Efficient Trajectory Generation for Robotic Systems Constrained by Contact Force via Nonlinear Programming

Jaemin Lee [a], Efstathios Bakolas [b], Luis Sentis [b]

[a] *Department of Mechanical Engineering, The University of Texas at Austin, TX, 78712-1221, USA*

[b] *Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, TX 78712-1221, USA*

Abstract

In this work, we propose a trajectory generation method for robotic systems with contact force constraint based on optimal control and reachability analysis. Normally, the dynamics and constraints of the contact-constrained robot are nonlinear and coupled to each other. Instead of linearizing the model and constraints, we directly solve the optimal control problem to obtain the feasible state trajectory and the control input of the system. A tractable optimal control problem is formulated which is addressed by dual approaches, which are sampling-based dynamic programming and rigorous reachability analysis. The sampling-based method and Partially Observable Markov Decision Process (POMDP) are used to break down the end-to-end trajectory generation problem via sample-wise optimization in terms of given conditions. The result generates sequential pairs of subregions to be passed to reach the final goal. The reachability analysis ensures that we will find at least one trajectory starting from a given initial state and going through a sequence of subregions. The distinctive contributions of our method are to enable handling the intricate contact constraint coupled with system's dynamics due to the reduction of computational complexity of the algorithm. We validate our method using extensive numerical simulations with a legged robot.

Key words: Robots; Reachability; Contact Force.

1 Introduction

This paper considers the optimal control of robotic systems with contact force constraints. Often, it is required that legged or humanoid robots maintain stable foot or body contacts while executing given tasks. In such cases, contact forces constrain and determine the robot's state reachability together with other state and input constraints. Therefore, we seek to devise control algorithms that can generate trajectories for contact-constrained robots via formal state reachability analysis. Often, control studies for robotics assume that task trajectories are predefined [20, 34, 37], then attempt to find an instantaneously optimal solution to accomplish them. However the desired trajectories are frequently infeasible and it is not straight-forward to check the feasibility of trajectories under contact constraints a priori. Many motion planning and trajectory generation approaches for humanoid robots use very simple models of a robot such as considering center of mass dynamics under contact constraints [18, 28, 38]. However, those methods result on lower performance of the robots since they cannot capture the robot's kinematics or input constraints among other limitations.

Optimal control is an alternative approach to solve the trajectory generation problem for contact-constrained robots. Recently, trajectory generation for the legged robots was formulated as a bi-level optimization problem and solved by an iterative Linear Quadratic Regulator (iLQR) [8]. However, the iLQR has still challenging issues to address generic nonlinear constraints without constraint softening. This could result on motion planers that violate important physical constraints of robots. To strictly consider the constraints and nonlinearity of the robot models, we should directly solve the optimal control problem via Nonlinear Programming (NLP) in the process of obtaining feasible trajectories. Although many NLP solvers, i.e. SNOPT [12] and IPOPT [40], are available, NLP has significantly challenging issues. One is high computational cost for obtaining the solution. Also, feasible initial conditions are necessary for NLP. In our work, we propose two complementary pro-

[⋆] Some part of this paper will be presented at American Control Conference 2019 [25]. Corresponding author L. Sentis.

Email addresses: jmlee87@utexas.edu (Jaemin Lee), bakolas@austin.utexas.edu (Efstathios Bakolas), lsentis@austin.utexas.edu (Luis Sentis).

cesses to resolve these problems: reachability analysis and sampling-based dynamic programming.

The reachability problem consists of checking whether the state of the system can reach a specific state over a finite time horizon, starting from a given initial state. The field of robotics has often focused on configuration space reachability guided by the given tasks such as selecting the stance location of humanoid robots [7,41]. Although these methods are useful for kinematic feasibility, they are limited to address the requirements of dynamical systems considering various constraints such as joint velocity/torque limits and contact force constraints. To address this gap, we will employ optimal control on the nonlinear dynamical system with constraints.

In optimal control, the reachability analysis has been often used for nonlinear systems [2–4, 35, 36], hybrid dynamical systems [15, 30–32], and stochastic systems [1, 27, 39]. We can categorize established reachability analysis methods for nonlinear systems into three groups: 1) solving Hamilton-Jacobi-Bellman PDE, 2) using linearization and mathematical approximation 3) using propagation and mappings of a set of reachable states. First, for low dimensional dynamical systems, the reachability analysis is often achieved via Hamilton-Jacobi-Bellman PDE [6, 19]. For some systems, it is impossible to perform the reachability analysis by solving Hamilton-Jacobin-Bellman PDE. Instead, many approaches have been proposed to compute reachable sets exploiting mathematical techniques, optimization, inherent characteristics of systems, etc.

Other than the methods using Hamilton-Jacobi-Bellman PDE, many methodologies have been proposed to obtain reachable sets of systems. The logarithmic norm of a type of system's Jacobian is utilized to obtain over-approximated reachable sets for nonlinear continuous-time systems [29] and that norm is utilized for simulation-based reachability analysis [5]. Another approach tries to do more accurate reachability analysis for uncertain nonlinear systems by using more conservative approximations [4, 35]. Also, for continuous-time piecewise affine systems, linear matrix inequalities (LMI) are employed to characterize the bounds of reachable regions [16]. Another class of reachability analysis uses convex sets for approximation such as ellipsoid [22, 23], polytopes, zonotopes [13], and support functions [14, 24]. Although those approximation-based approaches are capable of extending to nonlinear systems, they only consider convex sets and often ignore other constraints. These are challenges for extending the previous approaches to more sophisticated and complicated systems. Additionally, computational complexity exponentially increases with respect to the dimension of the state space and the time length for those methods.

Our problem considers a constrained nonlinear system with a constrained variable, e.g. a contact force, coupled with system dynamics. Since the contact force is time varying and our problem is also high dimensional, it is very difficult to do reachability analysis of our system via Hamilton-Jacobi-Bellman PDE. Linearization of dynamics and approximating reachable sets with convex sets is not applicable to our problem because reachable sets of constrained nonlinear systems may not be convex. Moreover, we want to be strictly compared to methods softening or linearizing constraints like iLQR. Thus, we devise a new method consisting of propagating system states and approximating the reachable set. NLP using optimal control utilizes the approximated reachable set. In order to address the increasing computational complexity, we propose various techniques which are described thereafter. And, we do so, in the context of robotic systems with contact force constraints, in a computational efficient way.

Concretely, we incorporate a sampling-based approach, quadratic programming (QP), NLP, and approximation techniques such as propagation of boundary samples to solve our problem. More specifically, for dividing the end-to-end trajectory generation problem, we obtain the constrained-state set using a sampling-based approach and QP, then, reformulate the problem as a Partially Observable Markov Decision Process (POMDP) in the system's output space. An optimal Markov policy resulting from the dynamic programming (DP) provides a sequence of output subregions. The sequence of output subregions guides the path of output with avoiding unsafe output regions such as locations of obstacles in the output space. In the next step, we implement a rigorous reachability analysis between given pairs of subregions by propagating the states from the given initial state. In addition, we propose a method to approximate the reachable set using propagation of boundary states. The algorithmic efficiency of our method is one of the contributions of our work.

This paper is organized as follows. Section 2 defines our problem and the target class of system. A sampling-based algorithm for obtaining the set of constrained states in Section 3 and a POMDP for obtaining an optimal Markov policy are described in Section 4. In Section 5, we propose an approach to obtaining the reachable set and analyzing the method in detail. Based on the result of rigorous reachability analysis in Section 5, an optimal controller is designed by implementing the NLP of Section 6. The proposed approach is validated by simulation of a robotic legged system with contact force constraint in Section 6.

2 Problem Formulation

2.1 Notation

We denote the set of real n-dimensional vectors and the set of real $n \times m$ matrices by $\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$, respectively.

The sets of non-negative and non-positive real numbers are represented as $\mathbb{R}_{\leq 0}$ and $\mathbb{R}_{\geq 0}$, respectively. The set of natural numbers and the set of integer numbers are denoted by $\mathbb{N}$ and $\mathbb{Z}$, respectively. The set of positive definite $n \times n$ matrices and the set of positive semi-definite $n \times n$ matrices are denoted by $\mathbb{S}^n_{>0}$ and $\mathbb{S}^n_{\geq 0}$. When considering $z_1, z_2 \in \mathbb{N}$ with $z_2 > z_1$, the discrete interval between z_1 and z_2 is defined as $[z_1, z_2]_{\mathbb{N}} := \{z_1, z_1 + 1, \dots, z_2 - 1, z_2\}$. In case of real numbers $z_1, z_2 \in \mathbb{R}_{\geq 0}$, $[z_1, z_2]^{\Delta}_d := \{z_1, z_1+\Delta, \dots, z_2-\Delta, z_2\}$ denotes a discrete interval with Δ being the increment. When n real numbers $a_1, \dots, a_n$ are consider, $\mathrm{Vec}[a_i]^n_{i=1} \in \mathbb{R}^n$ represents a vector whose i-th element is a_i. Given $n \times m$ real numbers $a_{11}, \dots, a_{mn}$, a matrix whose (i,j) element is a_{ij} is denoted by $\mathrm{Mat}[a_{ij}]^{n,m}_{i,j=1} \in \mathbb{R}^{n \times m}$. Given a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathrm{tr}(\mathbf{A})$ denotes its trace. $\overline{\sigma}(\mathbf{A})$ and $\underline{\sigma}(\mathbf{A})$ represent the largest and smallest singular values of $\mathbf{A}$, respectively. Given matrices $\mathbf{A}_i \in \mathbb{R}^{n_i \times m}$ $i \in [1, z]_{\mathbb{N}}$, $\mathrm{Vertcat}(\mathbf{A}_1, \dots, \mathbf{A}_z) \in \mathbb{R}^{(n_q + \cdots + n_z) \times m}$ indicates a block matrix constructed by vertically concatenating the matrices $\mathbf{A}_i$ $i \in [1, z]_{\mathbb{N}}$. Given a set of real vectors $\mathcal{A} \subseteq \mathbb{R}^n$, $\mathrm{card}(\mathcal{A})$ denotes its cardinality. When considering particular cases such that $\mathcal{A} \subset \mathbb{R}^n$ with $n \in [1, 3]_{\mathbb{N}}$, $\mathrm{ghull}(\mathcal{A})$ and $\mathrm{gbd}(\mathcal{A})$ represent the general hull and the set of vectors closest the boundary of $\mathcal{A}$. $\mathbb{E}[.]$ represents the probabilistic expectation operator.

