Contact-Implicit Trajectory Optimization With Learned Deformable Contacts Using Bilevel Optimization

Yifan Zhu, Zherong Pan, and Kris Hauser

***Abstract*— We present a bilevel, contact-implicit trajectory optimization (TO) formulation that searches for robot trajectories with learned soft contact models. On the lower-level, contact forces are solved via a quadratic program (QP) with the maximum dissipation principle (MDP), based on which the dynamics constraints are formulated in the upper-level TO problem that uses direct transcription. Our method uses a contact model for granular media that is learned from physical experiments, but is general to any contact model that is stick-slip, convex, and smooth. We employ a primal interior-point method with a pre-specified duality gap to solve the lower-level problem, which provides robust gradient information to the upper-level problem. We evaluate our method by optimizing locomotion trajectories of a quadruped robot on various granular terrains offline, and show that we can obtain long-horizon walking gaits of high qualities.**

I. Introduction

Legged robot locomotion planners typically assume rigid contacts with Coulomb friction, but non-rigid terrains with different frictional characteristics are pervasive in the world, e.g. sand, mud, carpet, etc. To tackle these challenging soft terrains, existing techniques rely on feedback control while assuming rigid contacts [1], or modeling the terrain as a mass-spring damper [2, 3] or as vistoplastic [4] in the controller. While inaccurate contact models result in sub-optimal motions, our proposed trajectory optimization (TO) framework adopts realistic soft contact models learned from physical experiments on granular terrains.

This research is motivated by our prior work [5] that built a semi-empirical simulator for robots traversing granular terrains. Based on the stick-slip behavior of a rigid body contacting granular media, a granular contact is first modeled as all possible contact wrenches between a robot ankle and the granular terrain, which is a convex volume learned from data. Then during online simulation, contact forces are solved by a quadratic program (QP) with linear constraints during each simulation frame, based on the maximum dissipation principle (MDP) [6].

In this work, we incorporate this simulator into motion planning with granular contacts, where we take advantage of the accurate learned contact models to obtain high-quality trajectories. In particular, we employ TO, which is a powerful framework for generating locally optimal trajectories for robotic systems (for details on TO, refer to the survey by Betts [7]). Compared to sampling-based motion planning methods [8], which can generate globally optimal trajectories, TO methods scale better to high-dimensional robotic systems. Our contribution in this work is two-fold. First, we propose a bilevel TO framework that is capable of automatically searching for locomotion trajectories of legged robots traversing granular terrains. Second, we build a differentiable rigid body simulator with learned granular contact models. Each simulation frame consists of solving a QP, whose gradients are robustly obtained by employing a primal interior-point method with a pre-specified duality gap.

On the upper-level of the TO framework, we adopt a direct transcription formulation that is also contact-implicit. The dynamics constraints of the TO are obtained through the lower-level optimization, which is our differentiable physics simulator. We are then able to supply analytic gradients of the dynamics constraints to the optimization solver for efficiency and accuracy.

In this work, we focus on locomotion planning on granular terrains for a Robosimian quadruped developed by JPL, although it could be applied to general contact-rich motion planning, with a contact model that is convex and twice-differentiable. This turns out to be a very challenging problem, ill-posed and with potentially many local minima. However, we show that our method can automatically generate locomotion trajectories of long horizons offline on a variety of terrain shapes, where competitive methods such as reinforcement learning and standard trajectory optimization fail to generate trajectories on a simple flat granular terrain.

II. Related Work

While TO has been applied successfully to smooth dynamical systems, planning contact-rich trajectories, e.g. locomotion planning for legged robots, is still a challenging problem due to the underlying non-smooth, hybrid dynamical system. Successes have been achieved by adopting a pre-specified contact sequence [9, 10], however, manually designing contact sequences requires significant engineering and could be daunting for complex systems.

Contact-implicit TO: Recently, contact-implicit TO methods [11, 12, 13, 14, 15] are proposed to solve this problem by folding the contact dynamics into the formulation of the mathematical program, which avoids the need of a contact sequence input. Our method falls into this category as well, and we review the closely related works here. Posa *et al.* [11] propose a direct transcription method that searches locally optimal trajectories for robots with inelastic impacts and rigid contacts with Coulomb friction. Based on the formulation of multi-contact dynamics as a Linear Complementarity Problem (LCP) [16] for dynamics simulation, the proposed algorithm introduces contact forces

*This work was supported in part by NSF grants NRI-#1911087 and 2025782. Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA. {yifan16, zherong, kkhauser}@illinois.edu

