Sampling-based Polytopic Trees for Approximate Optimal Control of Piecewise Affine Systems

Sadra Sadraddini and Russ Tedrake

Abstract— **Piecewise affine (PWA) systems are widely used to model highly nonlinear behaviors such as contact dynamics in robot locomotion and manipulation. Existing control techniques for PWA systems have computational drawbacks, both in offline design and online implementation. In this paper, we introduce a method to obtain feedback control policies and a corresponding set of admissible initial conditions for discrete-time PWA systems such that all the closed-loop trajectories reach a goal polytope, while a cost function is optimized. The idea is conceptually similar to LQR-trees [1], which consists of 3 steps: (1) open-loop trajectory optimization, (2) feedback control for computation of "funnels" of states around trajectories, and (3) repeating (1) and (2) in a way that the funnels are grown backward from the goal in a tree fashion and fill the state-space as much as possible. We show PWA dynamics can be exploited to combine step (1) and (2) into a single step that is tackled using mixed-integer convex programming, which makes the method suitable for dealing with hard constraints. Illustrative examples on contact-based dynamics are presented.**

I. Introduction

Many interesting behaviors in robotics are captured by highly nonlinear models. A prominent class is control through multiple contacts, where the robot must make and break contact with the environment in order to achieve its objectives. Examples include walking [2]–[4], running [5], dexterous manipulation [6], and push-recovery [7]. A popular approach to characterize contact-based dynamics is using piecewise affine (PWA) models. While accurate robot models are fully nonlinear, PWA models are reasonable approximations in the neighborhoods of nominal trajectories/points, just as linear approximations are, with the extra benefit of capturing contact dynamics [8], [9]. PWA models are also popular in traffic networks [10] and gene circuits [11].

Controlling PWA systems is difficult since the controller has to determine both the temporal order of modes and the inputs applied at each one. Completeness is important - not finding a solution when one exists is undesirable. Given an initial condition and a goal, the (optimal) trajectory that steers the state to the goal while respecting the dynamics and state/control constraints can be obtained using mixed-integer convex programming (MICP). Time is discretized to obtain a finite number of decision variables. Given a discrete-time PWA model and a fixed time horizon (steps required to get into the goal), MICP approaches are sound and complete - they find optimal solutions if they exist. However, they come at a large computational price. Their unreliable computation time, even for obtaining a feasible solution instead of the optimal one, hinders online implementation for robotic tasks with fast dynamics. While there is a great deal of research on improving the runtime of MICP solvers, they are still orders of magnitude too slow for most robot control problems.

An alternative is to move much of the computational burden to offline phase so the real-time implantation is a lookup table of simple control laws. However, existing approaches are not efficient even for relatively small systems. (Non-deterministic) finite-state abstractions [12] require state/control discretization, which scales poorly in high dimensions. Synthesizing PWA control laws corresponding to stabilizing piecewise quadratic (PWQ) Lyapunov functions was studied in [13], [14]. But the state-control partition producing the control laws was assumed to be the same as those of PWA dynamics, which is very restrictive and the method may fail while other solutions exist [14].

Multi-parametric programming [15] for model predictive control (MPC) results in explicit hybrid MPC schemes that also provide PWA control laws [16]. This approach is complete if the horizon is fixed. But it does not scale beyond very simple problems. A heuristic is to identify useful mode sequences in advance to fill the gap between explicit and full-blown online hybrid MPC [9], [17], [18]. However, these approaches still suffer from the computational complexity caused by the number of integers - for those initial conditions that require long horizons, computations become prohibitive.

Since the problem is of reachability class, it is amenable to anytime algorithms in sampling-based motion planning [19] such as rapidly exploring random trees (RRTs) and its variants [20]. RRTs for hybrid systems [21] provide little robustness understanding as the nodes in the tree correspond to points in the state-space. Therefore, there is no formal guarantee that a trajectory that deviates from the points is able to recover and achieve the goal. Moreover, PWA constraints pose a challenge for choosing an appropriate metric in exploring the state-space. The authors in [1] used sums-of-square (SOS) programming to obtain regions of attractions (funnels) for (time-varying) linear-quadratic regulators (LQR) that stabilize goal-reaching trajectories, which are obtained using nonlinear optimization in advance. These regions are grown backward from the goal in a tree fashion. This technique is called *LQR-trees* and was originally introduced for smooth continuous-time systems, and also was applied to limit cycle stabilization of hybrid systems in [22]. Transverse dynamics was studied to deal with switching surfaces [23]. Nonlinear optimization of trajectories and funnels for LQR-trees becomes complicated

The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar st, Cambridge, MA 02139, {sadra, russt}@mit.edu. This work was partially supported by ONR Award No.: N00014-17-1-2699.

for systems with many modes and state/input constraints. We desire an approach that exploits the properties of PWA systems and mitigates all the mentioned concerns.

In this paper, we propose a method that uses ideas both in hybrid MPC and funnels in LQR-trees. Our technique can be viewed as a discrete-time PWA version of LQR-trees. The main contributions of this paper are i) a framework for fusing trajectory optimization with the computation of polytopes of admissible states around them into a single MICP problem (Sec. III); ii) sampling-based approach (similar to RRT [19]) to grow a tree of polytopes backward from the goal such that the union of polytopes cover the state-space as much as possible (Sec. IV). Once the tree is computed, online implementation requires few matrix multiplications or small convex programs (Sec. V). We also obtain a cost-to-go function that over-approximates the optimal one. Similar to LQR-trees, we guarantee probabilistic feedback coverage: as the number of samples goes to infinity, the union of polytopes cover the whole region of admissible states with probability one. Three examples are presented in Sec. VI. The proof for lemmas and theorems are available in in the extended version of this paper [25].