2.2 Nonlinear System Model

We characterize the equation of motion for general robotic systems with contact forces and assuming rigid body linkages as follows:

$$\mathbf{M}(q)\ddot{q} + g(\dot{q}, q) = \mathbf{S}^\top u + \mathbf{J}_c^\top(q) F_c \tag{1}$$

where $q \in \mathbb{R}^{n_q}$, $\mathbf{M}(q) \in \mathbb{S}^{n_q}_{>0}$, $g(\dot{q}, q) \in \mathbb{R}^{n_q}$, $\mathbf{S} \in \mathbb{R}^{n_q \times n_u}$, $u \in \mathbb{R}^{n_u}$, $\mathbf{J}_c(q) \in \mathbb{R}^{n_c \times n_q}$, and $F_c \in \mathbb{R}^{n_c}$ denote the joint variable, sum of Coriolis/centrifugal and gravitational forces, selection matrix for the actuation, input actuating joint torques, contact Jacobian matrix, and contact force, respectively. We can bring the differential equation (1) into a state space form by defining the state $x := [x_1^\top x_2^\top]^\top \in \mathbb{R}^{n_x}$ where $x_1 = q$ and $x_2 = \dot{q}$:

$$\begin{aligned} \dot{x}(t) &= f_{\mathfrak{C}}(x(t), u(t), F_c(t)) && \text{(2a)}\\ &= f_x(x(t)) + f_u(x(t))u(t) + f_c(x(t))F_c(t) && \text{(2b)}\\ y(t) &= f_y(x(t)) && \text{(2c)} \end{aligned}$$

$$f_x(x) := \begin{bmatrix} x_2(t) \\ -\mathbf{M}^{-1}(x_1) g(x_1, x_2) \end{bmatrix}$$

$$f_u(x) := \begin{bmatrix} \mathbf{0}_{n_q \times n_u} \\ \mathbf{M}^{-1}(x_1)\mathbf{S}^\top \end{bmatrix}, f_c(x) := \begin{bmatrix} \mathbf{0}_{n_q \times n_c} \\ \mathbf{M}^{-1}(x_1)\mathbf{J}_c^\top(x_1) \end{bmatrix}$$

where $f_x : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x}$, $f_u : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_u}$, and $f_c : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_c}$ are nonlinear functions of the state x. $f_y : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ is a nonlinear output function that represents the desired tasks, i.e. forward kinematics of points of robot that we wish to control. We define a discrete time state space model of the contact-constrained robotic system (DTSCR) as the discrete counterpart of the state space model:

$$\begin{aligned} x(t_{k+1}) &= f_{\mathfrak{D}}\left(x(t_k), u(t_k), F_c(t_k)\right) \\ &= x(t_k) + \sum_{i=1}^{\infty} \frac{B_i(x(t_k), u(t_k), F_c(t_k))}{i!} (\Delta t)^i \end{aligned} \tag{3}$$

where Δt denotes the sampling period for the discretization of the model. $f_{\mathfrak{D}} : \mathbb{R}^{n_x + n_u + n_c} \mapsto \mathbb{R}^{n_x}$ and $B_i(x, u, F_c) = \frac{\partial B_{i-1}(x,u,F_c)}{\partial x} B_1(x, u, F_c)$ and $B_1(x, u, F_c) = f_x(x) + f_u(x)u + f_c(x)F_c$.

2.3 Constraints of the System

We refer to h_e and h_i as the equality constraint function and the inequality constraint function, respectively, where we assume that $h_e(x) = 0$ and $h_i(x) \leq 0$. Three types of constraint functions are considered in this paper, i.e. functions describing state, the input, and the mixed state-input constraints denoted as $h_x(x)$, $h_u(u)$, and $h_{xu}(x, u)$, respectively. In practice, the state constraint is introduced to avoid violating joint position or velocity limits when controlling the robot. Input constraints are considered for describing the joint torque limit or the underactuation of floating robots. The mixed state-input constraint contains physically more intricate conditions such as mechanical power. Since the DTSCR is required to maintain stable contact while being controlled, we consider an additional constraint that is known as the contact wrench cone constrained to prevent slip and flip on contact surface. In particular, it is required that

$$h_{xc}(x, F_c) \leq 0, \quad h_{xc}(x, F_c) := \mathbf{W}_c(x) F_c \tag{4}$$

where $\mathbf{W}_c(x)$ is a matrix describing the unilateral constraint using a polyhedral approximation of the friction cone of a surface [9]. The position at the contact point should be constant, which is identical to having null velocity on the robot's contact with respect to the surface contact. The zero velocity constraint corresponds to the state constraint. We will aim at controlling robots represented by the DTSCR model for desired output goals given the aforementioned constraints.

2.4 Overall Scheme

The proposed approach consists of three methods, which are sampling-based optimization, solving a POMDP, and reachability-based optimal control. The overall procedure is depicted in Fig. 1. The first part of the

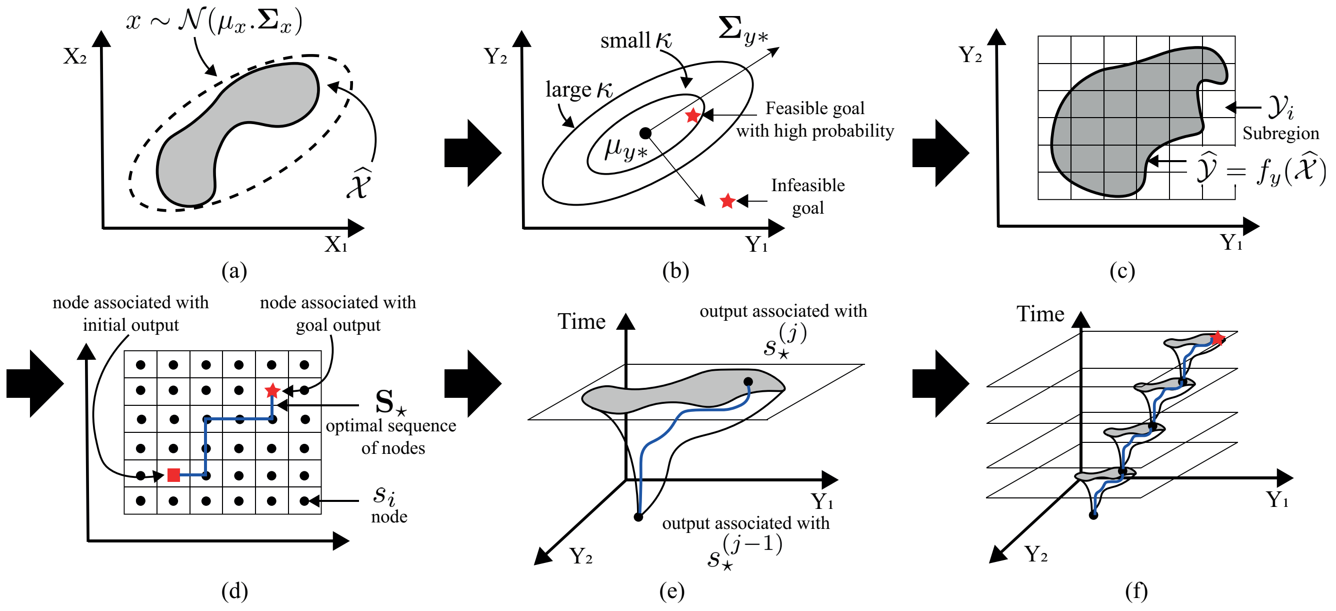
$x \sim \mathcal{N}(\mu_x.\boldsymbol{\Sigma}_x)$
X₂
X₁
$\widehat{\mathcal{X}}$
(a)
Y₂
small κ
large κ
$\boldsymbol{\Sigma}_{y*}$
μ_{y*}
Feasible goal with high probability
Infeasible goal
Y₁
(b)
Y₂
$\mathcal{Y}_i$
Subregion
$\widehat{\mathcal{Y}} = f_y(\widehat{\mathcal{X}})$
Y₁
(c)
node associated with initial output
node associated with goal output
$\mathbf{S}_\star$
optimal sequence of nodes
s_i
node
(d)
Time
output associated with $s_\star^{(j)}$
Y₁
Y₂
output associated with $s_\star^{(j-1)}$
(e)
Time
Y₁
Y₂
(f)

Fig. 1. Proposed method: (a) Generating random state vectors and obtaining a set of sample-wisely feasible states, Section 3.1, (b) Checking the feasibility goal output via output approximation, Section 3.2, (c) Defining output subregions and discrete node space, (d) Solving POMDP to obtain an optimal sequence of nodes, Section 4, (e) Obtaining reachable sets and optimizing trajectories between sequential nodes, Section 5 and 6 (f) Consecutively executing trajectory optimizations with reachable sets, Section 5 and 6

approach aims to obtain feasible states using sampling with respect to the given constraints as shown in Fig. 1(a), then to approximately check whether the goal output is feasible via output approximation as shown in Fig. 1(b). If the goal is achievable, we formulate and solve a POMDP process to create multiple tractable sub problems as shown in Fig. 1(c) and (d). Based on an optimal sequence of nodes, we obtain reachable sets and perform trajectory optimization between neighboring subregions associated with two sequential nodes, then we recursively iterate this process for all sequence of nodes until reaching the final goal output as shown in Fig 1(e) and (f). By connecting all trajectories, we obtain the entire trajectory in an efficient way.

