[16] M. Charitidou and D. V. Dimarogonas, "Barrier function-based model predictive control under signal temporal logic specifications," in *Proc. of the European Control Conf.*, 2021, pp. 734–739.

[17] W. Xiao, N. Mehdipour, A. Collin, A. Bin-Nun, E. Frazzoli, R. D. Tebbens, and C. Belta, "Rule-based optimal control for autonomous driving," in *Proc. of the ACM/IEEE Int. Conf. on Cyber-Physical Sys.*, 2021, pp. 143–154.

[18] L. Lindemann, C. K. Verginis, and D. V. Dimarogonas, "Prescribed performance control for signal temporal logic specifications," in *Proc. of the IEEE Conf. on Decision and Control*, 2017, pp. 2997–3002.

[19] L. Lindemann and D. V. Dimarogonas, "Funnel control for fully actuated systems under a fragment of signal temporal logic specifications," *Nonlinear Analysis: Hybrid Sys.*, vol. 39, pp. 1–17, 2021.

[20] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," in *Proc. of the Workshop on the Algorithmic Foundations of Robotics*, 2021, pp. 432–449.

[21] P. Halder, F. Christ, and M. Althoff, "Lexicographic mixed-integer motion planning with STL constraints," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Sys.*, 2023, pp. 1361–1367.

[22] S. Welikala, H. Lin, and P. J. Antsaklis, "Smooth robustness measures for symbolic control via signal temporal logic," *arXiv preprint arXiv:2305.09116*, 2023.

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

[24] A. Linard, I. Torre, E. Bartoli, A. Sleat, I. Leite, and J. Tůmová, "Real-time RRT* with signal temporal logic preferences," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Sys.*, 2023, pp. 8621–8627.

[25] S. Alsalehi, E. Aasi, R. Weiss, and C. Belta, "Learning spatio-temporal specifications for dynamical systems," in *Proc. of Machine Learning Research*, 2022, pp. 968–980.

[26] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[27] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: Theory, computation, and design*, 2nd ed. Nob Hill Publishing, 2017.

[28] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[29] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proc. of the Int. Conf. on Machine Learning*, 2007, pp. 745–750.

[30] N. Hansen, *The CMA Evolution Strategy: A Comparing Review*. Springer, 2006, pp. 75–102.

[31] M. Okada and T. Taniguchi, "Acceleration of gradient-based path integral method for efficient optimal and inverse optimal control," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2018, pp. 3013–3020.

[32] H. J. Kappen and H. C. Ruiz, "Adaptive importance sampling for control and inference," *Journal of Statistical Physics*, vol. 162, no. 5, pp. 1244–1266, 2016.

[33] R. Kusumoto, L. Palmieri, M. Spies, A. Csiszar, and K. O. Arras, "Informed information theoretic model predictive control," in *Proc. of the Int. Conf. on Robotics and Automation*, 2019, pp. 2047–2053.

[34] I. M. Balci, E. Bakolas, B. Vlahov, and E. A. Theodorou, "Constrained covariance steering based tube-MPPI," in *Proc. of the American Control Conf.*, 2022, pp. 4197–4202.

[35] H. Homburger, S. Wirtensohn, M. Diehl, and J. Reuter, "Feature-based MPPI control with applications to maritime systems," *Machines*, vol. 10, no. 10, pp. 1–23, 2022.

[36] E. Trevisan and J. Alonso-Mora, "Biased-MPPI: Informing sampling-based model predictive control by fusing ancillary controllers," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5871–5878, 2024.

[37] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.

[38] T. Lefebvre and G. Crevecoeur, "Path integral policy improvement with differential dynamic programming," in *Proc. of the IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2019, pp. 739–745.

[39] V. Gómez, S. Thijssen, A. Symington, S. Hailes, and H. Kappen, "Real-time stochastic optimal control for multi-agent quadrotor systems," in *Proc. of the Int. Conf. on Automated Planning and Scheduling*, 2016, pp. 468–476.

[40] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 1433–1440.

[41] H. Homburger, S. Wirtensohn, and J. Reuter, "Docking control of a fully-actuated autonomous vessel using model predictive path integral control," in *Proc. of the European Control Conf.*, 2022, pp. 755–760.

[42] L. Streichenberg, E. Trevisan, J. J. Chung, R. Siegwart, and J. Alonso-Mora, "Multi-agent path integral control for interaction-aware motion planning in urban canals," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2023, pp. 1379–1385.

[43] P. Nicolay, Y. Petillot, M. Marfeychuk, S. Wang, and I. Carlucho, "Enhancing AUV autonomy with model predictive path integral control," in *Proc. of the MTS/IEEE OCEANS Conf.*, 2023, pp. 1–10.

[44] L. Hou, H. Wang, H. Zou, and Y. Zhou, "Robotic manipulation planning for automatic peeling of glass substrate based on online learning model predictive path integral," *MDPI Sensors*, vol. 22, no. 3, pp. 1–20, 2022.

[45] K. Yamamoto, R. Ariizumi, T. Hayakawa, and F. Matsuno, "Path

integral policy improvement with population adaptation," *IEEE Transactions on Cybernetics*, vol. 52, no. 1, pp. 312–322, 2022.
[46] J. Carius, R. Ranftl, F. Farshidian, and M. Hutter, "Constrained stochastic optimal control with learned importance sampling: A path integral approach," *Int. Journal of Robotics Research*, vol. 41, no. 2, pp. 189–209, 2022.
[47] H. Homburger, S. Wirtensohn, and J. Reuter, "Swinging up and stabilization control of the furuta pendulum using model predictive path integral control," in *Proc. of the Mediterranean Conf. on Control and Automation*, 2022, pp. 7–12.
[48] B. Vlahov, J. Gibson, M. Gandhi, and E. A. Theodorou, "MPPI-Generic: A CUDA library for stochastic optimization," *arXiv preprint arXiv:2409.07563*, 2024.
[49] M. Kazim, J. Hong, M.-G. Kim, and K.-K. K. Kim, "Recent advances in path integral control for trajectory optimization: An overview in theoretical and algorithmic perspectives," *Annual Reviews in Control*, vol. 57, pp. 1–17, 2024.
[50] R. T. Rockafellar and R. J.-B. Wets, "Deterministic and stochastic optimization problems of bolza type in discrete time," *Stochastics*, vol. 10, no. 3-4, pp. 273–312, 1983.
[51] H. Homburger, F. Messerer, M. Diehl, and J. Reuter, "Optimality and suboptimality of MPPI control in stochastic and deterministic settings," *arXiv preprint arXiv:2502.20953*, 2025.
[52] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com
[53] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.
[54] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Sys. Letters*, vol. 5, no. 1, pp. 241–246, 2021.
[55] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.
[56] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.