Optimizing Dynamic Trajectories for Robustness to Disturbances Using Polytopic Projections

Henrique Ferrolho, Wolfgang Merkt, Vladimir Ivan, Wouter Wolfslag, Sethu Vijayakumar

***Abstract*— This paper focuses on robustness to disturbance forces and uncertain payloads. We present a novel formulation to optimize the robustness of dynamic trajectories. A straightforward transcription of this formulation into a nonlinear programming problem is not tractable for state-of-the-art solvers, but it is possible to overcome this complication by exploiting the structure induced by the kinematics of the robot. The non-trivial transcription proposed allows trajectory optimization frameworks to converge to highly robust dynamic solutions. We demonstrate the results of our approach using a quadruped robot equipped with a manipulator.**

I. Introduction

When an external force is applied to a legged robot with a manipulator it may cause the robot to slip, or to fail to track a path with its end-effector. Similarly, the performance degrades when the robot poorly estimates how slippery the ground is or how heavy is its payload. In either case the motion fails because completing the task while compensating for the external force requires the robot to either command more torque to its actuators than they are capable of delivering, to produce unrealistic contact forces, or both. These limitations impose constraints that the robot motion has to satisfy. Therefore, one way to look at robustness is to define it as some metric of distance to these constraints, for instance, as the force the robot can compensate for before violating the motion constraints. This kind of robustness could be optimized over by the robot controller, however, considering robustness during motion planning would allow us to avoid difficult-to-execute motions altogether.

We tackle the problem of robustness against external perturbations and unmodeled payloads for complex legged robots with manipulation capabilities. We focus on increasing robustness at the planning stage to provide any tracking controller, including robust control schemes, with greater margins of control authority. In previous work [1], we used the smallest unrejectable force (SUF) applied at some link of the robot as a robustness metric for improving single configurations via convex conic optimization. In this work, we propose a novel formulation to make the computation more tractable and versatile, allowing us to consider the optimization of entire trajectories with nonlinear dynamics.

This research is supported by the EPSRC UK RAI Hub for Offshore Robotics for Certification of Assets (ORCA, grant reference EP/R026173/1), EU H2020 project Memory of Motion (MEMMO, project ID: 780684), and the EPSRC Centre for Doctoral Training in Robotics and Autonomous Systems (EPSRC, grant reference EP/L016834/1).

All authors are with the School of Informatics, University of Edinburgh, United Kingdom. Wolfgang Merkt is also with the Oxford Robotics Institute, University of Oxford, United Kingdom.

Email: henrique.ferrolho@ed.ac.uk.

Fig. 1: A legged loco-manipulation system: ANYmal [2] is a fully torque-controlled quadruped robot. We equipped it with a Kinova Jaco [3] robot arm. An accompanying video is available at https://youtu.be/vDesP7IpThw.

Our new computational framework enables us to combine trajectory optimization (TO) with the SUF metric to produce highly robust and dynamic trajectories.

A. Related Work

[4] presented a motion planning and control framework for a platform similar to ours (see Fig. 1). The authors demonstrated successful execution of tasks such as opening a door and carrying a box alongside a human. The authors addressed robustness to external disturbances with an inverse dynamics-based whole-body controller and by re-planning locomotion continuously in a receding-horizon fashion. However, contrary to our approach, they did not take into account robustness explicitly at the planning-level.

[5] proposed a solution to improve the robustness to joint-torque tracking errors at the control stage. The authors modeled deterministic and stochastic uncertainties in joint torques within their control framework optimization. Their idea is similar to what we present in this paper, but we maximize the upper-bound force magnitude the system can withstand from any possible direction—and we do this during planning.

The authors of [6] included external forces estimation directly into their hierarchical controller. Their objective was to minimize actuator torques while enforcing constraints for the contact forces. However, contrary to our work, they did not enforce actuator limitations explicitly.

Modeling the capabilities of the system explicitly using polytopes has recently become more popular than using

simplified metrics for robustness. In [7], the authors derived the equations of a so-called Gravito-Inertial Wrench Cone (GIWC). It is a feasible region used as a general stability criterion. This representation is very efficient for testing robust static equilibrium of a legged robot, but it fails to take into account any actuation limits. [8] proposed to incorporate the properties of [7] with system torque limits. They use the resulting polytopes to optimize the Center of Mass (CoM) trajectory in the xy-plane for the base-transfer motion of a quadruped. Despite the reduced size of this problem, the technique used to compute polytopes was prohibitively expensive, and as a workaround they computed polytopes once at the beginning and used them as an approximation for the remaining motion.

We have followed this line of research in our previous work [9] and we proposed a force polytope representation considering system dynamics: the *residual force polytope*. The polytope is computed from the forces and torques remaining after accounting for Coriolis, centrifugal, and gravity terms, as well as nominal motion feed-forward torques.

All the polytope calculations proposed in the literature [10], [8], [9] require a significant amount of computation time. In general, deriving an explicit description of a projected polytope is NP-hard [11]. As a result, prior work using polytopes in trajectory optimization, e.g., [8], resorted to the approximation of fixing the polytope for an entire trajectory.

[12] recently formulated a computationally tractable approach for finding maximally sized convex bodies inscribed in a projected polytope. Their scheme does not require an explicit description of the projection and works by combining Fourier-Motzkin elimination with techniques from adjustable robust optimization. The scheme was adapted for robustness computations in robotics in [1], where the SUF were estimated for static configurations. However, despite an improvement over exact computation, due to the computational complexity of their formulation it was not previously possible to consider trajectory optimization of full system dynamics and maximization of robustness based on dynamic polytopes at the same time. We further adapt the technique of [12] to reformulate the problem of computing the SUF. The resulting reformulation allows to consider trajectory optimization and robustness maximization in a bilevel optimization setting.

Bilevel optimizations are mathematical programs that include the solution to other programs in their constraints or objectives. They are common in robustness settings, and have been used for robust control of robots. [13] optimized trajectories with full dynamics for robustness as a bilevel problem; it is particularly related to our work, but with some key differences: they considered robustness to noise and dealt with a fixed base manipulator—both differences allowed for simplifications in their optimization problem.

B. Statement of Contributions

We present a TO framework capable of planning robust and dynamic manipulation tasks for legged robots, such as the one shown in Fig. 1. Our main contributions are:

1) Proposal of a novel solution to a bilevel optimization problem that marries dynamic trajectory optimization with maximization of robustness against disturbances.
2) Explanation of the non-trivial transcription and reformulation required to make this problem tractable for a nonlinear programming (NLP) solver.
3) Comparison of our method's results against a traditional optimization objective across different scenarios.
4) Validation of the planned motions using both full-physics simulation and real-life hardware experiments.