3 Sample-Based Optimization

In this section, we obtain a sequence of subregions in output space considering initial and goal states. Our method breaks down the end-to-end trajectory generation problem into multiple intermediate points sequentially connected. We start by creating random samples from a Gaussian distribution $x \sim \mathcal{N}(\mu_x, \boldsymbol{\Sigma}_x)$ where $\mu_x \in \mathbb{R}^{n_x}$ and $\boldsymbol{\Sigma}_x \in \mathbb{S}^{n_x}_{>0}$ are the mean and the covariance matrix, respectively, that is, $\mu_x := \mathbb{E}[x]$ and $\boldsymbol{\Sigma}_x := \mathbb{E}[(x - \mu_x)(x - \mu_x)^\top]$.

3.1 Update for Random State Samples

We consider a set of states fulfilling desired constraints. Since the Monte-Carlo method is very inefficient, we apply a least-square QP process to obtain feasible states. Let us consider n_e^c state equality constraints and n_i^c state inequality constraints. We define constraint functions as $h_{x,e[k_e]}(x)$ and $h_{x,i[k_i]}(x)$ where $k_e \in [1, n_e^c]_\mathbb{N}$ and $k_i \in [1, n_i^c]_\mathbb{N}$ are indices. Let us define Jacobian matrices and error vectors as follows:

$$\mathbf{J}_e(x) := \text{Vertcat}(J_{e[k]}(x) : \forall k \in [1, n_e^c]_\mathbb{N}) \tag{5a}$$
$$\mathbf{J}_i(x) := \text{Vertcat}(J_{i[k]}(x) : \forall k \in [1, n_i^c]_\mathbb{N}) \tag{5b}$$
$$\mathbf{e}_e(x) := -\text{Vertcat}(h_{x,e[k]}(x) : \forall k \in [1, n_e^c]_\mathbb{N}) \tag{5c}$$
$$\mathbf{e}_i(x) := v_{\setminus i}^{int} - \text{Vertcat}(h_{x,i[k]}(x) : \forall k \in [1, n_i^c]_\mathbb{N}) \tag{5d}$$

where $J_{e[k]}(x) := \frac{\partial h_{x,e[k]}}{\partial x}(x)$ and $J_{i[k]}(x) := \frac{\partial h_{x,i[k]}}{\partial x}(x)$ denote the Jacoabians for equality and inequality constraint functions, respectively. $v_{\setminus i}^{int}$ is an arbitrary interior vector satisfying inequality constraints used as an attractor. The main idea for obtaining the state fulfilling constraints is to iteratively update the sampled state using the state increment Δx until the constraints are satisfied. The state increment Δx is obtained by using the QP method as follows:

$$\begin{aligned} \min_{\Delta x, w} \quad & \|w\|_2^2 \\ \text{s.t} \quad & \mathbf{J}_e(x)\Delta x \le \mathbf{e}_e(x) + w \\ & \mathbf{J}_i(x)\Delta x \le \mathbf{e}_i(x) \end{aligned} \tag{6}$$

and we update the state x using the optimal variable Δx. For numerical efficiency, we discard state samples if QP does not converge. Let us define a set $\mathcal{X}$ consisting of states fulfilling the constraints as follows:

$$\mathcal{X} := \{x \in \mathbb{R}^{n_x} : \|h_{x,e[k_e]}(x)\| \leq \varepsilon,\ \forall k_e \in [1, n_e^c]_{\mathbb{N}} \quad h_{x,i[k_i]}(x) \leq 0,\ \forall k_i \in [1, n_i^c]_{\mathbb{N}}\} \tag{7}$$

where ε is a desired tolerance.

For the next step, we take all elements of $\mathcal{X}$ and check whether they fulfill the input, mixed state-input, and contact force constraints. To achieve this, we formulate an optimization problem with a quadratic cost function as follows:

$$\begin{aligned} \min_{F_c} \quad & F_c^\top \mathbf{Q}_c F_c + u^\top \mathbf{Q}_u u \\ \text{s.t.} \quad & x(t_{k+1}) = f_{\mathfrak{D}}(x(t_k), u, F_c) \\ & h_{u[k_u]}(u) \leq 0,\ \forall k_u \in [1, n_u^c]_{\mathbb{N}} \\ & h_{xu[k_{xu}]}(x(t_k), u) \leq 0,\ \forall k_{xu} \in [1, n_{xu}^c]_{\mathbb{N}} \\ & h_{xc}(x(t_k), F_c) \leq 0,\ x(t_k), x(t_{k+1}) \in \mathcal{X} \end{aligned} \tag{8}$$

where $\mathbf{Q}_c \in \mathbb{S}_{>0}^{n_c}$ and $\mathbf{Q}_u \in \mathbb{S}_{>0}^{n_u}$ are weighting matrices for the cost. By solving the optimization problem (8) for all $x(t_k) \in \mathcal{X}$, we obtain a set of states, $\hat{\mathcal{X}}$, that fulfills all desired constraints.

3.2 Output Space Approximation

In this subsection, we check the feasibility of reaching the desired goal output. To do so, we approximate the output samples with a Gaussian distribution $y^* \sim \mathcal{N}(\mu_{y^*}, \mathbf{\Sigma}_{y^*})$ [17]. The mean and covariance matrix obtained after neglecting higher order terms are

$$\mu_{y^*} := f_y(\mu_x) + \text{Vec}\left[\text{tr}(\mathbf{H}_{y,i}(\mu_x)\mathbf{\Sigma}_x)\right]_{i=1}^{n_y} \tag{9a}$$

$$\mathbf{\Sigma}_{y^*} := \mathbf{J}_y(\mu_x)\mathbf{\Sigma}_x\mathbf{J}_y^\top(\mu_x) + \frac{1}{2}\text{Mat}\left[\text{tr}(\mathbf{\Sigma}_x\mathbf{H}_{y,i}(\mu_x)\mathbf{\Sigma}_x\mathbf{H}_{y,j}(\mu_x))\right]_{i,j=1}^{n_y,n_y} \tag{9b}$$

where $\mathbf{J}_y(\mu)$ and $\mathbf{H}_{y,i}(\mu)$ denote the Jacobian matrix of the output function $f_y(\mu)$ and the 2nd derivative matrix of the output function $f_{y,i}(\mu)$, for the i-th element. In particular,

$$\mathbf{J}_y(\mu) := \frac{\partial f_y}{\partial x}(\mu), \mathbf{H}_{y,i}(\mu) := \begin{bmatrix} \frac{\partial^2 f_{y,i}(\mu)}{\partial x_1^2} & \cdots & \frac{\partial^2 f_{y,i}(\mu)}{\partial x_1 x_{n_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_{y,i}(\mu)}{\partial x_{n_x} x_1} & \cdots & \frac{\partial^2 f_{y,i}(\mu)}{x_{n_x}^2} \end{bmatrix} \tag{10}$$

where $\mathbf{J}_u(\mu) \in \mathbb{R}^{n_y \times n_x}$ and $\mathbf{H}_{y,i}(\mu) \in \mathbb{R}^{n_x \times n_x}$ is a symmetric matrix. We construct a probabilistic ellipsoid in the output space to approximate whether an output sample y^* is feasible. We define a set of outputs that lie inside an ellipsoid $\mathcal{E}_\kappa$ with

$$\mathcal{E}_\kappa := \{y \in \mathbb{R}^{n_y} : (y - \mu_{y^*})^\top \mathbf{\Sigma}_{y^*}^{-1}(y - \mu_{y^*}) \leq \kappa\} \tag{11}$$

where κ is a coefficient determined by the cumulative probability of the Chi-square distribution. For instance, $\kappa = 5.991$ for $\Pr(y^* \in \mathcal{E}_\kappa) = 0.95$ and $y^* \in \mathbb{R}^2$. Our method to check if a goal output y_g is interior to $\mathcal{E}_\kappa$ is more efficient than using a Monte Carlo method, because we only need to compute μ_{y^*} and $\mathbf{\Sigma}_{y^*}$ using the mean and covariance matrix of the samples using (9).

4 POMDP for a Sequence of Subregions

After checking that the desired output goal y_g is located at the interior of the ellipsoid $\mathcal{E}_\kappa$ in (11), the end-to-end trajectory generation problem is divided using intermediate points in the output space. To start the process, we define output subregions:

$$\mathcal{Y}_i := \{y \in \mathbb{R}^{n_y} : \|y - y_{c,i}\|_\infty < \varepsilon_y\} \tag{12}$$

where $y_{c,i} \in \mathbb{R}^{n_y}$ denotes the center of the output subregion $\mathcal{Y}_i$ and $i \in [1, m]_{\mathbb{N}}$ where $\bigcup_{i \in [1,m]_{\mathbb{N}}} \mathcal{Y}_i = \overline{\mathcal{Y}} \subset \mathbb{R}^{n_y}$. Also, we obtain a set of outputs $\hat{\mathcal{Y}} := \hat{f}_y(\hat{\mathcal{X}}) = \{y \in \mathbb{R}^{n_y} : y = f_y(x), x \in \hat{\mathcal{X}}\}$ where $\hat{f}_y : \hat{\mathcal{X}} \rightrightarrows \hat{\mathcal{Y}}$. To formulate our problem as a POMDP, we define discrete nodes associated with the previous subregions as follows:

$$s_i = \text{node}(\mathcal{Y}_i), \quad i \in [1, m]_{\mathbb{N}} \tag{13}$$

where $\mathbf{S} := \{s_1, \ldots, s_m\}$. Based on these nodes, we transform the problem to a POMDP. We will formulate the probability of observations using the sampled states.