II. Problem Formulation and Approach

Notation: The set of real, non-negative real, integer numbers, and empty set are denoted by $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{N}$, and $\emptyset$, respectively. Given $\mathbb{S} \subset \mathbb{R}^n$ and $A \subset \mathbb{R}^{n_A \times n}$, we interpret $A\mathbb{S}$ as $\{As | s \in \mathbb{S}\}$. Set additions are interpreted in Minkowski sense. The vector of all ones is denoted by $\underline{1}$, where the dimension is unambiguously interpretable from the context. A *polyhedron* $\mathbb{H} \subset \mathbb{R}^n$ has the form $\mathbb{H} = \{x \in \mathbb{R}^n | Hx \le h\}$, where $H \in \mathbb{R}^{n_H \times n}, h \in \mathbb{R}^{n_H}$. All inequality relations are interpreted element-wise. A bounded polyhedron is called a *polytope*.

We study discrete-time systems of the form

$$x_{t+1} = F(x_t, u_t), \tag{1}$$

where $x_t \in \mathbb{X}, \mathbb{X} \subset \mathbb{R}^n$, is the state at time t, $u_t \in \mathbb{U}, \mathbb{U} \subset \mathbb{R}^m$, is the control input at time t, $t \in \mathbb{N}$, and $F : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ is a PWA function given as:

$$F(x, u) = A_i x + B_i u + c_i, (x, u) \in \mathbb{H}_i, \tag{2}$$

where $n_\mathcal{M} \in \mathbb{N}$ is the number of modes, $\mathbb{H}_i, i \in \mathcal{M}, \mathcal{M} := \{1, \cdots, n_\mathcal{M}\}$, construct a polytopic partition of $\mathbb{X} \times \mathbb{U}$, and $A_i \in \mathbb{R}^{n \times n}, B_i \in \mathbb{R}^{n \times m}$ and $c_i \in \mathbb{R}^n$ are constant matrices, representing affine dynamics in mode i.

Problem 1: Given a PWA system (2) and a goal polytope $\mathbb{X}_{\text{Goal}} \subset \mathbb{X}$, find the largest set of initial conditions $\mathbb{X}_{\text{initial}} \subseteq \mathbb{X}$ and a control policy $\pi : \mathbb{X} \to \mathbb{U}$ such that all the points in $\mathbb{X}_{\text{initial}}$ are steered into $\mathbb{X}_{\text{Goal}}$ in finite time. Moreover, if multiple strategies are available, select the one that

$$\text{minimize } J := \sum_{t=0}^{T_f} \gamma_i(x_t, u_t), \tag{3}$$

where $\gamma_i : \mathbb{H}_i \to \mathbb{R}, i \in \mathcal{M}$, x_0 is the initial state, $(x_t, u_t) \in \mathbb{H}_i$, $u_t = \pi(x_t), t = 0, \cdots, T_f - 1, T_f \in \mathbb{N}$, and $x_{T_f} \in \mathbb{X}_{\text{Goal}}$. Our framework is able to accommodate a finite union of polytopes as the goal, but we stick to single polytope for brevity. The solution to Problem 1 consists of both the control policy π and the set of admissible initial conditions $\mathbb{X}_{\text{initial}}$. Finding representations for both is difficult. However, given $x \in \mathbb{X}$ and $T \in \mathbb{N}$, the following MICP problem:

$$\begin{array}{ll} \min & \sum_{t=0}^{T} \gamma_i(x_\tau, u_\tau) \\ \text{s.t} & x_{\tau+1} = F(x_\tau, u_\tau), (x_\tau, u_\tau) \in \mathbb{H}_i, \\ & \tau = 0, 1, \cdots, T-1, x_0 = x, x_T \in \mathbb{X}_{\text{goal}}, \end{array} \tag{4}$$

yields the optimal control sequence $u_0^*, \cdots, u_{T-1}^*$, which is an open-loop plan. We have $x \in \mathbb{X}_{\text{initial}}$ if and only if (4) is feasible for some $T \in \mathbb{N}$. Optimality is more subtle, as one has to check the solutions for all $T \in \mathbb{N}$ and pick the best. For some problems the optimal solutions have the smallest T - an obvious example is time-optimality where $\gamma_i = 1, \forall i \in \mathcal{M}$.

Solving (4) online is effectively a closed-loop policy, but as stated earlier, it is often too slow. We desire faster feedback laws. We develop an anytime algorithm that incrementally builds a set of initial conditions that asymptotically covers $\mathbb{X}_{\text{initial}}$, but sacrifices strong claims on optimality.

III. Polytopic Trajectories

In this section, we introduce the first part of our solution to Problem 1. We propose a method for obtaining trajectories of polytopes to a set of target polytopes. We focus on solving the following subproblem throughout this section.

Subproblem 1: Given a finite number of polytopic targets $\mathbb{X}_{i,\text{target}}, i = 1, \cdots, N$, and $T \in \mathbb{N}$, find a sequence of polytopes $\mathbb{X}_0, \mathbb{X}_1, \cdots, \mathbb{X}_T$, such that: i) (polytope-to-polytope flow) for all $x \in \mathbb{X}_\tau, 0 \le \tau < T$, there exists $u \in \mathbb{U}$ such that $F(x, u) \in \mathbb{X}_{\tau+1}$ and $(x, u) \in \mathbb{H}_i$ for some $i \in \mathcal{M}$; ii) (target constraint) $\exists i \in \{1, \cdots, N\}, \mathbb{X}_T \subseteq \mathbb{X}_{i,\text{target}}$.
A special case is when the target is a single polytope, but we formulate the general case of multiple polytopes as it turns to be useful in Sec. IV. Note that Subproblem 1 often does not have a unique solution. We will add cost criteria later in the paper mainly in order to obtain "large" polytopes.