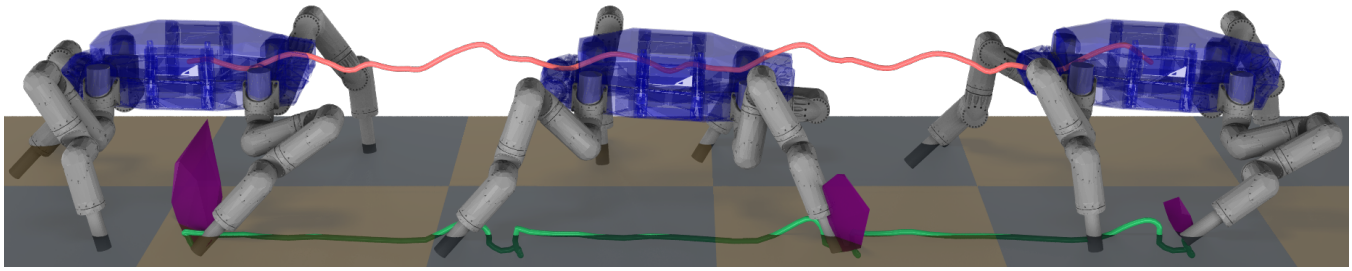

Fig. 1. An optimized trajectory on a flat terrain with sand (tiles are 1m in length). The centroid of the robot torso over the 9s trajectory is traced in red, and the front right foot position in green. The contact wrench space of the front right foot is in purple. [Best viewed in color.]

as additional optimization variables and incorporates complementarity constraints (CC) in the TO formulation. The resulting algorithm solves a mathematical program with CC and is demonstrated to generate locally optimized trajectories for high-dimensional systems. One drawback of this method is the high non-linearity introduced by the CC, which leads to poor convergence performance. To avoid having CC, Carius *et al.* [14] propose a bilevel TO framework, where the lower-level solves for the robot dynamics with rigid contacts using a time-stepping scheme of rigid body dynamics forward simulation, whose gradients can be obtained analytically. The upper-level TO uses iLQR [17], a shooting type method, and is shown to be able to compute a trajectory with 300 time steps for a 3 degree-of-freedom (DoF) robot in 2.18 ms. The idea of our method is closely related to this work. Instead of folding forward simulation with rigid contacts into system dynamics constraints, we use granular contacts. In addition, we use a direct TO method which is numerically advantageous for long trajectories. Another method that aims to remove CC is proposed by Landry *et al.* [13]. They also put forward a bilevel TO framework, where the upper-level is a direct transcription method and the lower-level solves a QP to obtain the tangential contact forces at the rigid contacts. Our method is most closely related to this work. While Landry *et al.* [13] only include part of the system dynamics in the lower-level optimization (only tangential contact forces), our method solves all the system dynamics implicitly in the lower-level. Our method considers a general contact model and we also adopt a different strategy to choose a gradient for the lower-level problem when it is not uniquely defined. Landry *et al.* [13] use a first-order method to solve the QP where the gradient of the solution could be non-unique and they resort to choosing a subgradient with least-squares, which does not have a strong theoretical foundation. However, we use an interior point method with a pre-specified duality gap where the gradient is always uniquely defined.

Bilevel Optimization: Our method falls into the general category of bilevel optimization [18], which is an optimization problem (upper-level) whose constraints involve other optimization problems (lower-level). In addition to the two works that are mentioned above, bilevel optimization has been applied to TO in a variety of works [19, 20]. Farshidian *et al.* [19] propose a bilevel optimization framework to plan motions for legged robots, where the upper-level optimizes time allocation for a fixed contact sequence, and the lower-level computes optimal continuous control inputs. Tang *et al.* [20] propose a method to generate time-optimal trajectories for smooth dynamical systems, which optimizes the trajectory shape on the upper-level and the time allocation on the lower-level. While these two methods decompose a challenging optimization via bilevel optimization, our method encodes system dynamics naturally into lower-level optimization.

Locomotion on Soft Terrain: Instead of computing motion plans on soft terrains, previous methods design controllers that compensate for soft contacts. Kang *et al.* [2] model a deformable terrain with compressive springs and stablize the CoM motion of a humanoid by tilting its ankle according to the attitude angle of the robot. Vasilopoulos *et al.* [4] calculate controls of a 1-DoF leg to maintain a desired height while jumping on a soft terrain based on a viscoplastic model. Fahmi *et al.* [3] model a soft terrain as a compressive spring, run a online controller to estimate the compliance, and incorporate the compliance explicitly into a whole-body-control framework for a quadruped robot. Compared to building controllers, our method plans trajectories with long horizons, allowing for more diverse and optimal motion plans.