Definition 1. *(POMDP) Partially Observable Decision Making Process is defined as a tuple* $\mathbf{P} = (\mathbf{S}, \mathbf{A}, \mathbf{O}, \mathbf{T}, \mathbf{Z})$*:*

- $\mathbf{S}$ *is a finite set of nodes,* $\mathbf{S} := \{s_1, \cdots, s_{m_s}\}$
- $\mathbf{A}$ *is a finite set of actions,* $\mathbf{A} := \{a_1, \cdots, a_{m_a}\}$
- $\mathbf{O}$ *is a finite set of observations,* $\mathbf{O} := \{o_1, \cdots, o_{m_o}\}$
- $\mathbf{T}$ *is the transition dynamics* $\mathbf{T}(s', s, a)$ *defining the transition from* $s \in \mathbf{S}$ *to* $s' \in \mathbf{S}$ *after taking an action* $a \in \mathbf{A}$.
- $\mathbf{Z}$ *is the observation* $\mathbf{Z}(s, a, o)$ *consisting of the probability of observing* $o \in \mathbf{O}$ *after taking an action* $a \in \mathbf{A}$ *from node* $s \in S$.

The problem concerning this section is on finding a sequence of feasible subregions towards an output goal using POMDP tools and analysis.

Definition 2. *(Markov Policy) A Markov policy* $\mathbf{\Pi}$ *is defined as a sequence:* $\mathbf{\Pi} := \{a^{(1)}, \cdots, a^{(n)}\}$. $a^{(j)} \in \mathbf{A}$, *where* $a^{(j)} : \mathbf{S} \to \mathbf{S}$ *is a measurable map from a node to another one,* $j \in [1, n]_{\mathbb{N}}$.

We convert the POMDP into a belief MDP. Belief $b[s_i]$ is defined with respect to discrete nodes $s_i \in \mathbf{S}$. Let suppose $b = b[s^{(j)}]$, $b' = b[s^{(j+1)}]$, and $a = a^{(j)}$ where

$s^{(j)}$ represents the node for the j-th step of the POMDP. The belief transition function, $\Gamma(b,a,b')$, is equal to

$$\Gamma(b,a,b') = \sum_{o\in\mathbf{O}} \Pr(b'|b,a,o)\Pr(o|b,a) \tag{14a}$$

$$\Pr(b'|b,a,o) = \begin{cases} 1, \text{ if belief update returns } b' \\ 0, \text{ otherwise} \end{cases} \tag{14b}$$

$$\Pr(o|b,a) = \sum_{s'\in\mathbf{S}} \mathbf{Z}(s',a,o)\sum_{s\in\mathbf{S}} \mathbf{T}(s',s,a)b. \tag{14c}$$

The key challenges of this POMDP are on defining useful observations and on finding their conditional probability. Let us consider that $\mathcal{Y} = \mathcal{Y}^{(j)}$ and $\mathcal{Y}' = \mathcal{Y}^{(j+1)}$ associated with the nodes $s^{(j)}$ and $s^{(j+1)}$. We propose to define observations as the set of feasible states after taking an action a, i.e.

$$\begin{aligned}\hat{O} :=\{&z \in \mathbb{R}^{n_x} : z = x(t_k),\ \exists(F_c,u) \text{ in (8) with} \\ &f_y(x(t_k)) \in \mathcal{Y}\cap\hat{\mathcal{Y}},\ f_y(x(t_{k+1}) \in \mathcal{Y}'\cap\hat{\mathcal{Y}}\}.\end{aligned} \tag{15}$$

where $\mathcal{Y}$ is the subregion before taking the action a. If $z_1 \in \hat{O}$, it holds that there exists at least one sample connecting $f_y(z_1)$ to another output in the subregion $\mathcal{Y}'$ satisfying the constraints. Considering the above observations, we define the conditional probability as

$$\mathbf{Z}(s',a,o) := \Pr(o|s',a) = \text{card}(\hat{O})/\text{card}(\mathcal{Y}'\cap\hat{\mathcal{Y}}). \tag{16}$$

Let us focus on the reward and transition dynamics. As a heuristic, a higher number of feasible samples falling into a subregion implies a higher probability of reaching it. Therefore we define the reward

$$\mathfrak{R}(s_i,a) := K_r\text{card}(\mathcal{Y}_i\cap\hat{\mathcal{Y}})/\text{card}(\hat{\mathcal{Y}}) + \eta_i \tag{17}$$

where $\mathcal{Y}_i$ is the subregion associated with node $s_i \in \mathbf{S}$. $K_r \in \mathbb{R}_{>0}$ and $\eta_i \in \mathbb{R}$ are the gain and reward offset, respectively. We take η_i to be large when $y_g \in \mathcal{Y}_i$. Also, to avoid unsafe output regions we set η_i to large negative values.

Definition 3. *Consider a node $s_i \in \mathbf{S}$ associated with an output subregion $\mathcal{Y}_i$ and a set $\hat{\mathcal{Y}} = f_y(\hat{\mathcal{X}})$. Consider $\mathbf{\Sigma}_{\mathcal{Y}_i}$ being the covariance matrix for the set $\mathcal{Y}_i\cap\hat{\mathcal{Y}}$, that is, $\mathbf{\Sigma}_{\mathcal{Y}_i} = \mathbb{E}[(y-\mu_y)(y-\mu_y)^\top]$ and $\mu_y = \mathbb{E}[y]$ where $y \in \mathcal{Y}_i\cap\hat{\mathcal{Y}}$. A principle singular vector is defined as*

$$\mathcal{V}(s_i) = \text{col}(\mathbf{V}_{\mathcal{Y}_i})_k, \quad \sigma_k(\mathbf{\Sigma}_{\mathcal{Y}_i}) = \overline{\sigma}(\mathbf{\Sigma}_{\mathcal{Y}_i}). \tag{18}$$

where $\mathbf{\Sigma}_{\mathcal{Y}_i} = \mathbf{V}_{\mathcal{Y}_i}^\top\mathbf{\Lambda}_{\mathcal{Y}_i}\mathbf{V}_{\mathcal{Y}_i}$, $\mathbf{\Lambda}_{\mathcal{Y}_i} = \text{diag}(\sigma_1,\dots,\sigma_{n_y})$, and σ_k denotes the singular value of $\mathbf{\Sigma}_{\mathcal{Y}_i}$.

For defining the transition dynamics, let an action $a \in \mathbf{A}$ map state $s \in \mathbf{S}$ to $s' \in \mathbf{S}$. d is a vector in our grid world defined as $d := (y_c' - y_c)$ where y_c and y_c' are the centers of subregions associated with s and s', respectively. We define the transition dynamics as

$$\mathbf{T}(s',s,a) := \begin{cases} \mathcal{T}(s',s,a)/\varpi, & \text{if } \varpi \neq 0, \\ 0, & \text{else} \end{cases} \tag{19a}$$

$$\mathcal{T}(s',s,a) := \max\{0, \mathcal{V}^\top(s)\mathbf{a}\} \tag{19b}$$

where $\varpi = \sum_{a'\in\mathbf{A}}\mathcal{T}(s',s,a')$ denotes a normalization constant.

Proposition 1. *Let $s = s_i \in \mathbf{S}$ and the corresponding subregion be $\mathcal{Y}_i$. If the output samples are uniformly distributed in $\mathcal{Y}_i$, then $\mathbf{T}(s',s,a) = 0$.*

Proof. Since the samples are uniformly distributed, it is possible to select any unit vector in $\mathbb{R}^{n_y}$ as the PSV of s_i, that is, $\mathbf{R}_a\mathcal{V}(s_i)$ where $\mathbf{R}_a \in \text{SO}(3)$ is a rotation matrix. If we select the rotation matrix $\mathbf{R}_a$ such that $d^\perp = \mathbf{R}_a\mathcal{V}(s_i)$, which is orthogonal to $\mathcal{V}(s_i)$, it follows that $\mathcal{T}(s',s,a) = \mathcal{V}^\top(s_i)d = (d^\perp)^\top d = 0$ for all $a \in \mathbf{A}$. □

We now solve a finite-horizon belief MDP. The optimal policy, denoted by $\mathbf{\Pi}_\star$, is obtained by solving the Bellman equation as follows:

$$\mathfrak{D}_\star(b) = \max_{a\in\mathbf{A}}[r(b,a) + \gamma\sum_{o\in\mathbf{O}}\Pr(o|b,a)\mathfrak{D}_\star(\Gamma(b,a,o))] \tag{20}$$

where $r(b,a) = \sum_{s\in\mathbf{S}} b(s)\mathfrak{R}(s,a)$ denotes the belief reward. The result of the DP provides an optimal Markov policy which we transform to a sequence of nodes as

$$\mathbf{\Pi}_\star = \{a_\star^{(1)},\cdots,a_\star^{(n_\pi)}\} \tag{21a}$$

$$\mathbf{S}_\star = \{s_\star^{(1)},\cdots,s_\star^{(n_\pi)}\} \tag{21b}$$

where $a_\star^{(i)}$ and $s_\star^{(i)}$ are i-th action and node in a sequence toward reaching the final output goal, respectively. The sequence of subregions in output space is

$$\mathbf{Y}_\star = \{\mathcal{Y}_\star^{(1)},\cdots,\mathcal{Y}_\star^{(n_\pi)}\}. \tag{22}$$

Based on the generated sequence of subregions, we will generate trajectories using reachability analysis connecting subregions in $\mathbf{Y}_\star$ in the next section.