We will also make our framework/implementation available upon acceptance/publication of this paper.

II. Trajectory Optimization

Trajectory Optimization (TO) is a well-known and powerful framework for planning locally-optimal trajectories of dynamic systems such as legged robots subject to constraints. TO falls under the broader category of optimal control problems. In general, TO aims to design a finite-time control trajectory as a function of time, $u(t)$, which drives the system from an initial state $x(t_I)$ towards a final state $x(t_F)$, and given the system dynamics $\dot{x} = f(x, u)$ which must be satisfied over the entire interval $t_I \leq t \leq t_F$. Optimal control problems can be solved using dynamic programming or by means of transcription (see [14]).

In this work, we employ a technique called direct transcription because it readily handles strict constraints on states and controls. Such constraints take a key role in computing the SUF. The main alternative, Differential Dynamic Programming (DDP), offers faster computation and provides a linear controller next to the optimized trajectory. However, handling constraints with DDP is a challenging subject of research [15], [16]. This currently makes DDP less applicable to our case. A second alternative, shooting methods, have been reported to result in slower computations and higher susceptibility to local optima [14]. Using direct transcription, we formulate the continuous optimization problem by explicitly discretizing the system state and control trajectories. This method results in the formulation of a large and sparse NLP problem [14]. The resulting constrained nonlinear optimization problem can then be solved using a sparse, large-scale nonlinear programming solver such as Knitro [17].

III. Model Formulation

The model of a legged robot can be formulated as a free-floating base B to which limbs are attached. For the specific case of the robot shown in Fig. 1, the kinematic tree stemming from the base branches into four legs and one robotic manipulator with six Degrees of Freedom (DoF). The motion of the system can be described with respect to (w.r.t.) a fixed inertial frame I. Let us express the position of the base w.r.t. the inertial frame, expressed in the inertial frame, as ${}_I\boldsymbol{r}_{IB} \in \mathbb{R}^3$. Let $\boldsymbol{q}_{IB} \in \mathbb{H}$ be a Hamiltonian unit quaternion defining the orientation of the free-floating base w.r.t. the inertial frame, and let $\boldsymbol{\psi}_{IB} \in \overline{\mathbb{R}}^3$ be the modified Rodrigues parameters (MRP) [18], [19] of the unit

quaternion $\boldsymbol{q}_{IB}$.[1] We use $\boldsymbol{\psi}_{IB}$ to parameterize the orientation of the free-floating base.[2] The joint angles describing the configuration of the 6-DoF arm and the four 3-DoF legs are stacked in a vector $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$, where $n_j = 18$. The generalized coordinates vector $\boldsymbol{q}$ and the generalized velocities vector $\boldsymbol{v}$ of this floating-base system may therefore be written as

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{\psi}_{IB} \\ \boldsymbol{q}_j \end{bmatrix} \in \mathbb{R}^3 \times \overline{\mathbb{R}}^3 \times \mathbb{R}^{n_j}, \quad \boldsymbol{v} = \begin{bmatrix} \boldsymbol{\nu}_B \\ \dot{\boldsymbol{q}}_j \end{bmatrix} \in \mathbb{R}^{n_v}, \quad (1)$$

where $n_v = 6 + n_j$ and the *twist* $\boldsymbol{\nu}_B = [{}_I\boldsymbol{v}_B \quad {}_B\boldsymbol{\omega}_{IB}] \in \mathbb{R}^6$ encodes the linear and angular velocities of the base B w.r.t. the inertial frame expressed in the I and B frames, respectively. The equations of motion of a floating base system that interacts with the environment are written as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{S}^\top \boldsymbol{\tau} + \boldsymbol{J}_s^\top(\boldsymbol{q})\boldsymbol{\lambda} + \boldsymbol{J}_e^\top(\boldsymbol{q})\boldsymbol{f}, \quad (2)$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_v \times n_v}$ is the mass matrix and $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^{n_v}$ is the vector of Coriolis, centrifugal, and gravity terms. The selection matrix $\boldsymbol{S} = [\boldsymbol{0}_{n_\tau \times (n_v - n_\tau)} \quad \mathbb{I}_{n_\tau \times n_\tau}]$ selects which DoF are actuated. Here, $n_\tau = n_j$ as all limb joints are actuated. The vector of ground-feet contact forces and contact torques $\boldsymbol{\lambda}$ is mapped to joint-space torques through the support Jacobian $J_s \in \mathbb{R}^{n_s \times n_v}$, which is obtained by stacking the Jacobians which relate generalized velocities to limb end-effector motion as $\boldsymbol{J}_s = [\boldsymbol{J}_{C_1}^\top \quad \cdots \quad \boldsymbol{J}_{C_{n_c}}^\top]^\top$, with n_c being the number of limbs in contact and n_s the total dimensionality of all contact wrenches. We assume ANYmal has point-feet and thus we only model linear contact forces at the feet. Finally, $\boldsymbol{f}$ represents any external force applied to the end-effector. This force may be the result of a push or some unpredicted disturbance. In a nominal scenario, this force is zero, i.e., $\boldsymbol{f} = \boldsymbol{0}$. The Jacobian $J_e \in \mathbb{R}^{3 \times n_v}$ is used to map a linear force $\boldsymbol{f}$ applied at the end-effector to joint-space torques.

IV. Problem Formulation

We transcribe the continuous optimization problem by explicitly discretizing the system state and the control trajectory using a *direct transcription* technique. We divide the trajectory into N equally spaced segments or intervals

$$t_I = t_1 < t_2 < \cdots < t_M = t_F, \quad (3)$$

where the points are referred to as *mesh points*.[3] The number of mesh points is given by $M = N + 1$. Henceforth, we use $x_k \equiv x(t_k)$ and $u_k \equiv u(t_k)$ to indicate the value of the state and control variables, respectively, at mesh point k. We treat the values of x_k and u_k as a set of NLP variables, and we finally formulate the general TO problem as:

$$\begin{aligned} \underset{\boldsymbol{\xi}}{\text{argmin}} \quad & g_M(x_M) + \sum_{k=1}^{M-1} g(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = x_k + h\, f(x_k, u_k) \\ & x_k \in \mathcal{X} \\ & u_k \in \mathcal{U} \end{aligned} \quad (4)$$

where $g(\cdot,\cdot)$ and $g_M(\cdot)$ form an optional cost function, $h = (t_F - t_I)/N$ is a fixed *integration step size*, and $\mathcal{X}$ and $\mathcal{U}$ are the sets of feasible states and inputs, respectively. We use the explicit Euler method to integrate the differential equations of the system dynamics, but other *K*-stage Runge-Kutta schemes could be used, e.g., the Trapezoidal method (implicit, $K = 2$) or the Hermite-Simpson method (implicit, $K = 3$).