III. Method

A. Problem statement

We search for a discretized trajectory for a legged robot, represented by the states $\boldsymbol{x}[\cdot] \in \mathbb{R}^{n_x}$ and controls $\boldsymbol{u}[\cdot] \in \mathbb{R}^{n_u}$. Let a robot state be the concatenation of robot configuration and velocity, i.e. $x = [q, \dot{q}]$ and $\boldsymbol{x}[k] = [q_k, \dot{q}_k]$, the dynamics of the robot is represented by the Newton-Euler equation:

$$M(q)\ddot{q} + H(q, \dot{q}) = u + J_c^T f, \tag{1}$$

where u is the joint torque, M is the generalized mass matrix, H is the centrifugal and Coriolis forces, J_c is the contact jacobian, and f is contact wrench. Using an Euler integration scheme, the robot dynamics along the discretized trajectory $\boldsymbol{x}[\cdot]$ and $\boldsymbol{u}[\cdot]$ is:

$$\begin{aligned} \dot{q}_{k+1} &= M(q_k)^{-1}(u_k + J_c^T f - H(q_k, \dot{q}_k)) \cdot dt + \dot{q}_k, \forall k \\ q_{k+1} &= \dot{q}_k \cdot dt + q_k, \forall k, \end{aligned} \tag{2}$$

where dt is the time step. Formulating this equation requires solving the contact wrench f, which is computed by a differentiable semi-empirical simulator for robots traversing

granular terrains. Details of the simulator are described in the next section.

B. Robot dynamics with granular contacts

1) Contact model: Inspired by the stick-slip behavior of granular media, we consider contact models that limit the possible frictional contact wrenches between a rigid robot foot and a granular substrate with a convex wrench space, which can be learned from data obtained in physical experiments [5]. If the external wrench can be resisted by a wrench in the wrench space, the contact sticks, and otherwise it slips. The shape of the wrench space changes as the penetration depth and orientation of the rigid body change, shown in Fig. 1, and here we have a learned twice-differentiable contact model w.r.t. the robot configuration:

$$f \in \mathcal{W} \iff f \in \left\{ \sum_{i=1}^{V} w_i v_i(q) \middle| w_i \geq 0 \wedge \sum_{i=1}^{V} w_i \leq 1 \right\}. \quad (3)$$

This is a vertex representation of the convex wrench space, where V is the total number of vertices and $v_i(q)$ is a learned vertex, which is dependent on the robot's configuration and the terrain shape.

2) Contact detection: We only consider contacts between the robot feet and the granular substrate. At a single contact, each learned contact wrench vertex $v_i(q)$ is a smooth function of penetration depth and orientation, i.e.:

$$v_i(q) = v_i(-z^T p(q), R(q)),$$

where $p(q)$ is the bottom center point of the robot foot that is having contact, z is the outward normal at the surface point that is closest to p, and $R(q)$ is the 3×3 local-to-global rotation matrix. In our prior work, we take the assumption that the environment is a flat terrain paved using granular materials. Here, we propose a heuristic method to generalize this model to an arbitrary terrain shape represented using a watertight mesh. Since the terrain is a watertight mesh, we can compute a singled distance value from $p(q)$ to the surface of the mesh, which is denoted as $d(p(q))$. We can also estimate the generalized outward normal direction as:

$$z(q) = \nabla d(p(q)).$$

We further denote $\bar{R}(\bar{z})$ as the minimal rotation matrix that transforms $[0, 0, 1]^T$ to $\bar{z}$. The generalized vertex of the contact wrench space is then defined as:

$$\bar{v}_i(q) = \bar{R} v_i(d(p(q)), \bar{R}^T R(q)),$$
$$f \in \mathcal{W} \iff f \in \left\{ \sum_{i=1}^{V} w_i \bar{v}_i(q) \middle| w_i \geq 0 \wedge \sum_{i=1}^{V} w_i \leq 1 \right\}. \quad (4)$$

Intuitively, this is a generalized pullback operator from the arbitrary shape terrain to the flat ground. Note that to form a well-defined wrench space that can be used in a bilevel optimization, $\bar{v}_i$ must be a twice-differentiable function of q. However, the generalized penetration depth computed using an exact triangular mesh is generally non-differentiable. To remedy this case, we first compute a signed distance field for the mesh on a uniform grid and then use cubic interpolation to approximate d.

3) Contact wrench solving: The robot dynamics is solved based on the MDP [6], which states that the frictional contact wrenches would maximize the energy dissipation of the system. Therefore, a lower-level optimization problem is given by:

$$\begin{aligned} &\mathbf{argmin}_{f} \\ &\frac{1}{2} \| M(q_k)^{-1} (u_k + J_c^T f - H(q_k, \dot{q}_k)) \cdot dt + \dot{q}_k \|^2_{M(q_k)} \quad (5) \\ &\mathbf{s.t.}\ f \in \mathcal{W}. \end{aligned}$$

Here we slightly abuse the notation and let f and $\mathcal{W}$ denote a concatenation of all the contacts instead of a single contact. The program is a QP which can be solved efficiently in polynomial time, and the gradients of this program w.r.t. all problem parameters can be obtained by sensitivity analysis, whose details we refer readers to [21]. One concern in sensitivity analysis is the loss of linear independence constraint qualification (LICQ), which gives non-unique gradients. The constraints in Eqn. 5 are learned from data and we cannot guarantee LICQ, and in practice we do observe the loss of LICQ in some cases. We handle this by using a primal, interior point method with a pre-specified duality gap to solve the QP, where the solution contact wrenches as a function of problem parameters, i.e. $f(q_k, \dot{q}_k, u_k)$, is always single-valued and twice-differentiable.