A. Parameterization

Here is the main technical idea of this paper. We characterize polytopes by affine transformations of a pre-defined polytope $\mathbb{P} \subset \mathbb{R}^{n_p}$, where $\mathbb{P} := \{x \in \mathbb{R}^{n_p} | Px \le \underline{1}\}$. The user has to choose $\mathbb{P}$. For example, the unit cube with $n_p = n$ is a simple option; we highlight its advantages later in the paper. The polytope of possible states at time t is given by:

$$\mathbb{X}_t = \{\bar{x}_t\} \oplus G_t \mathbb{P}, \tag{5}$$

where $\bar{x}_t \in \mathbb{R}^n$ and $G_t \in \mathbb{R}^{n \times n_p}$ are parameters that we search over. Given $x \in \mathbb{X}_t$, one can compute $p(x) \in \mathbb{P}$ such that $x = \bar{x}_t + G_t p(x)$. Note that $p(x)$ may not be unique. A special case is when $n = n_p$ and G_t is an invertible matrix, so $p(x) = G^{-1}(x - \bar{x}_t)$. Otherwise, given $\bar{x}_t$ and G_t, $p(x)$ is determined from a linear program (with zero or some ad-hoc cost). We propose the following control law:

$$u_t(x) = \bar{u}_t + \theta_t p(x), \tag{6}$$

where $\bar{u}_t \in \mathbb{R}^m$ and $\theta_t \in \mathbb{R}^{m \times n_p}$ are parameters that, again, we search over. The set of all possible control inputs induced by (6) at time t is $\mathbb{U}_t := \{\bar{u}_t\} + \theta_t \mathbb{P}$.

B. Mixed-Integer Encoding

1) Trajectory: Let $i \in \mathcal{M}$ be such that $\mathbb{X}_t \times \mathbb{U}_t \subseteq \mathbb{H}_i$. In other words, we restrict that the product of each state/control polytope to lie in a single mode of the PWA system - the constraint ensuring this is discussed shortly. Using (6), we arrive at the following evolution for $\mathbb{X}_t$ if the mode is i:

$$\bar{x}_{t+1} = A_i \bar{x}_t + B_i \bar{u}_t + c_i, \tag{7a}$$

$$G_{t+1} = A_i G_t + B_i \theta_t. \tag{7b}$$

We encode (7) using *big-M method*, which is a standard procedure for translating PWA systems into mixed-integer constraints. We introduce binary variables $\delta_t^i \in \{0,1\}$, which are used in a way that δ_t^i takes 1 if mode at time t is i, and zero otherwise. Thus, we have the constraint:

$$\Sigma_{i \in \mathcal{M}} \delta_t^i = 1, t = 0, \cdots, T. \tag{8}$$

Eq. (7) is encoded as follows. For all $i \in \mathcal{M}$, we have:

$$-\mathrm{M}(1-\delta_t^i) \leq \bar{x}_{t+1} - A_i \bar{x}_t - B_i \bar{u}_t - c_i \leq \mathrm{M}(1-\delta_t^i), \tag{9a}$$

$$-\mathrm{M}(1-\delta_t^i) \leq G_{t+1} - A_i G_t - B_i \theta_t \leq \mathrm{M}(1-\delta_t^i), \tag{9b}$$

where M is a sufficiently large positive number. We omit detailed discussions on big-M encoding, as they are thoroughly studied in the literature, see, e.g., [24].

2) Subset Maintenance: In order to ensure that (7) is sound, we need to enforce that for some $i \in \mathcal{M}$, $\mathbb{X}_t \times \mathbb{U}_t \subseteq \mathbb{H}_i$. This becomes a set of mixed-integer linear constraints due to the following lemma.

Lemma 1: Given a polytope $\mathbb{Y} = \{y \in \mathbb{R}^m | H_y y \leq h_y\}$, $Q \in \mathbb{R}^{n \times m}$, $\bar{q} \in \mathbb{R}^m$, and polytopes $\mathbb{Z}_i = \{z \in \mathbb{R}^m | H_{z,i} z \leq h_{z,i}\}, i = 1, \cdots, N$, the condition $\exists i \in \{1, \cdots, N\}$ such that $Q\mathbb{Y} + \bar{q} \subseteq \mathbb{Z}_i$ is equivalent to

$$\begin{aligned} &\Lambda_i H_y = H_{z,i} Q_i, \Lambda_i h_y \leq h_{z,i}\delta_i - H_y \bar{q}_i, \Lambda_i \geq 0, \\ &\delta_i \in \{0,1\}, \sum_{i=1}^N \delta_i = 1, \sum_{i=1}^N \bar{q}_i = \bar{q}, \sum_{i=1}^N Q_i = Q. \end{aligned} \tag{10}$$

In case of $N = 1$, Lemma 1 is reduced to an extension of Farkas' Lemma in [26]. It also can be shown that if binary variables in (10) are relaxed to continuous variables in $[0,1]$, then (10) is equivalent to $Q\mathbb{Y} + \bar{q} \subseteq$ Convexhull$(\{\mathbb{Z}_i\}_{i=1,\cdots,N})$, which corresponds to the tightest binary relaxation. We use Lemma 1 to encode both the target constraints and also ensuring the fact that $\exists i \in \mathcal{M}$ such that $\mathbb{X}_t \times \mathbb{U}_t \subseteq \mathbb{H}_i$.

The solution to Subproblem 1 involves numerical values for $\zeta := \{\bar{x}_\tau, \bar{u}_\tau, G_\tau, \theta_\tau\}_{\tau=0,1,\cdots,T}$, which are obtained by solving the following optimization problem:

$$\begin{array}{lll} \zeta^* = & \arg\min & \alpha(\zeta) \\ & \text{s.t} & \mathbb{X}_T \subseteq \mathbb{X}_{i,\text{target}}, \\ & & (8),(9), \exists i \in \mathcal{M}, \mathbb{X}_i \times \mathbb{U}_i \subseteq \mathbb{H}_i, \end{array} \tag{11}$$

where α is a cost function with appropriate domain. Note that if a trajectory of points $(\bar{x}_0, \bar{u}_0), \cdots, (\bar{x}_T, \bar{u}_T)$, is feasible, a trivial solution is $G_\tau = 0, \theta_\tau = 0, \tau = 0, \cdots, T$, i.e., singletons instead of full-dimensional polytopes. We address this issue by introducing heuristics for α to obtain large, preferably full-dimensional, polytopes.