A. Parameterization

Similarly to [20], we directly optimize over the space of feasible states, control inputs, and constraint forces, i.e., for each discretized mesh point k, the vectors of generalized coordinates $\boldsymbol{q}_k$, generalized velocities $\boldsymbol{v}_k$, control inputs $\boldsymbol{\tau}_k$, and contact forces $\boldsymbol{\lambda}_k$ form the vector of decision variables $\boldsymbol{\xi}_k$. The entire vector of NLP decision variables is:

$$\boldsymbol{\xi} \triangleq \{\boldsymbol{q}_1, \boldsymbol{v}_1, \boldsymbol{\tau}_1, \boldsymbol{\lambda}_1, \cdots, \boldsymbol{q}_N, \boldsymbol{v}_N, \boldsymbol{\tau}_N, \boldsymbol{\lambda}_N, \boldsymbol{q}_M, \boldsymbol{v}_M\}.^4 \quad (5)$$

Methods that treat contact forces as optimization variables are referred to as planning "through contact". This approach increases the number of decision variables, but the problem becomes better conditioned [21].

B. Objectives

We consider three different optimization objectives $\mathcal{G}_1$–$\mathcal{G}_3$. The first objective corresponds to the *feasibility problem*, i.e., a problem with constraints but without any cost to minimize.

The second objective $\mathcal{G}_2$ achieves the minimization of actuator torques and is defined as

$$\mathcal{G}_2: \quad \underset{\boldsymbol{\xi}}{\text{argmin}} \quad \sum_{k=1}^{M-1} \boldsymbol{\tau}_k^\top \boldsymbol{\tau}_k. \quad (6)$$

Finally, objective $\mathcal{G}_3$ corresponds to the maximization of the SUF at the end-effector. $\mathcal{G}_3$ involves a problem reformulation which is explained in Section V-A—the main contribution presented in this manuscript.

C. Constraints

We now analyze the constraints formulated in the NLP in detail. TABLE I shows a summary of these constraints.

[1] $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ is the *affinely extended set of real numbers*. We use the same notation as [19].

[2] The MRP encode a 3D rotation with the stereographic projection of a Hamiltonian unit quaternion. The derivatives of the rotation matrix w.r.t. the MRP parameters are rational functions, making this representation a particularly good choice for purposes of differentiation and optimization.

[3] Some authors also refer to these mesh points as *nodes*, *knots*, *way points*, or *grid points*.

[4] Notice that $\boldsymbol{\tau}_M$ and $\boldsymbol{\lambda}_M$ (i.e., the control and contact forces at the final state) are not required, and thus not part of $\boldsymbol{\xi}$.

1) Bounds on Decision Variables: We constrain the joint positions, velocities, and torques to be within their respective lower and upper bounds with (7)–(9).

$$\boldsymbol{q}_L \leq \boldsymbol{q}_k \leq \boldsymbol{q}_U \qquad \forall k = 1 : M \tag{7}$$
$$\boldsymbol{v}_L \leq \boldsymbol{v}_k \leq \boldsymbol{v}_U \qquad \forall k = 2 : M-1 \tag{8}$$
$$\boldsymbol{\tau}_L \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_U \qquad \forall k = 1 : M-1 \tag{9}$$

We further fix the initial and final velocities to zero:

$$\boldsymbol{v}_1 = \boldsymbol{v}_M = \boldsymbol{0}. \tag{10}$$

2) Friction Cones: Similarly to [7], we model friction at the contact points using an *inner linear approximation* with a four-sided friction pyramid. Consider the set of points $\{C_i\}$ where the robot is in contact with its environment. Let $\boldsymbol{n}_i$ and μ_i be the unit normal and the friction coefficient of the support region at each contact, respectively. A point contact remains fixed as long as its contact force $\boldsymbol{f}_i^c$ lies inside the linearized friction cone directed by $\boldsymbol{n}_i$:

$$|\boldsymbol{f}_i^c \cdot \boldsymbol{t}_i| \leq (\mu_i/\sqrt{2})(\boldsymbol{f}_i^c \cdot \boldsymbol{n}_i), \tag{11}$$
$$|\boldsymbol{f}_i^c \cdot \boldsymbol{b}_i| \leq (\mu_i/\sqrt{2})(\boldsymbol{f}_i^c \cdot \boldsymbol{n}_i), \tag{12}$$
$$\boldsymbol{f}_i^c \cdot \boldsymbol{n}_i > 0, \tag{13}$$

where $(\boldsymbol{t}_i, \boldsymbol{b}_i)$ form the basis of the tangential contact plane such that $(\boldsymbol{t}_i, \boldsymbol{b}_i, \boldsymbol{n}_i)$ is a direct frame.

3) System Dynamics: Using explicit Euler integration, we enforce the nonlinear system dynamics (f) with a finite set of *defect* (or *gap*) constraints in our NLP formulation:

$$\begin{bmatrix}\boldsymbol{q}_{k+1}\\ \boldsymbol{v}_{k+1}\end{bmatrix} - \begin{bmatrix}\boldsymbol{q}_k\\ \boldsymbol{v}_k\end{bmatrix} - h\, f\left(\begin{bmatrix}\boldsymbol{q}_k\\ \boldsymbol{v}_k\end{bmatrix}, \begin{bmatrix}\boldsymbol{\tau}_k\\ \boldsymbol{\lambda}_k\end{bmatrix}\right) = \boldsymbol{0}. \tag{14}$$

4) Stationary Feet: Let the *forward kinematics* function for a foot-point contact i be given by $f^{\text{fk}}(\boldsymbol{q}, i)$.

$$f^{\text{fk}}(\boldsymbol{q}_k, i) = \boldsymbol{p}_i \qquad \forall i = 1:4,\ k = 1 : M \tag{15}$$

5) Gripper Task: The gripper is constrained at the initial and final instants of the trajectory ($k = 1$ and $k = M$). For both of these mesh points, there exist five constraints: three to constrain the placement of the end-effector, and two for constraining the *pitch* and *roll* describing the orientation of the end-effector. This enforces a specific location for the pick and placing of a bottle (e.g., see Fig. 2), as well as the correct orientation of the fingers to embrace it, while leaving the grasp *yaw* as a degree of freedom to the solver.