If we assume that the QP in Eqn. 5 is strictly convex, then function f is single-valued. However, using this single-valued function f in a bilevel optimization is inappropriate because it is known that f is only C^0-continuous [22], while large-scale constraint optimizers usually require twice-differentiable constraints [23]. To ensure both strict convexity and twice-differentiability, we propose to use a primal, interior-point solver with a finite duality gap. In this way, the QP is transformed into the following unconstrained optimization:

$$\begin{aligned} &\mathbf{argmin}_{f} \\ &\frac{1}{2} \| M(q_k)^{-1} (u_k + J_c^T f - H(q_k, \dot{q}_k)) \cdot dt + \dot{q}_k \|^2_{M(q_k)} \quad (6) \\ &- \mu \cdot dt \sum_{i=1}^{V} \log(w_i) - \mu \cdot \log(1 - \sum_{i=1}^{V} w_i), \end{aligned}$$

where μ is the finite log-barrier coefficient. Since the log-barrier function always contributes a full-rank Hessian matrix, the above problem is strictly convex and the function f is single-valued. Further, by the implicit function theorem, the function f is twice-differentiable as long as the problem data, i.e. M, H, J, v_i, are sufficiently smooth. In practice, we solve the unconstrained optimization using Newton's method with the initial guess from last iteration of upper-level optimization.

As a result, for a twice-differentiable learned contact model, the contact wrenches are a smooth function of problem parameters, whose gradient information can be robustly obtained via sensitivity analysis, which we denote as $\nabla f(q_k, \dot{q}_k, u_k)$. In addition, since we already calculate the gradients of all problem data during sensitivity analysis.

Specifically, denoting Eqn. 2 as $g(q_k, \dot{q}_k, u_k)$, its gradient $\nabla g(q_k, \dot{q}_k, u_k)$ is obtained via the chain-rule.

C. Bilevel trajectory optimization

Our method uses a direct transcription TO method, where the TO is formulated as a non-linear program (NLP) that locally optimizes a physically correct discrete trajectory. Overall, our method solves the following NLP:

$$\begin{aligned} \underset{\boldsymbol{x}[\cdot],\boldsymbol{u}[\cdot],\boldsymbol{c}[\cdot]}{\textbf{argmin}} \; & J(\boldsymbol{x}[\cdot],\boldsymbol{u}[\cdot],\boldsymbol{c}[\cdot]) \\ \textbf{s.t. } & \boldsymbol{x}[0] \in \mathcal{X}_{init}, \boldsymbol{x}[N] \in \mathcal{X}_{goal}, \\ & \boldsymbol{x}[\cdot] \in \mathcal{X}, \boldsymbol{u}[\cdot] \in \mathcal{U}, \\ & \text{Eqn. 2 holds}, \\ & h(\boldsymbol{x}[\cdot],\boldsymbol{u}[\cdot],\boldsymbol{c}[\cdot]) = 0, \end{aligned} \tag{7}$$

where $\boldsymbol{c}[\cdot] \in \mathbb{R}^{n_c}$ represents additional problem-specific variables. The feasible regions for the initial and final states are specified in $\mathcal{X}_{init}$ and $\mathcal{X}_{goal}$. The feasible states and controls along the trajectory are defined in $\mathcal{X}$ and $\mathcal{U}$. N is the total number of grids in the discrete trajectory. J is some performance measure we aim to optimize. Other additional equality constraints are encoded in $h(\cdot)$.

This problem is bilevel because the dynamics constraints Eqn. 2 entail solving a set of lower-level optimization problems, i.e. Eqn. 5, which were introduced in the previous section. In practice, solving such a challenging TO problem using a NLP solver requires analytical gradients of the objective function and the constraints. In particular, the analytic gradients of the dynamics constraints $\nabla g(q_k, \dot{q}_k, u_k)$, given by our differentiable simulator, are supplied to the NLP solver, along with the gradients of other constraints and the objective function in Eqn. 7.

D. Spline constraints

To improve the smoothness of the optimized locomotion gaits, we present a scheme to use splines to regularize the end effector movement and avoid "stutter step". Intuitively, we expect an optimized robot gait where each robot foot traces a smooth path. In particular, for each robot foot, we represent its position and orientation with a cubic spline whose control points are $\boldsymbol{c}[\cdot] \in \mathbb{R}^n$, where n is 3 in 2D and 6 in 3D. We use $T_j(k) : [0, N-1] \to \mathbb{R}^n$, to denote the interpolated position and angle of robot foot j along the cubic spline specified by the control points, where N is the total number of grid points in the discrete robot trajectory. In addition, let $FK_j(x) : \mathbb{R}^{n_x} \to \mathbb{R}^n$ denote the forward kinematics of the robot that gives the position and angle of the j-th robot foot, the spline constraints are formulated as:

$$FK_j(x[k]) = T_j(k), \forall j, k. \tag{8}$$

In practice, we might only use partial dimensions of a spline, i.e. we may choose to use only the position for the spline control points, without the angle.