5 Reachability Analysis

In this section, we define discrete reachable sets and propose the way to obtain the reachable sets via optimizations. To overcome the computational complexity of propagation algorithm for the reachable sets, a method propagating the boundary samples is proposed and analyzed in the views of computational complexity.

5.1 Optimization-based Reachability Analysis

We define reachable sets in continuous time domain as

$$\begin{aligned}\mathcal{R}_x^C(t,x_0) \coloneqq \{x(t) : &\exists u([t_0,t]), \exists F_c([t_0,t])\\ &h_{x[k_x]}(x(\tau)) \le 0,\ \forall k_x \in [1,n_x^c]_{\mathbb{N}}\\ &h_{xu[k_{xu}]}(x(\tau),u(\tau)) \le 0,\ \forall k_{xu} \in [1,n_{xu}^c]_{\mathbb{N}}\\ &h_{xc}(x(\tau),F_c(\tau)) \le 0,\ u(\tau) \in \mathcal{U}\\ &\dot{x}(\tau) = f_{\mathcal{C}}(x(\tau),u(\tau),F_c(\tau))\\ &x(t_0) = x_0 \in \mathcal{R}_x^C(t_0,x_0),\ \tau \in [t_0,t]\}\end{aligned} \tag{23}$$

$\mathcal{R}_x^C(t_0,x_0)$ is equal to $\{x_0\}$ which means that the initial state fulfills all constraints.

Definition 4. *Let $x_0 \in \mathcal{R}_x^C(t_0,x_0)$, be an initial state and $t \in [t_0,t_f]_d^{\Delta t}$ be an arbitrary time interval. We define a reachable set in discrete time domain as:*

$$\begin{aligned}\mathcal{R}_x^D(t,x_0) \coloneqq \{x(t) : &\exists u([t_0,t]_d^{\Delta t}), \exists F_c([t_0,t]_d^{\Delta t})\\ &h_{x[k_x]}(x(\tau)) \le 0,\ \forall k_x \in [1,n_x^c]_{\mathbb{N}}\\ &h_{xu[k_{xu}]}(x(\tau),u(\tau)) \le 0,\ \forall k_{xu} \in [1,n_{xu}^c]_{\mathbb{N}}\\ &h_{xc}(x(\tau),F_c(\tau)) \le 0,\ u(\tau) \in \mathcal{U}\\ &x(\tau+\Delta t) = f_{\mathfrak{D}}(x(\tau),u(\tau),F_c(\tau))\\ &x(t_0) = x_0 \in \mathcal{R}_x^D(t_0,x_0),\ \tau \in [t_0,t]_d^{\Delta t}\}\end{aligned} \tag{24}$$

where $\Delta t > 0$ is the discretization step or sampling period for our discrete model.

We extend the reachable set defined above for the finite discrete time interval $[t_0,t_f]_d^{\Delta t}$ as $\mathcal{R}_x^D([t_0,t_f]_d^{\Delta t},x_0) \coloneqq \bigcup_{t\in[t_0,t_f]_{d,\Delta t}} \mathcal{R}_x^D(t,x_0)$. For any $t_f < \infty$, the reachable set satisfies the following bound $\mathcal{R}_x^D([t_0,t_f]_d^{\Delta t},\ x_0) \subseteq \mathcal{R}_x^D([t_0,+\infty),x_0) \subseteq \mathcal{R}_x^C([t_0,+\infty),x_0) \subsetneq \mathcal{X}$.

Consider x_0 and $\mathcal{R}_x^D([t_0,t_k]_d^{\Delta t},x_0)$. A random input is drawn from a Gaussian distribution $u \sim \mathcal{N}(\mu_u,\mathbf{\Sigma}_u)$ at each instant of time with the input set $\mathcal{U}$ defined as the collection of inputs fulfilling input constraint. We define a QP to check for feasible contact forces, i.e.

$$\begin{aligned}\min_{F_c,\Delta x}\quad & F_c^\top \mathbf{Q}_c F_c + \Delta x^\top \mathbf{Q}_x \Delta x\\ \text{s.t.}\quad & x(t_{k+1}) = f_{\mathfrak{D}}(x(t_k),u(t_k),F_c)\\ & x(t_k) \in \mathcal{R}_x^D([t_0,t_k]_d^{\Delta t},x_0)\\ & h_{x[k_x]}(x(t_{k+1})) \le 0,\ \forall k_x \in [1,n_x^c]_{\mathbb{N}}\\ & h_{xu[k_{xu}]}(x(t_k),u(t_k)) \le 0,\ \forall k_{xu} \in [1,n_{xu}^c]_{\mathbb{N}}\\ & h_{xc}(x(t_k),F_c) \le 0,\ u(t_k) \in \mathcal{U}\\ & \Delta x = x(t_{k+1}) - x(t_k)\end{aligned} \tag{25}$$

The reachable set at the next time instance t_{k+1} is obtained by collecting the $x(t_{k+1})$ updated by the QP solution Δx in (25). Then, we can extend the reachable set over $[t_0,t_{k+1}]_d^{\Delta t}$ such that $\mathcal{R}_x^D([t_0,t_{k+1}]_d^{\Delta t},x_0) \coloneqq \mathcal{R}_x^D([t_0,t_k]_d^{\Delta t},x_0) \cup \mathcal{R}_x^D(t_{k+1},x_0)$. Also, we define the set of outputs associated with reachable states such as $\mathcal{R}_y^D(t,x_0) = \hat{f}_y(\mathcal{R}_x^D(t,x_0))$ and $\mathcal{R}_y^D([t_0,t_f]_d^{\Delta t},x_0) = \hat{f}_y(\mathcal{R}_x^D([t_0,t_f]_d^{\Delta t},x_0))$.

Theorem 2. *Suppose that $\mathcal{R}_x^D(t_0,x_0)$ is a non-empty set and the desired output, y_d, is given. Let us assume that the set, $\mathcal{R}_y^D([t_0,t_f]_d^{\Delta t},x_0)$, is compact, connected, and*

$$y_d \in \mathcal{R}_y^D([t_0,t_f]_d^{\Delta t},x_0). \tag{26}$$

At least, one state trajectory $\mathbf{\Psi} \coloneqq \{\xi(t_0),\xi(t_1),\ldots,\xi(\tau)\}$ exists such that $f_y(\xi(\tau)) = y_d$ where $\tau \le t_f$.

Proof. Since $\mathcal{R}_y^D([t_0,t_f]_d^{\Delta t},x_0)$ is compact and f_y is continuous, $\mathcal{R}_x^D([t_0,t_f]_d^{\Delta t},x_0)$ is closed and f_y^{-1} is also continuous. Then, $\mathcal{R}_x^D([t_0,t_f]_d^{\Delta t},x_0)$ is connected because $\mathcal{R}_y^D([t_0,t_f]_d^{\Delta t},x_0)$ is connected and f_y^{-1} is continuous. Therefore, there exists at least one trajectory connecting x_0 to x_f satisfying $f_y(x_f) = y_d$ in $\mathcal{R}_x^D([t_0,t_f]_d^{\Delta t},x_0)$. □

Corollary 1. *The set, $\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0)$, is compact.*

Proof. Let us consider ghull($\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0)$), being compact. By the Heine−Borel theorem, all closed subsets of a compact set are also compact. Since $\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0) \subset$ ghull($\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0)$), the reachable set $\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0)$ is compact. □

Corollary 2. *Suppose that $x_0 \in \mathcal{R}_x^D(t_0,x_0)$ and f_y is continuous. Then, a set, $\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0)$, is connected.*

Proof. Consider three sets: $\mathcal{F}_1 = \mathcal{R}_x^D([t_0,t_{k-1}]_d^{\Delta t},x_0)$, $\mathcal{F}_2 = \mathcal{R}_x^D(t_k,x_0)$, and $\mathcal{F}_3 = \mathcal{F}_2 \cup \mathcal{F}_1'$, where $\mathcal{F}_1'$ is the collection of states $x \in \mathcal{F}_1$ producing the next feasible state via the optimization (27) with respect to $x \in \mathcal{F}_1$. Then, $\mathcal{R}_x^D([t_0,t_k]_d^{\Delta t},x_0) = \mathcal{F}_1 \cup \mathcal{F}_2 = \mathcal{F}_2 \cup \mathcal{F}_3$. Let us consider arbitrary two sets $\mathcal{H}_1$ and $\mathcal{H}_2$ satisfying $\mathcal{R}_x^D([t_0,t_k]_d^{\Delta t},x_0) = \mathcal{H}_1 \cup \mathcal{H}_2$ with $\mathcal{H}_1 \cap \mathcal{H}_2 = \emptyset$. Let $x_h \in F_1'$ and suppose $x_h \in \mathcal{H}_1$. Then, $\mathcal{H}_1 \cap \mathcal{F}_1 \neq \emptyset$ and $\mathcal{H}_1 \cap \mathcal{F}_3 \neq \emptyset$. This implies that $\mathcal{F}_1 \subseteq \mathcal{H}_1$ and $\mathcal{F}_3 \subseteq \mathcal{H}_1$, hence, $\mathcal{H}_2 = \emptyset$. This proves that $\mathcal{R}_x^D([t_0,t_k]_d^{\Delta t},x_0)$ is connected. Since the mapping f_y is continuous, we also conclude that the set $\mathcal{R}_y^D([t_0,t_k]_d^{\Delta t},x_0)$ is connected. □