TABLE I: Summary of the formulated NLP constraints.

Constraint	Structure	Relation
Bounds on $\boldsymbol{\xi}$	Linear	Mixed
Friction Cones	Linear	Inequality
System Dynamics	Nonlinear	Equality
Stationary Feet	Nonlinear	Equality
Gripper Task	Nonlinear	Equality

Building on the formulation above, we now want to implement additional terms to model the external disturbances. We are interested in maximizing the forces applied at the end-effector that the robot can compensate for while still satisfying all the NLP constraints from TABLE I.

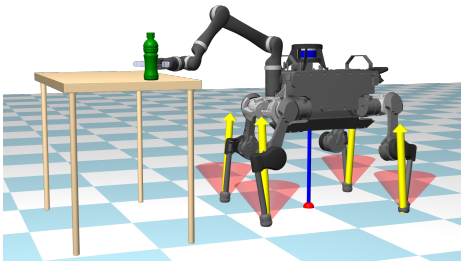


Fig. 2: The figure shows the robot at the beginning of a pick-and-place task. The ground-feet contact forces are shown in yellow and the friction cones are shown in red.

V. Robustness to Disturbances

External forces applied to the robot can cause the robot to slip, lose contact between a foot and the environment, or to fail to track the desired end-effector path. In each of these cases, the motion fails because the external force causes a violation of one of the motion constrains. We therefore define robustness as some metric of distance to the constraints. More specifically, we consider the friction cone constraints on the contact forces and the actuator torque bounds limiting the control commands that can be used to compensate for external forces. As pointed out by [8], when transformed into a common reference frame, these constraints form a convex polytope bounding the volume of all admissible external wrenches applied to the robot. In [9] we proposed a geometric method to inscribe a ball into the polytope and use its radius as a metric of robustness. This approximation is especially useful since the radius of the maximum volume inscribed ball gives a bound for the magnitude of forces from any direction that the system can compensate for without violating the constraints.

A. Maximum Volume Inscribed Ball of a Polytopic Projection

In order to reject a disturbance force, additional motor torques and ground reaction forces are needed. Our robustness metric, the SUF, is the smallest force for which no reaction forces/torques exist that also satisfy friction-cone constraints and motor limitations. In previous work [9], the SUF was computed via a Linear Programming (LP) problem. The trajectory optimization would have this LP problem inside its objective, which is not desirable. Hence, we propose a new way to compute the SUF. The key idea in this robustness analysis is to approximate the relationship between the disturbance force and reaction forces/torques as affine. Adapting the results from [12], we find a practically efficient way to simultaneously optimize the robustness metric and the affine relationship prescribing it. These results go beyond earlier adaptations in robotics by [1] because those, like our work relying on LP, were not suitable for use in a trajectory optimization setting.[5]

[5]This is due to the fact that computing derivatives of an LP would require a differentiable solver. Solving an LP inside an optimization problem can also lead to higher computational time.

Let us define the *extended* torques and ground-feet contact forces as $\boldsymbol{\tau}^+$ and $\boldsymbol{\lambda}^+$, respectively:

$$\boldsymbol{\tau}^+ = \boldsymbol{\tau} + \boldsymbol{K}_{\boldsymbol{\tau}} \boldsymbol{f} \tag{16}$$

$$\boldsymbol{\lambda}^+ = \boldsymbol{\lambda} + \boldsymbol{K}_{\boldsymbol{\lambda}} \boldsymbol{f}, \tag{17}$$

where $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$ are the *nominal* torques and ground-feet contact forces, $\boldsymbol{K}_{\boldsymbol{\tau}}$ and $\boldsymbol{K}_{\boldsymbol{\lambda}}$ are some (instantaneous) gain matrices which map a force expressed in end-effector space to joint-torque space and ground-feet contact space, respectively, and $\boldsymbol{f}$ is a potential external force applied at the end-effector. In a nominal situation, there are no disturbance forces and thus $\boldsymbol{f} = \boldsymbol{0}$, $\boldsymbol{\tau}^+ = \boldsymbol{\tau}$, and $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}$. Assuming no variation in accelerations, replacing $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$ in the equations of motion (2) with the right-hand side of (16) and (17) gives:

$$\boldsymbol{0} = \left(\boldsymbol{S}^\top \boldsymbol{K}_{\boldsymbol{\tau}} + \boldsymbol{J}_s^\top \boldsymbol{K}_{\boldsymbol{\lambda}} + \boldsymbol{J}_e^\top\right) \boldsymbol{f} \tag{18}$$

Alternatively to constraints (9) and (11)–(12), we can represent the actuator torque bounds and friction cones constraints using $\boldsymbol{\tau}^+$ and $\boldsymbol{\lambda}^+$ as:

$$\boldsymbol{A}_{\boldsymbol{\tau}} \boldsymbol{\tau}^+ \leq \boldsymbol{b}_{\boldsymbol{\tau}} \tag{19}$$

$$\boldsymbol{A}_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^+ \leq \boldsymbol{b}_{\boldsymbol{\lambda}}. \tag{20}$$

We then substitute (16)–(17) into (19)–(20) and for each row $\boldsymbol{a}_{\boldsymbol{\tau}}$ of matrix $\boldsymbol{A}_{\boldsymbol{\tau}}$ we write the constraint as:

$$\boldsymbol{a}_{\boldsymbol{\tau}}^\top \left(\boldsymbol{\tau} + \boldsymbol{K}_{\boldsymbol{\tau}} \boldsymbol{f}\right) \leq \boldsymbol{b}_{\boldsymbol{\tau}} \qquad \forall \, |\boldsymbol{f}| \leq \rho, \tag{21}$$

where ρ is the radius of the maximum volume inscribed ball of a polytopic projection, and it represents the magnitude of the smallest potential disturbance that cannot be directly rejected. We then define $\boldsymbol{f} = \rho \boldsymbol{\chi}$, where $\boldsymbol{\chi} \in \mathbb{R}^3$ is a vector with unit length, which allows us to find the greatest ρ with:

$$\left(\max_{\boldsymbol{\chi}} \quad \boldsymbol{a}_{\boldsymbol{\tau}}^\top \left(\boldsymbol{\tau} + \boldsymbol{K}_{\boldsymbol{\tau}} \rho \boldsymbol{\chi}\right)\right) \leq \boldsymbol{b}_{\boldsymbol{\tau}}. \tag{22}$$