C. Volume Maximization

We desire polytopes $\mathbb{X}_0, \cdots, \mathbb{X}_T$ that are large in the sense they cover the state-space as much as possible. The ideal is maximizing the volume of the union of polytopes, which is a nonlinear objective. Note that polytopes may overlap. When $n_p = n$, a simple heuristic is to maximize the trace of G_0 - a linear objective. If $\forall p \in \mathbb{P}$, $-p \in \mathbb{P}$ (symmetric set), then multiplying any column of G_0 by -1 does not change $\mathbb{X}_0$. Therefore, we can safely assume that all the diagonal terms of G_0 are positive without any restriction posed on the space of solutions. A drawback of trace maximization is that it may still lead to zero volume. Additionally, it often leads to sparse solutions in a way that few diagonal terms take large values, but others become zero. We found that including the following heuristics is useful in (11) when $\mathbb{P} = [-1,1]^n$: First, we include a weighted combination of l_1 and l_∞ norms of the vector of diagonal terms of G_0 in $\alpha(\zeta)$. Second, by restricting G_0 to be an upper/lower triangular matrix and constraining all the diagonal terms to be greater than a small positive, non-zero volumes for $\mathbb{X}_0$ are guaranteed. Third, we include weighted summation of traces of $G_1, \cdots, G_T$ in $\alpha(\zeta)$. This promotes larger polytopes for $\mathbb{X}_1, \cdots, \mathbb{X}_T$. But there is no guarantee for this heuristic to prove useful as dynamical constraints dominate the relation between subsequent polytopes. Finally, if very small volume is reached, we reject the solution and resolve the optimization problem with a different objective. The mentioned heuristics make (11) a mixed-integer linear program (MILP) problem.

Remark 1: Maximizing the determinant of a square matrix subject to linear constraints can be cast as a semidefinite program (SDP) [27], which may prove useful for our application. While we leave investigation of this approach to future work, we note that it converts (11) to mixed-integer semidefinite programming (MISDP), for which solvers are still not as mature as MILP/MIQP solvers.

Complexity: The complexity of MICP solvers grow exponentially with number of integers involved, in general. The method introduced in this section introduces $Tn_\mathcal{M}$ binary variables, which is the same as (4). However, the price is introducing more continuous variables and constraints due to Lemma 1. The reward is obtaining a polytopic family of trajectories and a set of admissible initial conditions.

IV. Polytopic Trees

In this section, we provide the solution to Problem 1 by solving Subproblem 1 multiple times such that a tree is grown backward from the goal. The tree is formalized in Sec. IV-A. The technique is conceptually similar to the procedure in RRT and LQR-trees, which is detailed in Sec. IV-B.

A. Tree Structure

Definition 1: A directed, rooted, labeled tree (which we simply refer to as *tree* in the rest of the paper) is a tuple

$\mathcal{T} = (\mathcal{V}, v_0, \text{child}, \mathcal{C})$, where $\mathcal{V}$ is the set of nodes, $v_0 \in \mathcal{V}$ is the root node, $\text{child} : \mathcal{V} \setminus \{v_0\} \to \mathcal{V}$ is a function that maps each node to its unique successor (child) in the graph - the root does not have a child, and $\mathcal{C} : \mathcal{V} \to \mathbb{R}$ is a cost function that maps each node to a real value.
For all nodes $v \in \mathcal{V} \setminus \{v_0\}$, a unique path toward the root exists: $v, \text{child}(v), \text{child}(\text{child}(v)), \cdots, v_0$. The correspondence of the elements in Definition 1 with the solution to Problem 1 are as follows. Each node is a polytope in $\mathbb{X}$. We denote polytope corresponding to $v \in \mathcal{V}$ by $\mathbb{X}_v$. The root node is the goal polytope: $\mathbb{X}_{v_0} = \mathbb{X}_{\text{goal}}$. Note that for $\mathbb{X}_{\text{child}(v)}$, $v \in \mathcal{V} \setminus \{v_0\}$, we have already computed a control policy $\mu_v : \mathbb{X}_v \to \mathbb{U}$ such that $\{(x, \mu_v(x)) | x \in \mathbb{X}_v\} \subseteq \mathbb{H}_i$ for some $i \in \mathcal{M}$, and $\{A_i x + B_i \mu_v(x) + c_i | x \in \mathbb{X}_v\} \subseteq \mathbb{X}_{\text{child}(v)}$. We define $\mathcal{C}(\mathbb{X}_v)$ as the worst-case cost induced by moving from a point in $\mathbb{X}_v$ to $\mathbb{X}_{\text{child}(v)}$: $\mathcal{C}(v) = \max_{x \in \mathbb{X}_v} c_i(x, \mu(x))$. Using control law (6), it becomes:

$$\mathcal{C}(\mathbb{X}_v) = \max_{p \in \mathbb{P}} \gamma_i(\bar{x}_v + G_v p, \bar{u}_v + \theta_v p), \tag{12}$$

where $\mathbb{X}_v = \{\bar{x}_v\} + G_v \mathbb{P}$, $G_{\text{child}(v)} = A_i G_v + B_i \theta_v$, and $\bar{x}_{\text{child}(v)} = A_i \bar{x}_v + B_i \bar{u}_v + c_i$. We assume γ_i's are given such that (12) is tractable. Note that over-approximations of values in $\mathcal{C}$ are still valid for statements made in Sec. V. Once the tree is available, we recursively construct the cost-to-go (value) function $V : \mathcal{V} \to \mathbb{R}$ as:

$$V(v) = \mathcal{C}(v) + V(\text{child}(v)), \tag{13}$$

where $v \in \mathcal{V} \setminus \{v_0\}$, and $V(v_0) = 0$. As explained in Sec. V, the cost-to-go of nodes provides an upper-bound for the cost-to-go of states, and is a key component of the controller. The set of all states in the tree is given as $\mathbb{X}_{\text{tree}} = \bigcup_{v \in \mathcal{V}} \mathbb{X}_v$.