5.2 Propagation of Boundary States

The basic algorithm for reachability analysis suffers from exponential complexity with respect to the number of time steps. Although the previous POMDP contributes to reducing the time horizon to be checked for reachability analysis, full-state propagation would still result in heavy computational burden. In this section, we propose a method for reducing the computational complexity of the algorithm by only propagating selected states. This approach results in more conservative reachable sets. We implement the propagation of boundary states by solving (25) for only state samples $x(t_k) \in \mathcal{B}_x^D([t_0, t_k]_d^{\Delta t}, x_0)$ such that

$$\mathcal{B}_x^D([t_0, t_k]_d^{\Delta t}, x_0) \coloneqq \{x \in \mathbb{R}^{n_x} : x \in \mathcal{R}_x^D([t_0, t_k]_d^{\Delta t}, x_0), \\ f_y(x) \in \text{gbd}(\mathcal{R}_y^D([t_0, t_k]_d^{\Delta t}, x_0))\}. \tag{27}$$

We then compute the reachable set by collecting the updated states and extend the time horizon. In this way, the complexity becomes linear with respect to the number of boundary samples, $\text{card}(\overline{\mathcal{B}}^D([t_0, t_k]_d^{\Delta t}, x_0))$. In order to replace full-state propagation with boundary-state propagation, we show that the set of reachable outputs is compact and connected. First, the set $\overline{\mathcal{R}}_y^D([t_0, t_k]_d^{\Delta t}, x_0)$ is compact, because we are able to obtain the hull of the set as shown in Corollary 1. Next, we prove the reachable set $\mathcal{R}_y^D$ is connected.

Corollary 3. *Suppose that $x_0 \in \mathcal{R}_x^D(t_0, x_0) \neq \emptyset$ and f_y is continuous. Then, the set, $\overline{\mathcal{R}}_y^D([t_0, t_k]_d^{\Delta t}, x_0)$, is connected*

Proof. The proof is similar to that of Corollary 2 and therefore is omitted. □

5.3 Computational Complexity Analysis

We analyze the computational complexity to compare the efficiency of the propagation of full states and that of boundary states. There exists many algorithms to obtain the concave hull from the set of data [10, 11, 33]. They have $\mathcal{O}(n^3)$ or $\mathcal{O}(n \log n)$ time complexity with n data in 2-D space. General QPs are non-deterministic polynomial-time hard, which means the algorithms are more complex than the polynomial time complexity to be solved. In the case that the QP is convex, it is widely known that the time complexity of the QP is $\mathcal{O}(m^3)$ where m is the number of decision variables.

Based on the aforementioned discussion, we can compare the computational complexity of two cases: propagation of full states and propagation of boundary states. Let us consider N_t steps over the time interval $[t_0, t_k]_d^{\Delta t}$ where $\Delta t = (t_k - t_0)/N_t$, and N_u is the number of input samples. For each propagation method, the computational complexity can be represented as $\mathbf{C}_f \sim \mathcal{O}(\sum_{i=1}^{N_t} N_u^i (n_c + n_x)^3) \approx \mathcal{O}\left(N_u^{N_t+1}(n_c + n_x)^3\right)$, and $\mathbf{C}_b \sim \mathcal{O}(\sum_{i=1}^{N_t} N_b(n_c + n_x)^3 + (iN_b)^3) \approx \mathcal{O}(N_t N_b (n_c + n_x)^3 + N_t^4 N_b^3)$ where $\mathbf{C}_f$, $\mathbf{C}_b$, and N_b denote the complexity of full state propagation, that of boundary state propagation, and the number of boundary samples. Normally, a set of boundary samples contains much smaller samples than a set of entire states, that is, $N_b \ll N_u$. The effect of the boundary sampling on computational complexity becomes significantly advantageous in terms of the number of time steps. We will show the comparison of the computational complexity using an example in the simulation section.

6 Optimal Control

In this work, we describe the use of sequential optimal control for nonlinear programs without constraint softening. Instead of considering end-to-end trajectory generation, we focus on finding a trajectory connecting two subregions obtained by the POMDP process described earlier. By iterating this process for connecting subregions (22), the optimal control process is able to attain the desired output with reduced computational cost.

6.1 Nonlinear Programming

In order to formulate the optimal control problem solved by NLP, a performance measure is defined in the discrete time and state space, that is,

$$\begin{aligned} \mathcal{L}(\mathbf{U}, N) &\coloneqq \sum_{k=0}^{N} (u^\top(t_k)\mathbf{Q}_u u(t_k) + F_c^\top(t_k)\mathbf{Q}_c F_c(t_k) \\ &\quad + e_x^\top(t_k)\mathbf{Q}_x e_x(t_k)) + e_y^\top(t_N)\mathbf{Q}_y e_y(t_N) \\ e_x(t_k) &\coloneqq \xi(t_k) - x_0, \quad e_y(t_k) \coloneqq (f_y(\xi(t_k)) - y_d) \end{aligned}$$

where $\mathbf{U} \coloneqq \{u(t_0), u(t_1), \ldots, u(t_N)\}$. In addition, $\mathbf{Q}_x \in \mathbb{S}_{>0}^{n_x}$ and $\mathbf{Q}_y \in \mathbb{S}_{>0}^{n_y}$ denote the weighting matrices for the state and output terms, respectively. $\xi(t) \in \mathbb{R}^{n_x}$ and $y_d \in \mathbb{R}^{n_y}$ denote the trajectory of the state and the desired goal of the output of the NLP, respectively. Based on the set of reachable states, we can formulate the NLP as follows:

$$\begin{aligned} \min_{\mathbf{\Psi}, \mathbf{U}} \quad & \mathcal{L}(\mathbf{U}, N) \\ \text{s.t.} \quad & \xi(t_{k+1}) = f_{\mathfrak{D}}(\xi(t_k), u(t_k), F_c(t_k)) \\ & \xi(t_k) \in \mathcal{R}_x^D([t_0, t_k]_d^{\Delta t}, x_0) \\ & \xi(t_{k+1}) \in \mathcal{R}_x^D([t_0, t_{k+1}]_d^{\Delta t}, x_0) \\ & h_{x[k_x]}(\xi(t_{k+1})) \leq 0, \ \forall k_x \in [1, n_x^c]_{\mathbb{N}} \\ & h_{xu[k_{xu}]}(\xi(t_k), u(t_k) \leq 0, \ \forall k_{xu} \in [1, n_{xu}^c]_{\mathbb{N}} \\ & h_{xc}(\xi(t_k), u(t_k)) \leq 0, \ u(t_k) \in \mathcal{U}. \end{aligned} \tag{29}$$

The proposed approach recursively executes the formulated NLP (29) by changing the initial conditions, the constraints, and the goal output. Let us represent the optimization problem for connecting $\mathcal{Y}_\star^{(j)}$ to $\mathcal{Y}_\star^{(j+1)}$ as

$$(\mathbf{\Psi}^{(j)}, \mathbf{U}^{(j)}) = \text{NLP}(x_0^{(j)}, y_d^{(j)}, t_0^{(j)}, N^{(j)}) \tag{30}$$

where $j \in [1, n_\pi - 1]_{\mathbb{N}}$. We replace $t_0^{(j+1)}$, $x_0^{(j+1)}$ and $y_d^{(j+1)}$ with $t_N^{(j)}$, $\xi^{(j)}(t_N^{(j)})$ and $y_c^{(j+1)}$, respectively.

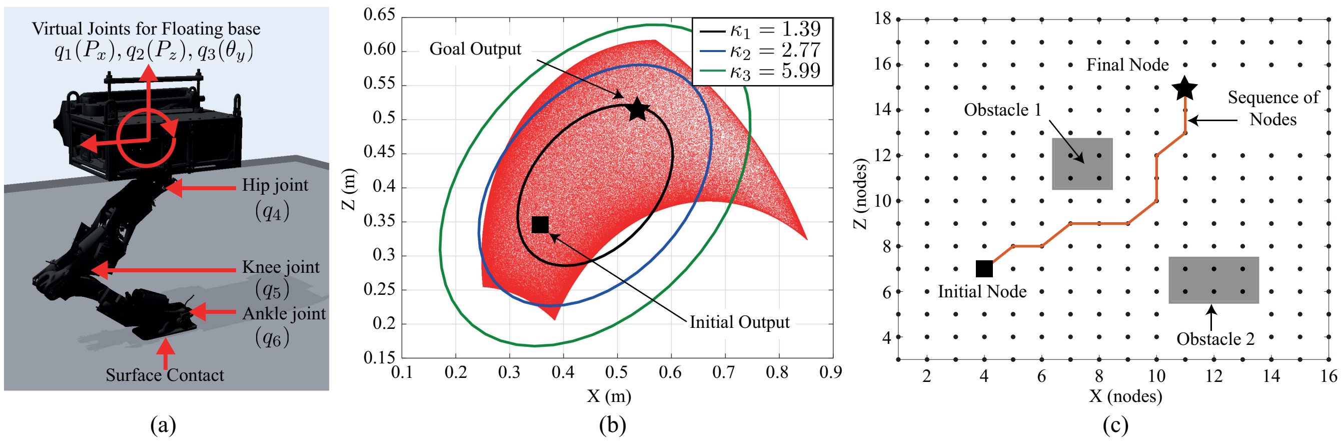

Fig. 2. (a) Legged robotic system with surface contact, consisting of three virtual joints and three actuated joints, (b) A set of output samples satisfying the constraints and the approximated ellipsoids with respect to κ in (11), (c) A sequence of nodes solved by POMDP.