The objective function of the left-hand side of equation (22) can be seen as a scalar product of the vectors $\boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{K}_{\boldsymbol{\tau}} \rho$ and $\boldsymbol{\chi}$, which is greatest when these vectors are collinear:

$$\operatorname*{argmax}_{\boldsymbol{\chi}} \quad \boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{K}_{\boldsymbol{\tau}} \rho \boldsymbol{\chi} \quad \equiv \quad \frac{\boldsymbol{K}_{\boldsymbol{\tau}}^\top \boldsymbol{a}_{\boldsymbol{\tau}}}{\|\boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{K}_{\boldsymbol{\tau}}\|}. \tag{23}$$

Simplifying (22) with the right-hand side of (23) leads to:

$$\boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{\tau} + \left\|\boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{K}_{\boldsymbol{\tau}}\right\| \rho \leq \boldsymbol{b}_{\boldsymbol{\tau}}. \tag{24}$$

Equations (21)–(24) address the constraints on actuation limits. We repeat the same process for the ground-feet contact forces to obtain:

$$\boldsymbol{a}_{\boldsymbol{\lambda}}^\top \boldsymbol{\lambda} + \left\|\boldsymbol{a}_{\boldsymbol{\lambda}}^\top \boldsymbol{K}_{\boldsymbol{\lambda}}\right\| \rho \leq \boldsymbol{b}_{\boldsymbol{\lambda}}. \tag{25}$$

Next, we substitute $\overline{\boldsymbol{K}}_{\boldsymbol{\tau}} = \boldsymbol{K}_{\boldsymbol{\tau}} \rho$ and $\overline{\boldsymbol{K}}_{\boldsymbol{\lambda}} = \boldsymbol{K}_{\boldsymbol{\lambda}} \rho$ into (18), (24) and (25) and write:

$$\boldsymbol{S}^\top \overline{\boldsymbol{K}}_{\boldsymbol{\tau}} + \boldsymbol{J}_s^\top \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}} + \boldsymbol{J}_e^\top \rho = \boldsymbol{0}, \tag{26}$$

$$\boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{\tau} + \left\|\boldsymbol{a}_{\boldsymbol{\tau}}^\top \overline{\boldsymbol{K}}_{\boldsymbol{\tau}}\right\| \leq \boldsymbol{b}_{\boldsymbol{\tau}}, \tag{27}$$

$$\boldsymbol{a}_{\boldsymbol{\lambda}}^\top \boldsymbol{\lambda} + \left\|\boldsymbol{a}_{\boldsymbol{\lambda}}^\top \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}}\right\| \leq \boldsymbol{b}_{\boldsymbol{\lambda}}. \tag{28}$$

This substitution removes the bilinear products between $\boldsymbol{K}_{\boldsymbol{\tau}}$, $\boldsymbol{K}_{\boldsymbol{\lambda}}$ and ρ while keeping the equality and inequalities valid.

B. Constraints' Structure Exploitation

We now extend the problem formulation and transcribe the constraints (26)–(28) directly into NLP constraints and we extend the vector of decision variables with $\overline{\boldsymbol{K}}_{\boldsymbol{\tau}}$, $\overline{\boldsymbol{K}}_{\boldsymbol{\lambda}}$, and ρ. However, by trying this, one will soon realize we face an NP-hard problem. Additionally, there is a significant increase in the amount of decision variables, and the dependency of both $\boldsymbol{J}_s$ and $\boldsymbol{J}_e$ on joint positions means that constraint (26) is nonlinear and non-convex. Ultimately, this quickly renders any efforts of a naïve transcription ineffective, as the solver would be unable to digest it.

In order to solve this issue, we have to analyze the inherent structure of the problem and its constraints. Let $\overline{\boldsymbol{K}}_{\boldsymbol{\tau}}$ be the unknown in constraint (26). Splitting the structure of the constraint as

$$\begin{bmatrix} \boldsymbol{0} \\ \mathbb{I} \end{bmatrix} \overline{\boldsymbol{K}}_{\boldsymbol{\tau}} = - \begin{bmatrix} \boldsymbol{J}_s^{\top_{\text{base}}} \\ \boldsymbol{J}_s^{\top_{\text{limbs}}} \end{bmatrix} \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}} - \begin{bmatrix} \boldsymbol{J}_e^{\top_{\text{base}}} \\ \boldsymbol{J}_e^{\top_{\text{limbs}}} \end{bmatrix} \rho \tag{29}$$

highlights that $\overline{\boldsymbol{K}}_{\boldsymbol{\tau}}$ can be obtained as a function of $\overline{\boldsymbol{K}}_{\boldsymbol{\lambda}}$ and ρ without performing any inversions. Doing this satisfies the bottom equality implicitly. The top part of the equality affecting the floating base still needs to be enforced.

This key-insight into the structure of the constraints allows us to transcribe the problem so that the solver will converge successfully.

C. NLP Reformulation

1) Parameterization: We extend the previous definition of $\boldsymbol{\xi}$ to accommodate for the extra decision variables required. Recall that $\overline{\boldsymbol{K}}_{\boldsymbol{\tau} k}$ need not be discretized.

$$\boldsymbol{\xi}^+ \triangleq \boldsymbol{\xi} \cup \{\rho_1, \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}1}, \cdots, \rho_N, \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}N}\}. \tag{30}$$

2) Objective: $\mathcal{G}_3$ is the sum of all the ρ_k in $\boldsymbol{\xi}^+$:

$$\mathcal{G}_3: \quad \operatorname*{argmax}_{\boldsymbol{\xi}^+} \quad \sum_{k=1}^{M-1} \rho_k \tag{31}$$

3) Constraints: We bound all the ρ_k in $\boldsymbol{\xi}^+$ to $\mathbb{R}^+$ with a linear one-sided inequality:

$$\rho_k \geq 0 \quad \forall k = 1 : M-1. \tag{32}$$

We enforce the top part of constraint (29) explicitly:

$$\boldsymbol{J}_s^{\top_{\text{base}}} \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}} + \boldsymbol{J}_e^{\top_{\text{base}}} \rho = \boldsymbol{0} \tag{33}$$