E. Cost function

We propose a cost function that includes 3 different components, where the third component is optional: $J_{total} = J_{path} + J_{periodic} + J_{spline}$. J_{path} is the quadratic path cost defined as $J_{path} = \sum_k \|\boldsymbol{x}[k] - x_{ref}\|_Q + \|\boldsymbol{u}[k]\|_R$, where x_{ref} is the reference robot state that is manually selected. We further regularize the trajectory by introducing a periodic cost that encourages the robot limbs to move periodically: $J_{periodic} = \sum_{k=0}^{N-P} c_p (F \cdot q_k - F \cdot q_{k+P})^2$, where c_p is a weight constant, F is a selection matrix that selects only the limbs of the robot, and P is the desired period.

Instead of having the splines be hard constraints, we find in practice that soft constraints might work better for certain terrains. Therefore, we propose an alternative formulation where we use a spline cost to replace the spline constraints. Letting $h(\cdot)$ denote the constraints in Eqn. 8, the spline cost is $J_{spline} = c_s \|h(\cdot)\|^2$, where c_s is a weight constant.

F. Full formulation

A final consideration is that for contact models learned from data, it is prudent to limit the input, i.e. depth and orientation, to stay close to the range of collected data, because learned models have questionable extrapolation capability. We encode this constraint, along with the robot joint position and velocity limits, in $\mathcal{X}$. In addition, we have the joint torque limits $u[\cdot] \in \mathcal{U}$.

The complete formulation is the following:

$$\begin{aligned} \underset{\boldsymbol{x}[\cdot],\boldsymbol{u}[\cdot],\boldsymbol{c}[\cdot]}{\textbf{argmin}} \; & J_{total} \\ \textbf{s.t. } & \boldsymbol{x}[0] \in \mathcal{X}_{init}, \boldsymbol{x}[N] \in \mathcal{X}_{goal}, \\ & x[\cdot] \in \mathcal{X}, \boldsymbol{u}[\cdot] \in \mathcal{U}, \\ & \text{Eqn. 2 holds}, \\ & \text{Eqn. 8 holds.} \quad \text{(if not using spline cost)} \end{aligned} \tag{9}$$

Note that $\boldsymbol{c}[\cdot]$ represents the cubic spline control points for all limbs.

IV. Results

A. Robot and contact model

We use a Robosimian quadruped robot model for all the experiments, shown in Fig. 1. We consider a 2D model where only sagittal movement of Robosimian traversing soft terrain presents. The robot has a total of 4 limbs and 12 revolute joints, making $n_x = 30$ (position and velocity of the joints and torso) and $n_u = 12$.

The contact model we consider here is the learned granular contact model for sand from our previous work [5]. We use radial basis function (RBF) interpolation as the learning model for data gathered from physical experiments.

B. Desired gait and performance metric

Our experiment attempts to generate a walking gait that follows a desired torso forward velocity on different terrains. Since the cost of an optimized solution does not directly reflect the quality of a gait, we define 2 metrics to quantitatively evaluate it. The first metric measures how well the desired torso velocity is tracked: $M_v = \sum_{i=0}^{N} |x_{v,i} - \hat{x}_v| / N$, where $x_{v,i}$ is the torso forward velocity at grid point i, and $\hat{x}_v$ is the desired torso forward velocity. The second metric M_c measures the cost of transport, which is $\sum_{k=0}^{N} \sum_{i=0}^{n_u} |(F \cdot \dot{q}_k)[i] \cdot \boldsymbol{u}[k][i]| / (smg)$, where $(F \cdot \dot{q}_k)[i]$ and $\boldsymbol{u}[k][i]$ are