B. Growing the tree

Distance From a Polytope: First, we require a routine that computes the distance between $x \in \mathbb{X}$ and $\mathbb{X}_v = \{\bar{x}_v\} + G_v \mathbb{P}$ - it should be zero if and only if $x \in \mathbb{X}_v$. A natural candidate is the following optimization problem:

$$\begin{array}{lll} d(x, \mathbb{X}_v) = & \min & \|\delta\|_l \\ & \text{s.t} & x + \delta = \bar{x}_v + G_v p, p \in \mathbb{P}, \end{array} \tag{14}$$

where $l \in \{1, 2, \infty\}$ makes (14) a LP/QP. However, we desire closed-form expressions since we implement the distance function many times both in tree construction (offline) and controller implementation (online). By assuming $\mathbb{P} = [-1, 1]^n$, all full-dimensional polytopes become paralleltopes for which $x = \bar{x}_v + G_v p, p \in \mathbb{P}$, is equivalent to $\|G_v^{-1}(x - \bar{x}_v)\|_\infty \le 1$. Despite heuristics for obtaining full-dimensional polytopes, it is still possible that some polytopes have zero, or close to zero, volume. We circumvent this numerical issue by adding ϵ, a small number, to the singular values of G_v that are smaller than ϵ, to obtain G_v^ϵ. Define

$$\mathbb{X}_v^\epsilon = \{\bar{x}_v\} + G_v \mathbb{P}.$$

It is easy to verify that $\mathbb{X}_v^\epsilon \subseteq \mathbb{X}_v + \epsilon \mathbb{P}$. We Introduce

$$p_v(x) := {G_v^\epsilon}^{-1}(x - \bar{x}_v).$$

Notice that $x \in \mathbb{X}_v^\epsilon \Leftrightarrow p_v \in \mathbb{P}$. Let $p_v^*(x) := \min(1, \max(p_v(x), -1))$ - $\min$ and $\max$ operations are implemented row-wise - and

$$d_v(x) := \lim_{\epsilon \to 0} \|G_v^\epsilon (p_v(x) - p_v^*(x))\|_\infty, \tag{15}$$

which is basically the l_∞ distance between x and the point in $\mathbb{X}_v^\epsilon$ for which their l_∞ distance is minimal in the space transformed by ${G_v^\epsilon}^{-1}$ (which inherits metric properties as it is a one-to-one transformation). The main advantage of the unconventional distance (15) is that it tends to cancel out the effect of ϵ, and more importantly, it can be cast as elementary matrix operations (multiplications and row-wise $\min, \max$). Thus, it is efficiently implementable for a large number of polytopes in parallel by stacking (15) for all $v \in \mathcal{V}$. In case of other choices of $\mathbb{P}$, one may need to stick to (14), which may be too slow for some applications.

Sample and Rejection: We assume we are given a routine `sample` which randomly selects points from $\mathbb{X}$ with total support, and is repeated in an independent and identically distributed (i.i.d.) fashion. For sampling from polytopes, we used hit and run polytopic sampler in [28]. We select points from $\mathbb{X} \setminus \mathbb{X}_{\text{tree}}$ by rejecting samples in $\mathbb{X}_{\text{tree}}$. Note that $x \in \mathbb{X} \setminus \mathbb{X}_{\text{tree}}$ iff $\{d_v(x) = 0 | v \in \mathcal{V}\} = \emptyset$.

1) Tree initialization: We initially solve Subproblem 1 with no constraints for $\bar{x}_0$ and focus on obtaining large volumes for polytopes. There is a trade-off in choosing T. If T is small, then computations are faster, but we obtain fewer polytopes. Larger T leads to larger problem size, and often (but not necessarily) finds larger polytopes. Once a solution to Subproblem 1 is obtained, we initialize the tree by adding nodes $v_0, \cdots, v_{T-1}$ corresponding to $\mathbb{X}_0, \cdots, \mathbb{X}_{T-1}$, and set $\text{child}(v_{\tau+1}) = v_\tau, \tau = 0, \cdots, T-1$. The cost and value functions are constructed unambiguously.

2) Tree Growth: The procedure is outlined in Algorithm 1. A heuristic that is useful and is essential in ensuring probabilistic coverage in Theorem. 1 is $\bar{x}_0 = x_{\texttt{sample}} + \eta$ in line 5:, where $\eta_k \in \beta \eta_{\max k} \mathbb{P}$ (subscript k stands for k'th cartesian direction), and $\eta_{\max k}$ a positive number such that $\mathbb{X} \subseteq \prod_{k=1}^n \eta_{\max k}[-1, 1]$. This heuristic allows moving from the $x_{\texttt{sample}}$ to gain feasibility/larger polytopes. We randomly select $\beta \in [0, 1]$. Note that we lose the guarantee that $\mathbb{X}_0 \not\subseteq \mathbb{X}_{\text{tree}}$. It suffices to reject solutions if the centroid and (all or some of) vertices of $\mathbb{X}_0^*$ are in $\mathbb{X}_{\text{tree}}$, which are rapidly checked using (15). As $\mathbb{X}_{\text{tree}}$ gets larger, we may bias β toward smaller values to cut the corners. Moreover, as the MILP for line 2: gets larger, it is beneficial to split the target set into smaller polytypic clusters so we solve multiple smaller MILPs. Notice that our tree extension routine is also governed by MICP. Finally, we add nodes corresponding to the obtained polytopes to the tree $\mathcal{T}$. The child, cost and value functions are updated accordingly. The default criteria in Line 4: is $\text{Volume}(\mathbb{X}_0) > 0$.

Remark 2: There is no optimality consideration in Algorithm 1, hence it may lead to unnecessarily large cost-to-go values. Inspired by the rewiring procedure in RRT* [19], child function can be modified when lower cost-to-go values for some nodes according to (13) become available.