$N^{(j+1)}$ is determined by reachability analysis using $t_0^{(j+1)}$, $x_0^{(j+1)}$, and $y_d^{(j+1)}$. In particular, we set $x_0^{(1)} = x_0$ and $y_d^{(n_\pi-1)} = y_g$. After performing the iterative NLP, we obtain the entire state and input trajectories to reach the goal output such that

$$\mathbf{\Psi}^\star := \{\mathbf{\Psi}^{(1)}, \dots, \mathbf{\Psi}^{(n_\pi-1)}\} \tag{31a}$$
$$\mathbf{U}^\star := \{\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(n_\pi-1)}\}. \tag{31b}$$

The solutions, $\mathbf{\Psi}^\star$ and $\mathbf{U}^\star$, enable the robotic system to reach the goal output state maintaining all constraints and contacts.

7 Numerical Simulations

In this section, we validate the proposed approach by simulating a legged robot *Draco*, which is developed for efficient and dynamic locomotion using liquid-cooled series elastic actuators [21]. The dynamic simulation is implemented by DART [26]. We utilize two optimizers: Goldfarb for QP and IPOPT implementing a primal-dual interior point method [40]. The simulation is executed on a laptop with a Core i7-8650U CPU and 16.0 GB RAM.

7.1 A Robotic System and Constraints

A simulation model of *Draco* consists of three virtual joints, i.e. position and orientation of its floating base (q_1, q_2, q_3) and three actuated joints, i.e. knee and ankle joints (q_4, q_5, q_6) as shown in Fig. 2 (a). For our simulations, the state and input constraints are defined as follows:

$$\begin{aligned} &q_{j_1}^{\ominus} \le q_{j_1} \le q_{j_1}^{\oplus},\ \dot{q}_{j_1}^{\ominus} \le \dot{q}_{j_1} \le \dot{q}_{j_1}^{\oplus},\ u_{j_1}^{\ominus} \le u_{j_1} \le u_{j_1}^{\oplus} \\ &u_{j_2} = 0, \quad j_1 \in \{4,5,6\}, \quad j_2 \in \{1,2,3\} \end{aligned} \tag{32}$$

where superscripts $(.)^{\ominus}$ and $(.)^{\oplus}$ represent the lower and upper limits, respectively. The specific conditions are $q^{\ominus}_{j_4,j_5,j_6} = [-1.2,\ 0.5.\ -1.5]$, $q^{\oplus}_{j_4,j_5,j_6} = [-0.2,\ 2.6,\ -0.5]$, $u^{\ominus}_{j_4,j_5,j_6} = [-1000,\ -1000,\ -1000]$, and $u^{\oplus}_{j_4,j_5,j_6} = [1000,\ 1000,\ 1000]$. The position, orientation, and velocity of the foot should satisfy the kinematic constraints for the contact. We consider a surface contact with rectangular support polygon on the foot so that the friction cone constraints can be characterized as

$$|f_x| \le k_\mu f_z, \quad f_z > 0, \quad |\tau_y| \le d_x f_z \tag{33}$$

where d_x denotes the distance between the center of the polygon and the vertex in the local frame of the foot and k_μ represents the friction coefficient. $F_c := [f_x, f_z, \tau_y]^\top$ is the contact wrench, which is a resultant contact force at the center of the support polygon. Based on the inequality constraints of (33) and the coordinate transformation from local frame on the foot to the global frame, we can represent the friction cone constraints in the form $\mathbf{W}_{local}\mathbf{R}_c(q)F_c = \mathbf{W}_c(q)F_c \le 0$ where $\mathbf{W}_{local}$ is a coefficient matrix derived from (33) and $\mathbf{R}_c(q)$ is a rotational matrix from global to foot frames. In $\mathbf{W}_{local}$, we set the friction coefficient $k_\mu = 0.4$. Considering all constraints, we will control the robot's motion while maintaining the contact.

7.2 Setup for Simulation

To start the reachability analysis, we generate 10^6 state samples and gather the states fulfilling the constraints. The mean and covariance of the Gaussian distribution for sampling are

$$\begin{aligned} \mu_x &= [0.352, 0.384, -0.95, 2.2, -1.25, \mathbf{0}_{1\times6}]^\top \\ \mathbf{\Sigma}_x &= \begin{bmatrix} \pi\mathbf{I}_{6\times6} & \mathbf{0}_{6\times6} \\ \mathbf{O}_{6\times6} & 2\pi\mathbf{I}_{6\times6} \end{bmatrix}. \end{aligned} \tag{34}$$

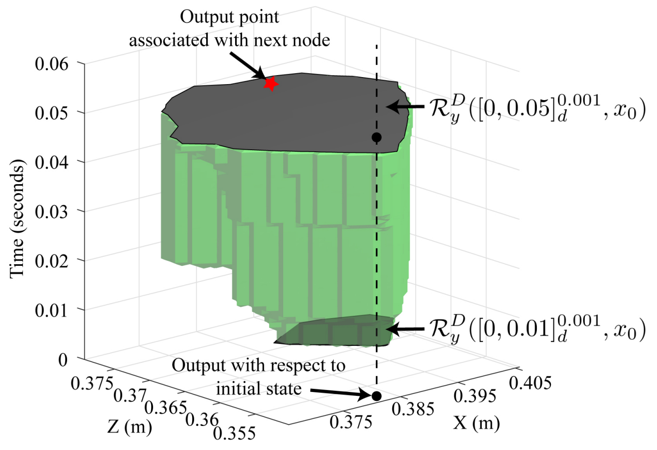

Fig. 3. Reachable sets over a finite time horizon $[0, 0.05]_d^{0.001}$: The reachable sets are obtained by boundary-state propagation.

We define a threshold for numerical convergence (7) with value 1.0×10^{-7} and the maximum number of iterations is 1.0×10^6. After implementing the sampling-based approach described in Section 3, we obtain 3.47×10^5 states among 10^6 state samples. For formulating the POMDP problem in 2D space, we set 40 nodes defined by $\mathbf{S} = \{s_i\}$ where $i \in [1, 40]_\mathbb{N}$ and 8 actions $\mathbf{A} = \{a_j\}$ where $j \in [1, 8]_\mathbb{N}$, and each action consists of moving up, down, right, left, up-right, up-left, down-right, and down-left in the grid world, respectively. We consider two static obstacles for the robot to avoid in the output space. The objective of our numerical simulation is to obtain an optimized trajectory to reach the goal output while avoiding the obstacles and fulfilling all constraints. In addition, we generate 1.0×10^5 input samples for propagating the states in the reachability analysis.

7.3 Simulation Results

First, the results of our sample-based optimization is shown in Fig. 2(b). The set with red dots contains the outputs associated with the states fulfilling the constraints given by the optimization process described in Section 3. As shown in Fig. 2(b), both the initial [0.384, 0.352] and goal [0.51, 0.52] outputs are located at the interior of the feasible set. Then, we solve the POMDP problem to find a sequence of nodes, which result in 12 of them, avoiding the obstacles as shown in Fig. 2(c). After obtaining the sequence of nodes, we obtain the reachable sets as shown in Fig. 3. The reachable set $\mathcal{R}_y^D([0, 0.05]_d^{0.001}, x_0)$ contains the desired output associated with the first node. Based on this result, we solve the NLP (29) to find a trajectory to reach the desired output from the initial configuration.

The computational complexity is analyzed by measuring the execution time of the algorithm for computing the reachable sets. We repeat 10 simulations to measure the computation time for both the full-state and boundary-state propagation methods and display the results in Fig. 4. The algorithm cannot compute the reachable sets via the full-state propagation method for more than two steps due to the lack of memory capacity as shown in the blue dotted line in Fig 4. As we predicted in the complexity analysis of Section 5.3, the boundary-states propagation method significantly reduces the computational time for computing reachable sets.

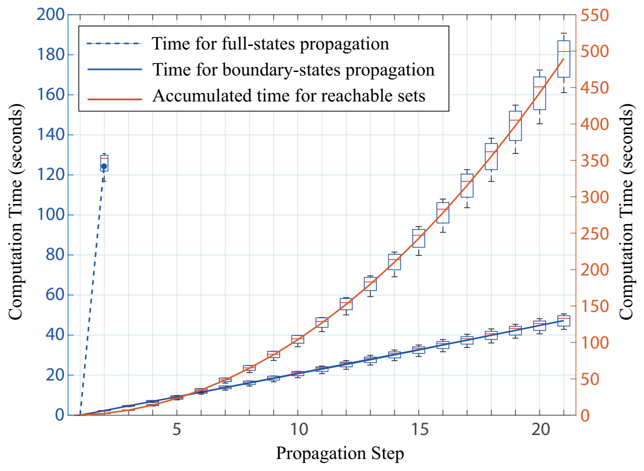

Fig. 4. Computational time for obtaining reachable sets with discretization step $\Delta t = 0.01$ and 10 simulations.

Fig. 5 shows the results of trajectory optimization to reach the goal position with respect to a given initial state. The optimization result includes both joint position and velocity trajectories fulfilling kinodynamic constraints, e.g. joint position, velocity, and torque limits and contact kinematic and force constraints. The optimization results for the actuated joints in the phase space have stabilizable end points which are marked as blue diamonds in Fig. 5(a), (b), and (c). As shown in Fig 5(d), the generated trajectories pass through the subregions in an optimized sequence obtained by solving the POMPD problem and reaching the final output goal position.