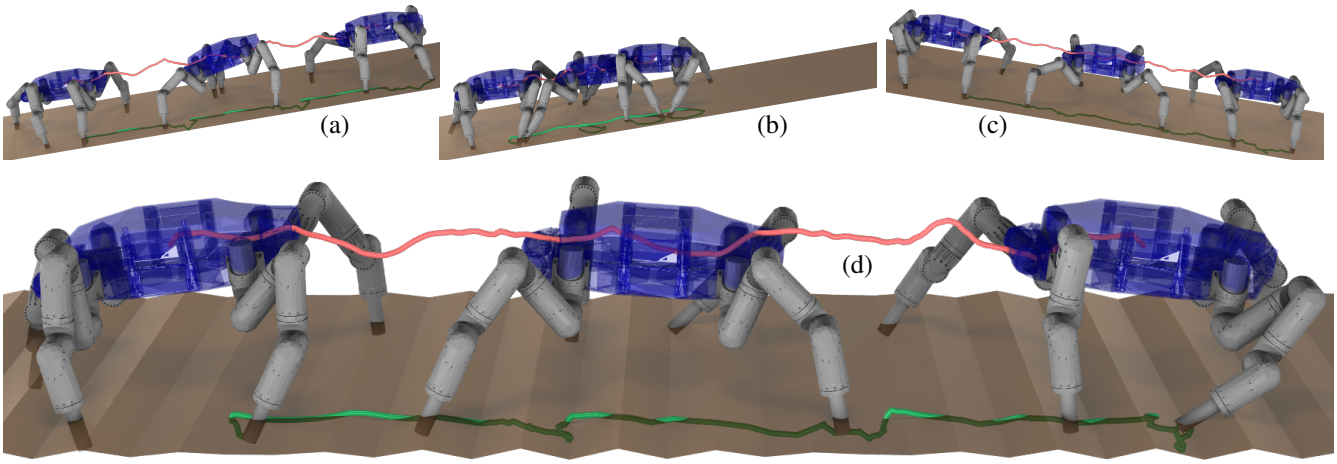

Fig. 2. (a): optimized gait on the 10° slope. (b): non-optimized gait on the 10° slope. (c): optimized gait on the -10° slope. (d): optimized gait on the wavy terrain. The torso centroid is traced in red, while the front right foot position in green. [Best viewed in color.]

the i-th joint velocity and joint control at grid k, s is the total translation of the torso, and mg is the weight of the robot. Note that for humans, the cost of transport of walking on sand has been reported to be 1.6–2.5 times that of walking on hard ground [24].

C. Parameter tuning

The optimization problem tackled here is very challenging, and we found in practice that the convergence behavior and trajectory quality are sensitive to parameters. Here is a summary of all the parameters in our formulation:

- Total number of grid points N, time step dt.
- The quadratic state cost matrix Q is a diagonal matrix, whose only nonzero diagonal entries are those for the torso forward velocity, torso orientation, and torso height, denoted by Q_v, Q_o, Q_h.
- The nonzero entries of x_{ref} include the desired forward velocity, torso height, and torso orientation, denoted by $\hat{x}_v, \hat{x}_o, \hat{x}_h$.
- The quadratic control cost matrix R, which is diagonal.
- The period P and the periodic cost weight c_p.
- Spline dimensions, and the number of control points N_c for each limb. Whether to use spline constraints or cost. If using constraint, constraint scaling for spline constraints. If using spline cost, weight c_s.
- μ, the log-barrier coefficient in lower-level optimization.

In our experiments, we manually pick the parameters $N, dt, \bar{x}_v, \bar{x}_o, \bar{x}_h, P, N_c, \mu$, which can be chosen based on the desired properties of the robot gait. For the rest of the parameters, we run a grid search, and pick the set of parameters that generate the gait of highest quality.

D. Results on a horizontal terrain

In this experiment, the goal is to generate gaits for the Robosimian on a flat horizontal sandy terrain while maintaining a desired forward torso velocity. The parameters are: $N = 181$, $dt = 0.05$ (9s trajectory), $Q_v = 100.0, Q_o = 0.01, Q_h = 0.01$, $x_v = 0.4, x_o = 0.0, x_h = 0.75$, $R = 10^{-5}I$ (I is an identity matrix), $P = 60$, $c_p = 0.1$, $\mu = 10e^{-3}$, all in standard units. Instead of using both the positions and orientations for the splines, we use only the orientations as constraints without constraint scaling, and $N_c = 30$.

We use the Knitro NLP solver [23] for solving the TO. For all the experiments, we set the feasibility and optimality tolerence to 10^{-3}. All experiments are conducted on a standard PC with a Xeon 20-core CPU. During experiments, we found that the optimized results are sensitive to the initial guess. We first experimented with a trivial initialization, where the robot stands still. The optimized trajectory shows the robot sliding and dragging its feet to move forward, without ever lifting them off the terrain. To minimize the amount of manual tuning, we end up using a simple trotting in place initialization, where the robot walks in place for 3 steps. In particular, we manually design the trajectory, track the trajectory with a PID controller at 500 Hz, down-sample the states and average the controls to obtain the trajectory with $dt = 0.05$ s.

The cost and constraint violation over the iterations are shown in Fig. 3, while the optimized trajectory is shown in Fig. 1. The performance metrics are $M_v = 0.0023$ and $M_c = 14.6$, showing that the desired velocity is tracked very well. Visually, the robot has a smooth trajectory with little jerkiness. This particular gait took 0.499 hr to compute.

1) Baseline comparison: Next, we compare with 2 baselines: standard trajectory optimization (iLQR) and reinforcement learning (RL). We use the open source implementation of the iLQR algorithm [17] at https://github.com/anassinator/ilqr. For RL, we tested open source implementation of the PPO [25] and TRPO [26] algorithms at https://github.com/openai/baselines. Default algorithm settings were used.