Algorithm 1 Random Trees of Polytopes

```
Require: Initialize $\mathcal{T}^1 = (\mathcal{V}, v_0, \text{child}, \mathcal{C})$
1: for step 2 to Number of iterations do
2:     Select $x_{\texttt{sample}} \in \mathbb{X} \setminus \mathbb{X}_{\text{tree}}^{K-1}$, and $1 \le T \le T_{\max}$
3:     Solve Subproblem 1 with $\bar{x}_0 = x_{\texttt{sample}} + \eta$ and $\mathbb{X}_{\text{target}} = \mathbb{X}_{\text{tree}}$
4:     if feasible solution exists and meets criteria then
5:         Add $v_0^*, \cdots, v_{T-1}^*$, corresponding to $\mathbb{X}_0, \cdots, \mathbb{X}_T$, to $\mathcal{V}$, $\text{child}(v_\tau^*) = v_{\tau+1}^*, \tau = 0, \cdots, T-1$, $\text{child}(v_{T-1}^*) = v, \mathbb{X}_T \subseteq \mathbb{X}_v$.
6:         Update cost and value functions of $\mathcal{T}^K$.
```

Note that unlike RRT*, we do not seek asymptotic optimality guarantees since we do not resample from $\mathbb{X}_{tree}$.

Theorem 1: Let $\mathbb{X}_{\text{tree}}^K$ be the set of states covered by the tree in the K'th iteration in Algorithm 1. Then the following property holds with probability 1:

$$\lim_{K \to \infty} (\mathbb{X}_{\text{initial}} \setminus \mathbb{X}_{\text{tree}}^K) = \emptyset. \tag{16}$$

V. Control Synthesis

Once we obtain the tree, synthesizing the controller for Problem 1 is straightforward. In this section, we consider both the cases when the state is in $\mathbb{X}_{\text{tree}}$ or outside of $\mathbb{X}_{\text{tree}}$ - for the latter we do not have formal guarantees that we can steer the state to $\mathbb{X}_{\text{goal}}$, but we provide heuristics.

1) $x \in \mathbb{X}_{\text{tree}}$: The following theorem provides the control policy. The proof follows from Sec. III and the tree structure.

Theorem 2: Let $\mu_{\mathbb{X}_{\text{tree}}} : \mathbb{X}_{\text{tree}} \to \mathbb{U}$ be

$$\mu_{\mathbb{X}_{\text{tree}}}(x) = \bar{u}_{v^*} + \theta_{v^*} p_{v^*}(x) \tag{17}$$

where $v^* = \arg\min_{v \in V} \{V(v) | x \in \mathbb{X}_v\}$. Then, given $x_0 \in \mathbb{X}_{\text{tree}}$, implementing μ_{tree} yields a trajectory $x_0, \cdots, x_T$, such that $x_\tau \in \mathbb{X}_{\text{tree}}, \tau = 0, \cdots, T$, $x_T \in \mathbb{X}_{\text{goal}}$, and the total cost J in (3) is upper bounded by $V(v^*)$.

In words, the control policy (17) finds the set of polytopes which the current state belongs to, selects the one with the least cost-to-go, and implements the control law to get to its child polytope. All operations required for implementing (17) are basic matrix operations - orders of magnitude faster than solving MICPs. The polytope search routine can be further accelerated using binary search trees for online implementation of PWA control laws [29].

2) $x \in \mathbb{X} \setminus \mathbb{X}_{tree}$: In case $x \notin \mathbb{X}_{tree}$, we still want to feed the system with some control input - no matter if the state can be steered toward $\mathbb{X}_{\text{goal}}$ or not. We propose the following heuristic: use (15) to find the closest polytope $\mathbb{X}_{v^*}$, and apply control input that steers x toward $\mathbb{X}_{\text{child}(v^*)}$ as much as possible by solving the following convex program:

$$\begin{aligned} \mu_{\mathbb{X} \setminus \mathbb{X}_{\text{tree}}}(x) = \quad & \arg\min \quad \|\delta\|_l \\ & \text{s.t} \quad F(x,u) + \delta \in \mathbb{X}_{\text{child}(v^*)}, \end{aligned} \tag{18}$$

where $l \in \{1, 2, \infty\}$. Note that if optimal δ is zero, then we have succeeded in getting x back into the tree. We do not have any formal guarantees for the policy driven by (18), but we have examples of its successful implementations (see Sec. VI), even when x is quite far from $\mathbb{X}_{tree}$. More elaborate formulations of (18) are possible in the price of higher computational cost. One can consider multiple polytopes or steps to search a wider range of getting-to-tree possibilities. Note the contrast here with full-blown online hybrid MPC as regions/policies are at least partially computed in advance.

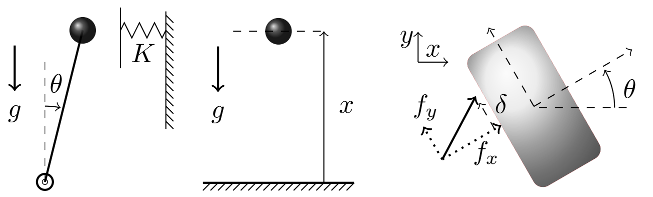

Fig. 1. Examples: **Left:** Inverted Pendulum with Wall **Middle**: Bouncing Ball **Right:** Planar Pushing.

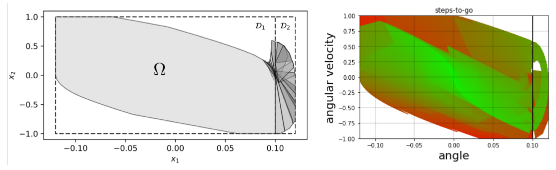

Fig. 2. Example 1: The results in **Left:** [9], **Right:** this paper.

VI. Examples

Software. Python scripts are publicly available in [25]. Instructions are included to guide the user to define its own problem and use our method, or reproduce the results here. Examples (including three shown in this paper) are included. The high-level details of the examples are explained here and one may refer to [25] for full details. The cost criteria in all examples is time ($c_i = 1, i \in \mathcal{M}$).