8 Conclusion

This paper proposes a method to generate trajectories for complex robotic systems considering contact constraints. We utilize NLP to solve the optimal control problem. Our approach focuses on efficiently solving the NLP problem so that we can scale the method to many types of complex robotic systems. We devise a new approach to obtain discrete-time reachable sets for trajectory generation and solve the nonlinear optimization problem. Although the computational cost is significantly reduced, it is still challenging to employ this approach to real-time control. Therefore, in the near future, we will investigate ways to combine this approach with feedback controllers and extend the proposed method for hybrid dynamical systems, such as

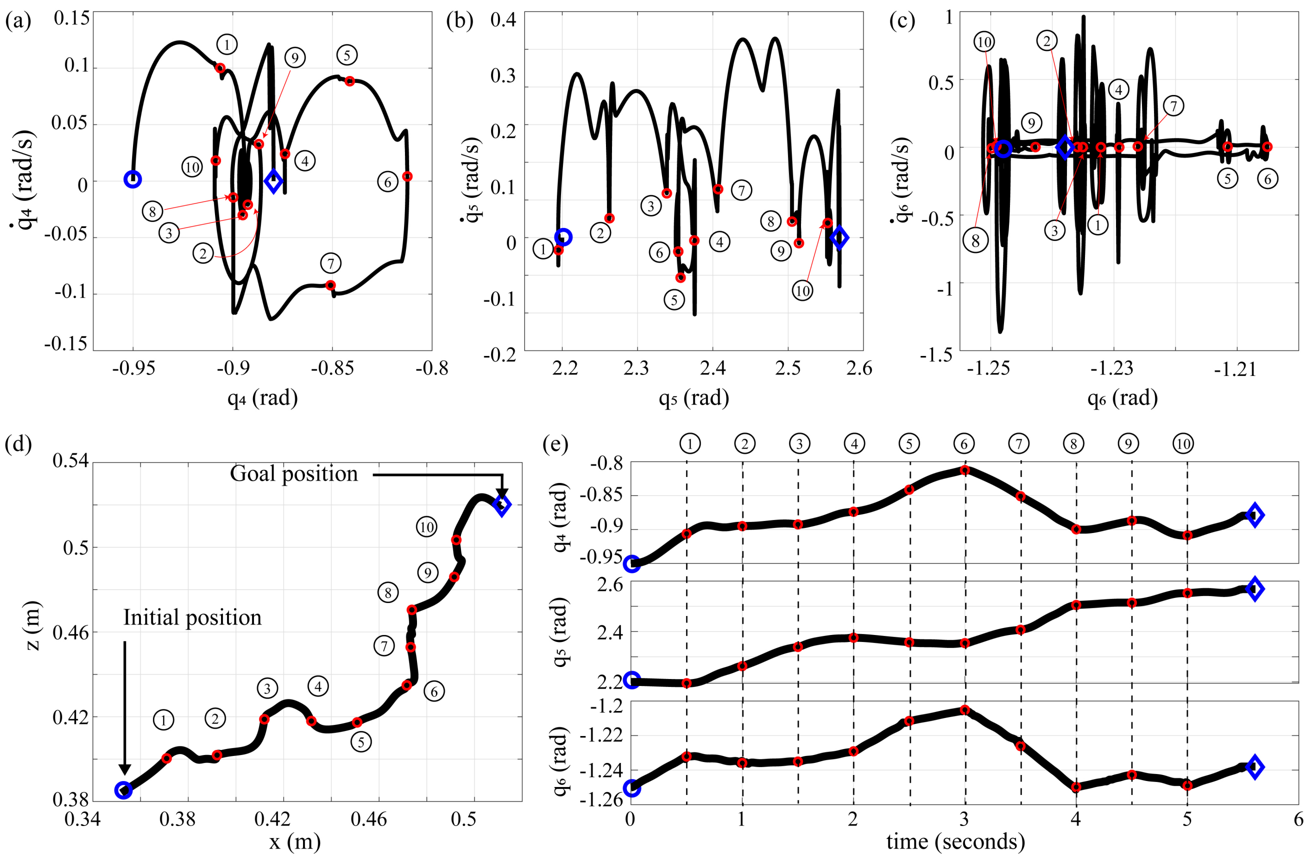

Fig. 5. Simulation results: (a), (b), and (c) illustrate the joint position and velocity trajectories of the actuated joints in the phase space, (d) shows the optimized trajectory in the output space, (e) shows the joint position trajectory of the actuated joints in the time domain. The blue circle and red diamond indicate the initial and final values, respectively.

biped humanoid robots or dual arm manipulators.

Acknowledgements

The authors would like to thank the members of the Human Centered Robotics Laboratory at The University of Texas at Austin for their great help and support. This work was supported by an NSF Grant# 1724360 and partially supported by an ONR Grant# N000141512507.

References

[1] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[2] Matthias Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 173–182. ACM, 2013.

[3] Matthias Althoff and Bruce H Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Transactions on Automatic Control*, 59(2):371–383, 2014.

[4] Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4042–4048. IEEE, 2008.

[5] Murat Arcak and John Maidens. Simulation-based reachability analysis for nonlinear systems using componentwise contraction properties. *arXiv preprint arXiv:1709.06661*, 2017.

[6] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.

[7] Felix Burget and Maren Bennewitz. Stance selection for humanoid grasping tasks by inverse reachability maps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5669–5674. IEEE, 2015.

[8] Jan Carius, René Ranftl, Vladlen Koltun, and Marco Hutter. Trajectory optimization with implicit hard contacts. *IEEE Robotics and Automation Letters*, 3(4):3316–3323, 2018.

[9] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *IEEE International Conference on Robotics and Automation*, pages 5107–5112, 2015.

[10] Matt Duckham, Lars Kulik, Mike Worboys, and Antony

Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern recognition*, 41(10):3224–3236, 2008.

[11] Antony Galton and Matt Duckham. What is the region occupied by a set of points? In *International Conference on Geographic Information Science*, pages 81–98. Springer, 2006.

[12] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[13] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 291–305. Springer, 2005.

[14] Antoine Girard and Colas Le Guernic. Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes*, 41(2):8966–8971, 2008.

[15] LCGJM Habets, Pieter J Collins, and Jan H van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948, 2006.

[16] Abdullah Hamadeh and Jorge Goncalves. Reachability analysis of continuous-time piecewise affine systems. *Automatica*, 44(12):3189–3194, 2008.

[17] Gustaf Hendeby and Fredrik Gustafsson. On nonlinear transformations of gaussian distributions. *Technical Report from Automatic Control at Link? pings Universitet*, 2007.

[18] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *ICRA*, volume 3, pages 1620–1626, 2003.

[19] Nikolaos Kariotoglou, Sean Summers, Tyler Summers, Maryam Kamgarpour, and John Lygeros. Approximate dynamic programming for stochastic reachability. In *Proceedings of European Control Conference*, pages 584–589, 2013.

[20] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[21] Donghyun Kim, Junhyeok Ahn, Orion Campbell, Nicholas Paine, and Luis Sentis. Investigations of a robotic testbed with viscoelastic liquid cooled actuators. *IEEE/ASME Transactions on Mechatronics*, 2018.

[22] Jin-Hoon Kim. Improved ellipsoidal bound of reachable sets for time-delayed linear systems with disturbances. *Automatica*, 44(11):2940–2943, 2008.

[23] Alexander B Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis: internal approximation. *Systems & control letters*, 41(3):201–211, 2000.

[24] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[25] Jaemin Lee, Efstathios Bakolas, and Luis Sentis. Trajectory generation for robotic systems with contact force constraints. *arXiv preprint arXiv:1809.10598*, 2018.

[26] Jeongseok Lee, Michael X Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S Srinivasa, Mike Stilman, and C Karen Liu. Dart: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500, 2018.

[27] Kendra Lesser and Meeko Oishi. Reachability for partially observable discrete time stochastic hybrid systems. *Automatica*, 50(8):1989–1998, 2014.

[28] Yiping Liu, Patrick M Wensing, David E Orin, and Yuan F Zheng. Trajectory generation for dynamic walking in a humanoid over uneven terrain using a 3d-actuated dual-slip model. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 374–380. IEEE, 2015.

[29] John Maidens and Murat Arcak. Reachability analysis of nonlinear systems using matrix measures. *IEEE Transactions on Automatic Control*, 60(1):265–270, 2015.

[30] Moussa Maiga, Nacim Ramdani, Louise Travé-Massuyès, and Christophe Combastel. A comprehensive method for reachability analysis of uncertain nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 61(9):2341–2356, 2016.

[31] Ian Mitchell, Alexandre M Bayen, and Claire J Tomlin. Validating a Hamilton-Jacobi approximation to hybrid system reachable sets. In *International Workshop on Hybrid Systems: Computation and Control*, pages 418–432. Springer, 2001.

[32] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[33] Adriano Moreira and Maribel Yasmina Santos. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. 2007.

[34] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Inverse dynamics control of floating-base robots with external constraints: A unified view. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1085–1090. IEEE, 2011.

[35] Matthias Rungger and Majid Zamani. Accurate reachability analysis of uncertain nonlinear systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 61–70. ACM, 2018.

[36] Joseph K Scott and Paul I Barton. Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49(1):93–100, 2013.

[37] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.

[38] Benjamin J Stephens and Christopher G Atkeson. Dynamic balance force control for compliant humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1248–1255, 2010.

[39] Sean Summers, Maryam Kamgarpour, Claire Tomlin, and John Lygeros. Stochastic system controller synthesis for reachability specifications encoded by random sets. *Automatica*, 49(9):2906–2910, 2013.

[40] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[41] Yiming Yang, Wolfgang Merkt, Henrique Ferrolho, Vladimir Ivan, and Sethu Vijayakumar. Efficient humanoid motion planning on uneven terrain using paired forward-inverse dynamic reachability maps. *IEEE Robotics and Automation Letters*, 2(4):2279–2286, 2017.