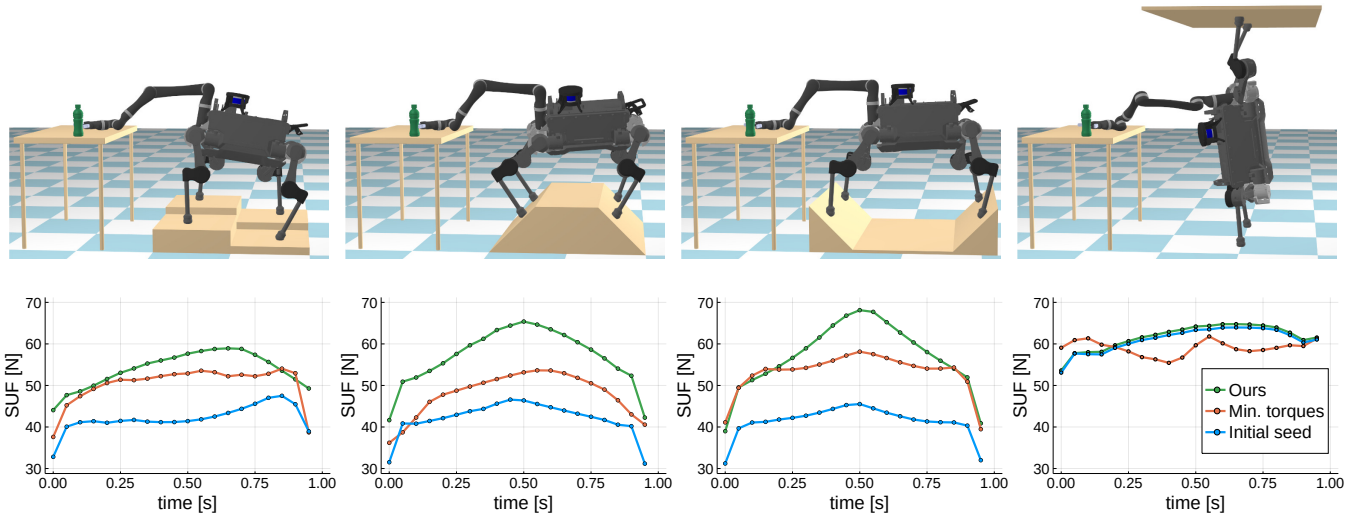
Finally, (27) is rewritten as:

$$\boldsymbol{a}_{\boldsymbol{\tau}}^\top \boldsymbol{\tau} + \left\|\boldsymbol{a}_{\boldsymbol{\tau}}^\top \left(-\boldsymbol{J}_s^{\top_{\text{limbs}}} \overline{\boldsymbol{K}}_{\boldsymbol{\lambda}} - \boldsymbol{J}_e^{\top_{\text{limbs}}} \rho\right)\right\| \leq \boldsymbol{b}_{\boldsymbol{\tau}}, \tag{34}$$

A summary of the constraints added to the NLP with the reformulation is shown in TABLE II.

TABLE II: Summary of the reformulated NLP constraints.

Constraint	Structure	Relation
Bounds on ρ	Linear	Inequality
Equation (33)	Nonlinear	Equality
Equation (34)	Nonlinear	Inequality
Equation (28)	Conic	Inequality



(a) "Steps" terrain. (b) "Ramp" terrain. (c) "Valley" terrain. (d) "Handstand" scenario.

Fig. 3: We set up a multitude of terrains for testing a pick-and-place task: flat ground, slabs at different heights (a), and inclined supports (b)–(c). An extreme scenario where the robot performs a "handstand" is shown in (d). The plots underneath each scene show the SUF (in newtons) for the trajectories computed using different optimization objectives.

VI. Performance Evaluation

In order to evaluate our work, we compared the robustness of the three objective functions proposed in Section IV-B: feasibility ($\mathcal{G}_1$), minimum torques ($\mathcal{G}_2$), and maximum SUF ($\mathcal{G}_3$). We ran the comparison across different scenarios for a pick-and-place task. Furthermore, we benchmarked the times taken to evaluate all NLP constraints and the convergence times for problems of different sizes.

Fig. 3 shows four different settings for a pick-and-place task of a bottle on a table. We set up scenarios with challenging terrain, where the robot stands on steps with different heights or inclined slabs. The trajectories optimized with our method ($\mathcal{G}_3$) demonstrated greater robustness, as shown in plots (a)–(c) in Fig. 3. The initial guess used for $\mathcal{G}_2$ and $\mathcal{G}_3$ was the result of the feasibility problem $\mathcal{G}_1$.

We also verified the robustness of trajectories for different inclines. For this, we varied the grade of the slopes in the "ramp" scenario (Fig. 3b) from $0°$ to $60°$. The trajectories computed with our metric consistently showed a greater SUF for all the tested slopes (see Fig. 4).

As an extreme example, we created a scenario where the robot has to perform a "handstand", i.e., support its own weight on two of its legs (see Fig. 3d) while using the remaining two for keeping its balance. In this scenario, it is

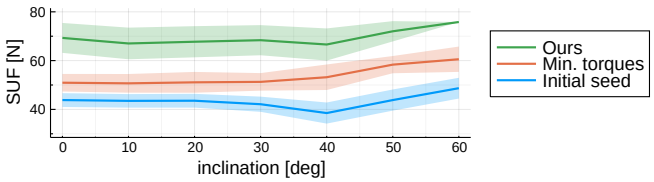


Fig. 4: Mean and standard deviation of the SUF at the end-effector for varying inclinations on the "ramp" scenario.

TABLE III: Times taken to evaluate the NLP constraints.

Constraint	Function (µs)	Jacobian (µs)
Gripper Task	9.15 ± 25.18	14.93 ± 2.47
Stationary Feet	18.29 ± 2.07	57.67 ± 225.53
System Dynamics	55.07 ± 169.64	3801.85 ± 1353.08
SUF Constraints	73.76 ± 239.97	1396.90 ± 989.31