Example 1 (Inverted Pendulum with Wall): We adopt 2D example 1 from [9]. Consider an inverted pendulum hitting a vertical wall at $\theta = 0.1$ as in Fig. 1 [Left]. The contact model is characterized by linear spring $K = 1000$. Time is discretized by 0.01. The control input $u \in [-0.4, 0.4]g$ corresponds to the torque applied to the pendulum. The goal is to steer the state $(\theta, \dot{\theta})$ to the origin - a singleton. The authors in [9] precomputed a set of admissible initial states in contact-free mode, denoted by Ω, which is a set of states that can be driven into the origin using a linear feedback law. Explicit hybrid MPC was used to steer other states into Ω. Here, we deliberately ignore exploiting the fact that Ω can be easily pre-computed, and entirely rely on our method to find $\mathbb{X}_{\text{initial}}$. Ideally, our method should recover Ω. The final tree after 60 iterations is shown in Fig. 2. In comparison to the result in [9], illustrated in Fig. 2 [Left], not only we recover most of Ω (and a bit beyond, as richer PWA laws are considered), but we also find a fairly large set of initial conditions in the top right of the state-space that the method in [9] did not find. The reason is that the MPC horizon was limited to 10 in [9], whereas our method does not directly suffer from short horizons. The green-red color spectrum in Fig. 2 [Right] correspond to cost-to-go values.

Example 2 (Bouncing Ball): We consider the vertical mo-

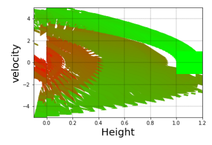
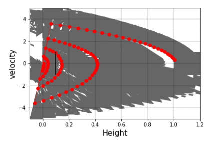
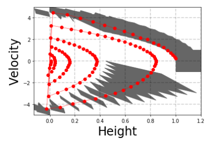

Fig. 3. Example 2: **Left:** Cost-to-go. **Middle:** $\mu_{\mathbb{X}_{\text{tree}}}$. **Right:** $\mu_{\mathbb{X}\setminus\mathbb{X}_{\text{tree}}}$.

tion of a ball falling under the gravity and ground impacts (see Fig. 1 [Middle]). We have $\ddot{x} = -g + u$ with velocity sign change after hitting the ground, where $g = -9.8m/s^2$ and u is the control input. We use $\Delta t = 0.02s$ for time-discretization. We set $\mathbb{U} = \{u| \|u\| \le 3m/s^2\}$, which indicates that the control actuation is not powerful enough to keep the ball from accelerating downward. The goal is to design a feedback strategy and find the set of admissible initial conditions such that the ball height and velocity reach $[1, 1.2]m$ and $[-0.5, 0.5]m/s$, respectively, while always satisfying the hard constraint that the velocity is within $[-5, 5]m/s^2$. This problem is quite challenging because a lot of contacts may be necessary. The authors in [30] also consider at a similar problem, but the method is not correct-by-design as it involves heuristics. Similarities exist between this example and swinging up an inverted pendulum in [1]. Both have nonlinear nature, and applying maximal control input in the direction of velocity increases energy.

Using $T_{\max} = 20$, Algorithm 1 "discovers" at the third iteration that by using impacts, more states can be driven into the goal. By increasing the number of iterations, nearly all states close to the origin are covered as the number of bounces are increased. Note that finding polytopes becomes more difficult as the space shrinks and polytopes close to the switching surface become small. The final tree after 100 iterations and 1049 polytopes is shown in Fig. 3. A clear disadvantage of the method in this paper versus LQR-trees in [1] is caused by the discrete-time nature of the problem that leads to larger number of iterations required to fill the state-space. If the empty space between connecting polytopes is filled in a continuous-time sense, larger trees can be formed more quickly. We leave formal investigation of this issues to our future work. We randomly selected 500 points in $\mathbb{X}$ and found 359 of them are in $\mathbb{X}_{\text{tree}}^{100}$. By checking feasibility of (4) for different values of T, we found 416 points are drivable to the goal in less than 80 steps - the tree has covered nearly 86 percent of them. However, when (4) is performed with smaller horizons, the tree has much more coverage. For instance, for $T \le 20(30)$, only 106(311) of 359 points in $\mathbb{X}_{\text{tree}}$ lead to feasible (4). Two sample trajectories using control policies in (17) and (18) (implemented on a much lesser grown tree) are shown - note the success of the latter.

Example 3 (Planar Pushing): The problem and the model is adopted from [18]. The 6D state consists of (x, y, θ) for the box, and (δ, f_x, f_y) for the contact point and forces, as illustrated in Fig. 1 [Right] - we have augmented the state with controls to construct the PWA dynamics and cells in the state-space. When the contact exists, there are three modes based on whether the contact point is fixed (pusher sticked), slides up, or slides down. The

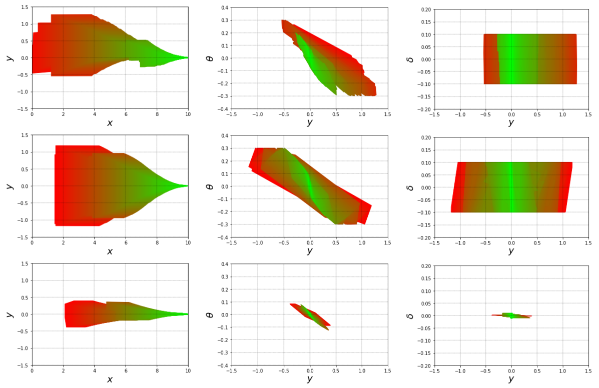

Fig. 4. Example 3: **Top Row:** Tree projections for $T_{\max}$ = after 473 iterations (1159 polytopes). **Middle Row:** Tree Projections for $T_{\max} = 30$ after 16 iterations (251 polytopes). **Bottom Row:** Tree projections for constrained linear model (only sticking) after 66 iterations (363 polytopes).