In iLQR, we use J_{path} directly as the cost function, where the weights and parameters are the same as our method. Tests indicated that iLQR needs a feasible initial guess and is numerically sensitive to the time step and planning horizon. Any combination that would give a 9 s horizon would cause the algorithm to fail completely, and could only achieve feasibility with a short horizon. Our tuning showed the best

TABLE I

OPTIMIZED TRAJECTORY PERFORMANCE ON DIFFERENT TERRAINS (M_v: VELOCITY ERROR, M_c: COST OF TRANSPORT, NO: NON-OPTIMIZED)

Terrain	0°	2.5°	5°	7.5°	10°	12.5°	15°	17.5°	20°	-2.5°	-5°	-7.5°	-10°	-12.5°	-15°	-17.5°	-20°	wavy	10° NO
$\mathbf{M_v}$	0.0010	0.0023	0.013	0.0012	0.013	0.0031	0.036	0.0054	0.29	0.0056	0.0013	0.014	0.0044	0.046	0.040			0.010	0.22
$\mathbf{M_c}$	14.6	12.5	14.8	14.0	17.6	24.5	25.4	27.6	97.3	7.58	12.6	21.5	29.2	48.1	73.2	fail	fail	26.73	35.2
Time (hr)	0.50	2.52	0.25	0.48	0.57	1.03	0.55	1.17	1.1	0.90	0.58	0.36	1.1	3.9	8.2			0.51	NA

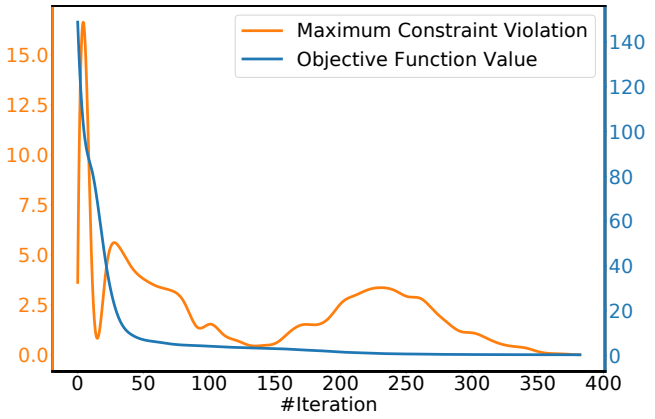

Fig. 3. The cost and maximum constraint violation over the iterations for optimizing a trajectory on the flat horizon terrain.

results at $dt = 0.005$ s and 0.25 s horizon. We then apply iLQR in an online fashion, where the trajectory is re-planned after executing each optimized control. Because of the short horizon, iLQR is only able to obtain trajectories that shift the body slightly forward before tipping over.

For fairness of comparison, we implement a tuned PD joint controller and have joint position command be the action space, instead of torque for RL. The observation space includes the robot state, ankle poses, and the PD controller torques. The reward function is: $-10\|x_v - 0.4\|^2 - 0.2\|x_o\|^2 - 0.001\|a\|^2 - 10^{-5}\|u\|^2 + 1.0$, where a is a vector of the 4 feet angles and u is the control. In addition, we terminate an episode when $x_h < 0.25 \vee x_h > 1.5 \vee |x_o| > 1.1 \vee \|a\|_1 > 1.2$, and give a termination reward of -10. The time step is 0.005 s and we run the training for 1 M time steps. Overall, results did not demonstrate stable gaits. The roll-out trajectories from the trained controllers caused the robot to fall quickly, and terminated in less than 50 time steps.

These comparisons demonstrate that locomotion on granular terrain is a challenging problem for standard trajectory optimization and RL due to its long time horizons and sensitivity to initial guesses and action choices. In comparison, our method is numerically favorable for long horizons and is fairly tolerant to infeasible initial guess for the NLP solver. However, due to its long computation time it is only suitable for offline gait generation.

E. Results on different terrain shapes

Next, we consider different terrain shapes. We test planar terrains of different slopes, ranging from -20° (downward slope) to 20° (upward slope), with 2.5° apart, and a wavy terrain. Our experiments find that steeper terrains are more challenging for the NLP solver. However, since we already have an optimized trajectory on a 0° terrain, we can obtain a high-quality initialization for similar terrains with minimal effort. Using this idea, starting from a 0° terrain, we track its optimized trajectory on the next slightly steeper terrain and use the tracked trajectory as initialization for this terrain. For the wavy terrain, we use the gait generated on 0° terrain as the reference trajectory during tracking. The parameters used here are the same as the 0° terrain, except for the optimizer's parameters that are tuned using grid search. The performance metrics of the optimized trajectories are summarized in Table I. We also include the performance of a trajectory on the 10° terrain where we simply track the gait optimized on the 0° terrain (denoted as 10° NO). Selected optimized and non-optimized gaits are also displayed in Fig. 2. Animated results on all terrains are available in the supplement video. Our method is able to find high-quality trajectories with small velocity error, except for the 20°, -17.5° and -20° terrains (for the -20° terrain, we use the optimized gait on the -15° terrain as reference during tracking), where the NLP solver fail to converge on the later two terrains. This is because these 3 steep terrains are the most challenging among all. We also observe that the steeper the terrain is, the more energy it consumes to cover the same distance, which matches our intuition. It is interesting that going downhill is actually more energy-consuming for the robot. In addition, looking at the non-optimized trajectory in Fig. 2.b, the robot ankles would slip, as shown by the foot position trace, which causes the robot to have much larger M_v and M_c. Meanwhile, the optimized gait in Fig. 2.a avoids slippage by optimally orienting the feet when touching the ground.