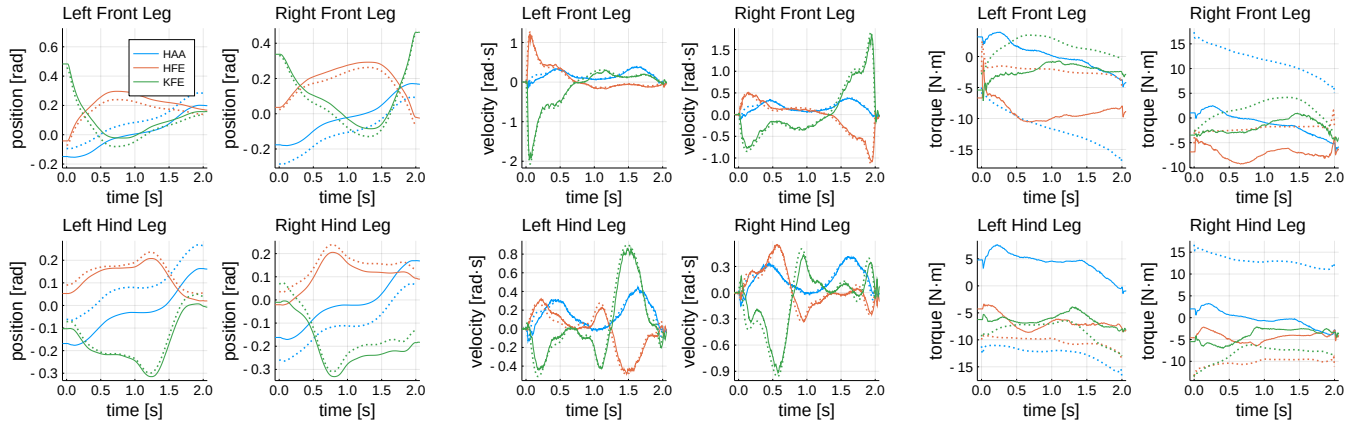
TABLE IV: Convergence times of objectives $\mathcal{G}_1$–$\mathcal{G}_3$ for problems with different size: 11, 21, and 41 mesh points.

M	$\mathcal{G}_1$ (s)	$\mathcal{G}_2$ (s)	$\mathcal{G}_3$ (s)
11	0.46 ± 0.007	115.45 ± 0.27	229.34 ± 0.39
21	0.74 ± 0.009	143.48 ± 5.56	608.09 ± 8.04
41	1.21 ± 0.005	835.81 ± 15.59	1775.23 ± 12.85

especially important for the robot to press downwards against the floor and upwards against the ceiling to maintain stability. Because of this, torque minimization ($\mathcal{G}_2$) is not an appropriate objective for this scenario, as confirmed by the degraded SUF when compared to the initial seed in plot (d). On the other hand, our method is able to increase the robustness of the initial seed by a small amount, because it allows to trade off torque expenditure for more stable ground/ceiling-feet contact forces. We would like to emphasize that the motions in all of the scenarios are within the actual physical capabilities of the robot, even the "handstand" scenario.

TABLE III shows the computation times for function and Jacobian evaluations of the NLP problem constraints. The longest time is spent computing the Jacobian of the system dynamics. Evaluating the Jacobian of the SUF—which is involved when optimizing $\mathcal{G}_3$—takes the second longest time.

TABLE IV shows the total time it takes for the solver to converge for problems of different size. We benchmarked problems with 11, 21, and 41 mesh points (for a 1-second



(a) Position (b) Velocity (c) Effort

Fig. 5: Joint positions, velocities, and torques of ANYmal for a 2-seconds long trajectory on flat ground. The dotted lines correspond to the planned trajectory. The solid lines show the data collected by the state estimation on the real robot.

trajectory, this is the equivalent of a discretization at 10, 20, and 40 Hz). Each average and standard deviation are taken from five samples. $\mathcal{G}_1$ is the fastest to solve as it is a feasibility problem and does not consider any optimality function. In our tests, the overall best robustness of $\mathcal{G}_3$ (shown in Fig. 3) also comes with the trade-off of the longest times required until convergence.

All evaluations in this section were carried out in a single-threaded process on an Intel i7-6700K CPU with 4.0 GHz and 32 GB 2133 MHz memory. The proposed optimization framework has been implemented using Julia [22] and the optimization library Knitro [17]. The chosen solving algorithm was the interior-point method[6] presented by [23].

VII. Experiments

We conducted hardware experiments on an ANYmal [2] quadruped equipped with a Kinova Jaco [3] robot arm. The motion planning is performed a priori and the optimized trajectories are then sent to the controller for playback.

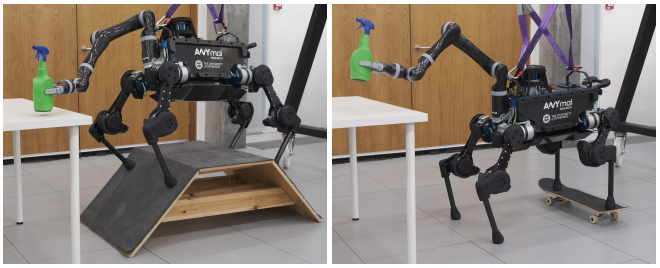
A. Robot Control

To execute our whole-body motions, we tracked the joint position with feed-forward velocity and torque. We updated the references for joint position, joint velocity, and joint torque at 400 Hz. The decentralized motor controller at every joint closes the loop compensating for friction effects. During our experiments, we used $k_p = [150, 150, 100]$ as proportional and $k_d = [0.5, 0.5, 0.45]$ as derivative joint space gains for each leg, respectively. The arm used Kinova's driver for joint trajectory control. We synchronized the execution of both controllers.

In order to evaluate how well the real robot tracks motions using our controller, we compared the planned joint states over time with the state estimation data from the robot. We computed a 2-seconds long trajectory using our framework for a pick-and-place task sampled at 400 Hz and commanded the robot at the same frequency. Fig. 5 shows the plots of the planned trajectory against the data collected during our experiments. The plots show that joint positions are within acceptable tolerances and joint velocities are tracked well, but joint efforts are significantly different. This validates that the motions generated by our trajectory optimization are dynamically consistent. The mismatch in joint efforts is expected due to differences between the real robot and our model, and also due to signal delays. Additionally, as we executed our trajectory open-loop without re-planning, the errors accumulated. Nonetheless, the tracking controller can execute the dynamic motions we planned.

B. Description of the Experiments

We executed the pick-and-place of a bottle on a table for different terrain: on flat ground and on a "ramp" (Fig. 6a). The object being grasped was not modeled and it is therefore an external disturbance. We also tested the trajectories for ground-feet friction coefficient mismatches by placing a skateboard underneath the feet of the robot (Fig. 6b). A video is available at `https://youtu.be/vDesP7IpThw`.



(a) "Ramp" terrain (b) "Skateboard" scenario

Fig. 6: Snapshots of the real robot executing the planned motions on a ramp (see Fig. 3b) and on a skateboard.