Coulomb friction coefficient between the pusher and the box is set to $\mu = 0.02$, which means it is very hard to stick the pusher to the box. The model is nonlinear, including bilinear terms for exerted wrench. We consider hybrid stabilization of a nominal trajectory and use the local PWA dynamics to compute the tree around the nominal trajectory, which is the horizontal line with $y = \theta = \delta = f_y = 0, f_x = 1$. We set $0.8 \le |f_x| \le 1.2, |\theta| \le 0.3, |\delta| \le 0.1$, to approximately linearize the bilinear terms. The goal is a box of size 0.01 around $x = 10$ and other variables set to zero. The results after 473 iterations and obtaining 1159 polytopes are illustrated in shown Fig. 4. We have projected the tree on $x - y$, $y - \theta$, and $y - \delta$ planes. The asymmetry is due to the sampling nature of our solution - the tree was grown toward greater y's and we expect more iterations are required to grow the tree in the other direction. A sample trajectory from $x = 0, y = 1.5, \theta = 0.1, \delta = 0, f_x = 1, f_y = 0$, deliberately selected from outside of the tree, is shown. It is observed that the controller manages to bring the state into the tree and remains thereafter until reaching the goal at $t = 227$. As a basic numerical comparison, we found solving the hybrid MPC with horizon $T = 100 - 200$ rates about 0.5-1.2Hz, whereas a rudimentary setup of our controller is much faster at 20-50Hz. Our solution compares to [18], which uses machine learning to "learn" the mode sequences. In contrast, our synthesis is formal and the solution is correct-by-design - at least as long as $x_0 \in \mathbb{X}_{\text{tree}}$. Finally, we expect a linear controller to perform poorly on this problem as it can not take into account mode switches. A tree for the system constrained to remain in the sticking mode was computed and is shown in Fig. 4 for comparison.

VII. Discussion and Future work

We believe our method is a step toward formal design of fast hybrid feedback policies for multi-contact tasks such as robotic manipulation. With biased sampling, which RRT methods are shown to greatly benefit from in high dimensions [31], [32], around pre-computed nominal trajectories, trees can be grown locally and connected to each other in an efficient way for complex manipulation problems.

References

[1] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[2] S. H. Collins and A. Ruina, "A bipedal walking robot with efficient and human-like gait," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 1983–1988.

[3] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, "Models, feedback control, and open problems of 3d bipedal robotic walking," *Automatica*, vol. 50, no. 8, pp. 1955–1988, 2014.

[4] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 279–286.

[5] J. W. Grizzle, J. Hurst, B. Morris, H.-W. Park, and K. Sreenath, "Mabel, a new robotic bipedal walker and runner," in *American Control Conference, 2009. ACC'09*. IEEE, 2009, pp. 2030–2036.

[6] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 255–262.

[7] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 200–207.

[8] A. K. Valenzuela, "Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[9] T. Marcucci, R. Deits, M. Gabiccini, A. Biechi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*. IEEE, 2017, pp. 31–38.

[10] N. Mehr, D. Sadigh, R. Horowitz, S. S. Sastry, and S. A. Seshia, "Stochastic predictive freeway ramp metering from signal temporal logic specifications," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 4884–4889.

[11] H. De Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann, "Qualitative simulation of genetic regulatory networks using piecewise-linear models," *Bulletin of mathematical biology*, vol. 66, no. 2, pp. 301–340, 2004.

[12] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta, "Temporal Logic Control of Discrete-Time Piecewise Affine Systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1491–1504, 2012.

[13] A. Hassibi and S. Boyd, "Quadratic stabilization and control of piecewise-linear systems," in *American Control Conference, 1998. Proceedings of the 1998*, vol. 6. IEEE, 1998, pp. 3659–3664.

[14] W. Han and R. Tedrake, "Feedback design for multi-contact push recovery via lmi approximation of the piecewise-affine quadratic regulator," in *n Proceedings of the 2017 IEEE-RAS International Conference on Humanoid Robots, 2017*. IEEE-RAS, 2017.

[15] V. Dua and E. N. Pistikopoulos, "An algorithm for the solution of multiparametric mixed integer linear programming problems," *Annals of operations research*, vol. 99, no. 1-4, pp. 123–139, 2000.

[16] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *American Control Conference, 2000. Proceedings of the 2000*, vol. 2. IEEE, 2000, pp. 1190–1194.

[17] F. R. Hogan and A. Rodriguez, "Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics," *arXiv preprint arXiv:1611.08268*, 2016.

[18] F. R. Hogan, E. R. Grau, and A. Rodriguez, "Reactive planar manipulation with convex hybrid mpc," *arXiv preprint arXiv:1710.05724*, 2017.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[20] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[21] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems," *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 5, pp. 575–590, 2006.

[22] S. Rajasekaran, R. Natarajan, and J. D. Taylor, "Towards planning and control of hybrid systems with limit cycle using lqr trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 5196–5203.

[23] I. R. Manchester, M. M. Tobenkin, M. Levashov, and R. Tedrake, "Regions of attraction for hybrid limit cycles of walking robots," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5801–5806, 2011.

[24] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[25] `https://github.com/sadraddini/pwa-control`.

[26] S. Raković, E. C. Kerrigan, D. Q. Mayne, and K. I. Kouramas, "Optimized robust control invariance for linear discrete-time systems: Theoretical foundations," *Automatica*, vol. 43, no. 5, pp. 831–841, 2007.

[27] L. Vandenberghe, S. Boyd, and S.-P. Wu, "Determinant maximization with linear matrix inequality constraints," *SIAM journal on matrix analysis and applications*, vol. 19, no. 2, pp. 499–533, 1998.

[28] H. O. Mete and Z. B. Zabinsky, "Pattern hit-and-run for sampling efficiently on polytopes," *Operations Research Letters*, vol. 40, no. 1, pp. 6–11, 2012.

[29] P. Tøndel, T. A. Johansen, and A. Bemporad, "Evaluation of piecewise affine control via binary search tree," *Automatica*, vol. 39, no. 5, pp. 945–950, 2003.

[30] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems." *IEEE Trans. Robotics*, vol. 32, no. 5, pp. 1196–1214, 2016.

[31] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[32] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.