V. CONCLUSION

In conclusion, we propose a bilevel trajectory optimization framework for generating locally optimized trajectories with learned contact models. We demonstrate through our experiments that we can generate high-quality trajectories offline with a small amount of parameter tuning effort. In addition, by leveraging optimized results on easier tasks, we are able to find trajectories on challenging terrains.

One drawback of our method is that we generate the trajectories offline for a specific contact model. However, there exist a wide variety of contacts in the real world, and any tiny variation in the composition of a granular terrain would lead to a different contact model. In future work, we could first generate a database of trajectories offline with different wrench spaces and learn a model that generates trajectories given a contact model. During online execution, we could first identify the contact model from gathering a small amount of data, query a trajectory from the learned model, and execute it. This follows the general idea of trajectory learning and meta-learning. In addition, we would like to test the trajectories on a physical robot.

References

[1] B. Henze, M. A. Roa, and C. Ott, "Passivity-based wholebody balancing for torque-controlled humanoid robots in multicontact scenarios," *Int'l. Journal of Robotics Research*, vol. 35, no. 12, pp. 1522–1543, 2016.

[2] H.-j. Kang, K. Hashimoto, K. Nishikawa, E. Falotico, H.-o. Lim, A. Takanishi, C. Laschi, P. Dario, and A. Berthoz, "Biped walking stabilization on soft ground based on gait analysis," *IEEE Int'l. Conf. Biomed. Rob. and Biomech.*, no. 2, pp. 669–674, 2012.

[3] S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini, "STANCE: Locomotion Adaptation over Soft Terrain," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 443–457, 2020.

[4] V. Vasilopoulos, I. S. Paraskevas, and E. G. Papadopoulos, "Compliant terrain legged locomotion using a viscoplastic approach," *IEEE Int'l. Conf. on Intelligent Robots and Systems*, no. Iros, pp. 4849–4854, 2014.

[5] Y. Zhu, L. Abdulmajeid, and K. Hauser, "A Data-driven Approach for Fast Simulation of Robot Locomotion on Granular Media," in *IEEE Int'l. Conf. on Robotics and Automation*, 2019.

[6] E. Drumwright and D. A. Shell, "Modeling contact friction and joint friction in dynamic robotic simulation using the principle of maximum dissipation," *Springer Tracts in Advanced Robotics*, vol. 68, no. STAR, pp. 249–266, 2010.

[7] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int'l. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783–792, 2010.

[10] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," *IEEE Int'l. Conf. on Robotics and Automation*, vol. 2016-June, pp. 1366–1373, 2016.

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

[12] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, 2019.

[13] B. Landry, J. Lorenzetti, Z. Manchester, and M. Pavone, "Bilevel Optimization for Planning through Contact: A Semidirect Method," *arXiv preprint*, 2019.

[14] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, 2018.

[15] I. Chatzinikolaidis, Y. You, and Z. Li, "Contact-Implicit Trajectory Optimization Using an Analytically Solvable Contact Model for Locomotion on Variable Ground," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6357–6364, 2020.

[16] D. Stewart and J. Trinkle, "Friction, An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb," *Int'l Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[17] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," *IEEE Int'l . Conf. on Intelligent Robots and Systems*, no. October 2012, pp. 4906–4913, 2012.

[18] A. Sinha, P. Malo, and K. Deb, "A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2018.

[19] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," *IEEE Int'l. Conf. on Robotics and Automation*, pp. 93–100, 2017.

[20] G. Tang, W. Sun, and K. Hauser, "Time-Optimal Trajectory Generation for Dynamic Vehicles: A Bilevel Optimization Approach," *IEEE Int'l. Conf. on Intelligent Robots and Systems*, pp. 7644–7650, 2019.

[21] A. V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. 1983.

[22] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Zico Kolter, "Differentiable convex optimization layers," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.

[23] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An Integrated Package for Nonlinear Optimization," *Large-scale nonlinear optimization*, pp. 35–59, 2006.

[24] T. M. Lejeune, P. A. Willems, and N. C. Heglund, "Mechanics and energetics of human locomotion on sand.," *Journal of Experimental Biology*, vol. 201, no. 13, pp. 2071–2080, 1998.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, pp. 1–12, 2017.

[26] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," *32nd Int'l. Conf. on Machine Learning*, vol. 3, pp. 1889–1897, 2015.