For the motions shown in the video, we optimized the trajectories at 100 Hz and then linearly interpolated them to

[6]Interior-point methods (also known as barrier methods) replace the NLP problem by a series of barrier subproblems controlled by a barrier parameter. They are generally preferable for large-scale problems.

400 Hz. It was the interpolation result that was then tracked by the controller. We did this because computing optimal trajectories with $\mathcal{G}_3$ gets more computationally expensive as the problem discretization increases (see TABLE IV).

To select the frequency of the trajectory before interpolation, we computed the root-mean-square error (RMSE) of the SUF over time for the same trajectory using different discretization resolutions, with a 400 Hz resolution as a baseline. As shown in Fig. 7, for a trajectory discretized at 100 Hz its SUF RMSE ≈ 0.5 N, which is acceptable for our purposes.

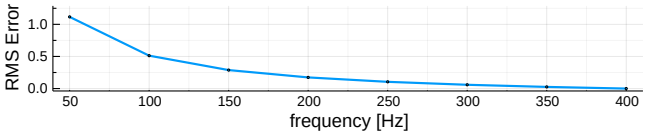


Fig. 7: Root-mean-square error (RMSE) in newtons of the SUF given different discretizations, for a 400 Hz baseline.

VIII. Future Work

Robustness Through Environment Exploitation: This work has focused on isotropic robustness, but for some tasks it may be more appropriate to be able to resist disturbances along particular directions, e.g., as in Fig. 8. Based on our work [9], it should be straightforward to adapt Section V-A to maximize either the volume of an ellipsoid or the magnitude of a specific vector inscribed in the projection of the dynamic force polytope.

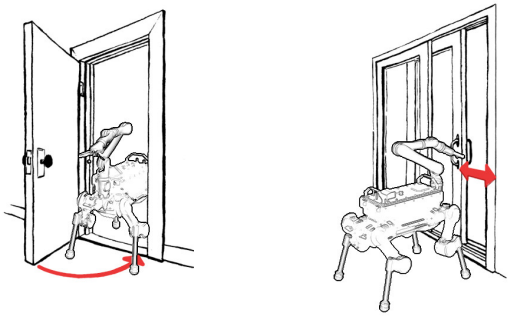


Fig. 8: *Left:* Door rotating over its hinges as a pivot point. *Right:* Door sliding back and forth on a frame.

Robust Dynamic Loco-manipulation: In this work, we did not study scenarios involving the making or breaking of support contacts with the environment. It would be interesting to adapt the trajectory optimization transcription to allow for multi-contact tasks or for dynamic loco-manipulation.

Minimization of Robustness Loss: Our framework maximizes the robustness of a trajectory assuming reliable execution. However, the system dynamics around a nominal trajectory are not linear, and given large enough perturbations the robot may be driven into areas of low robustness. This can be alleviated by accounting for neighboring states and taking the control noise into account.

Acknowledgments

We would like to thank Twan Koolen, João Moura, Romeo Orsolino, François Pacaud, Theodoros Stouraitis, and Matt Timmons-Brown for their valuable help and feedback. We would also like to thank the anonymous reviewers for their constructive comments.

References

[1] W. J. Wolfslag, C. McGreavy *et al.*, "Optimisation of Body-ground Contact for Augmenting Whole-Body Loco-manipulation of Quadruped Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[2] M. Hutter, C. Gehring *et al.*, "ANYmal - toward legged robots for harsh environments," *Advanced Robotics*, 2017.

[3] A. Campeau-Lecours, H. Lamontagne *et al.*, "Kinova modular robot arms for service robotics applications," in *Rapid Automation: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2019.

[4] C. D. Bellicoso, K. Krämer *et al.*, "ALMA — Articulated Locomotion and Manipulation for a Torque-Controllable Robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[5] A. D. Prete and N. Mansard, "Robustness to Joint-Torque-Tracking Errors in Task-Space Inverse Dynamics," *IEEE T-RO*, 2016.

[6] G. Xin, H. Lin *et al.*, "A Model-Based Hierarchical Controller for Legged Systems Subject to External Disturbances," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[7] S. Caron, Q. Pham, and Y. Nakamura, "Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics," in *Robotics: Science and System*, 2015.

[8] R. Orsolino, M. Focchi *et al.*, "Application of Wrench-Based Feasibility Analysis to the Online Trajectory Optimization of Legged Robots," *IEEE Robotics and Automation Letters*, 2018.

[9] H. Ferrolho, W. Merkt *et al.*, "Residual Force Polytope: Admissible Task-Space Forces of Dynamic Trajectories," *arXiv preprint arXiv:1908.05380*, 2019.

[10] T. Yoshikawa, "Manipulability of Robotic Mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[11] H. R. Tiwary, "On Computing the Shadows and Slices of Polytopes," *arXiv e-prints*, p. arXiv:0804.4150, 2008.

[12] J. Zhen and D. den Hertog, "Computing the Maximum Volume Inscribed Ellipsoid of a Polytopic Projection," *INFORMS Journal on Computing*, vol. 30, no. 1, pp. 31–42, 2018.

[13] B. Landry, Z. Manchester, and M. Pavone, "A Differentiable Augmented Lagrangian Method for Bilevel Nonlinear Optimization," in *Robotics: Science and System*, 2019.

[14] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. Society for Industrial and Applied Mathematics, 2010.

[15] M. Giftthaler, M. Neunert *et al.*, "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[16] C. Mastalli, R. Budhiraja *et al.*, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[17] R. H. Byrd, J. Nocedal, and R. A. Waltz, *Knitro: An Integrated Package for Nonlinear Optimization*. Springer US, 2006.

[18] P. G. Gormley, "Stereographic Projection and the Linear Fractional Group of Transformations of Quaternions," *Proceedings of the Royal Irish Academy. Mathematical and Physical Sciences*, vol. 51, 1945.

[19] G. Terzakis, M. Lourakis, and D. Ait-Boudaoud, "Modified Rodrigues Parameters: An Efficient Representation of Orientation in 3D Vision and Graphics," *Journal of Mathematical Imaging and Vision*, 2018.

[20] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[21] B. Landry, J. Lorenzetti *et al.*, "Bilevel Optimization for Planning through Contact: A Semidirect Method," in *International Symposium on Robotics Research (ISRR)*, 2019.

[22] J. Bezanson, A. Edelman *et al.*, "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, vol. 59, 2017.

[23] R. A. Waltz, J. L. Morales *et al.*, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Mathematical Programming*, vol. 107, no. 3, pp. 391–408, 2006.