Rapid and Robust Trajectory Optimization for Humanoids

Bohao Zhang[1] and Ram Vasudevan[2]

***Abstract*— Performing trajectory design for humanoid robots with high degrees of freedom is computationally challenging. The trajectory design process also often involves carefully selecting various hyperparameters and requires a good initial guess which can further complicate the development process. This work introduces a generalized gait optimization framework that directly generates smooth and physically feasible trajectories. The proposed method demonstrates faster and more robust convergence than existing techniques and explicitly incorporates closed-loop kinematic constraints that appear in many modern humanoids. The method is implemented as an open-source C++ codebase which can be found at https://roahmlab.github.io/RAPTOR/.**

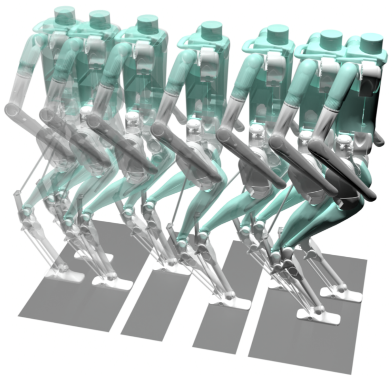
Fig. 1: This figure illustrates a physically feasible six-step periodic gait with different step lengths for humanoid robot Digit that is found by RAPTOR, the trajectory optimization algorithm developed in this paper. The duration of each step is fixed to be 0.4 seconds, which yields a total duration of 2.4 seconds of the whole gait. It only takes RAPTOR 16 seconds to converge to a feasible solution.

I. Introduction

Numerically constructing energy efficient trajectories that accomplish a user-specified task for high degree of freedom robots is an important challenge. A classic approach to perform this type of synthesis for bipeds has relied on representing the problem as a nonlinear optimization program. These methods usually take several minutes to compute an optimal trajectory for bipeds (e.g., Cassie or Digit) [1], [2]. Notably many modern bipeds have closed-loop kinematic constraints that require incorporating nonlinear constraints during optimization, which can make it even more computationally taxing to construct a trajectory via optimization. Solving this nonlinear program often requires managing numerous hyperparameters that can significantly affect the performance of the optimization process. Notably solving these nonlinear optimization problems often requires a good initial guess.

To improve the speed at which trajectory synthesis occurs, many researchers have simplified model complexity by utilizing reduced-order dynamic models [3]–[7]. Of particular note is the centroidal dynamics representation of a legged robot when performing trajectory optimization [8]. It is able to more tractably deal with kinematic obstacle avoidance constraints, but because it utilizes a reduced-order dynamic model during trajectory optimization the solution that it finds may not be directly realizable on a real legged system due to torque limits [9, Section IV.A].

Rather than utilize a reduced order model, others have tried to improve the speed of computation of these nonlinear programs by reformulating the trajectory optimization problem. For example, researchers have utilized Differential Dynamic Programming (DDP) to perform energy efficient trajectory design for high degree of freedom robotic systemsin seconds [10], [11]. However, the contact constraints or other kinematics constraints are treated as soft constraints that are directly added to the cost function to minimize. This means that these kinematic constraints may not be perfectly satisfied at an optimal solution. To address this limitation, researchers have recently illustrated how to perform DDP with rigid constraints [12]. This approach is able to generate a solution that respects the system dynamics while satisfying other state or input constraints, but is slower than [10].

The emphasis of this work is on performing trajectory optimization when a user pre-specifies the contact sequence. However, there are methods that try to optimize over the contact sequence as well [13], [14], but because this dramatically increases the number of optimization variables it can be difficult to apply these approaches while performing gait optimization for humanoids.

More recent work has applied deep learning to perform trajectory design for bipeds [15], [16]. Using iterative reinforcement learning, researchers have learned a policy that can enable a robot to successfully traverse a variety of terrains, including slopes, stairs, or even blocks [17]. These methods usually begin from or rely on an existing library of reference trajectories for the deep neural network to imitate [18]. This can be nontrivial to automatically generate on a high-dimensional robot.

This paper proposes a new gait optimization framework, RAPTOR, that optimizes over only the actuated joints of a robot and then uses inverse dynamics to reconstruct rest of the states. This results in fewer decision variables and

[1]Robotics Institute, University of Michigan, Ann Arbor, MI jimzhang@umich.edu.
[2]Mechanical Engineering, University of Michigan, Ann Arbor, MI ramv@umich.edu.

constraints during optimization-based trajectory design. As a result, RAPTOR can perform trajectory design faster for modern bipeds which usually incorporate closed-loop kinematic constraints. The contributions of this work are threefold: First, in Section III, a new formulation of a generalized gait optimization framework for dynamic bipedal locomotion that directly generates a smooth trajectory that is physically feasible and can incorporate closed-loop constraints. Second, in Section IV, we demonstrate that our method can be effectively scaled to a high-dimensional full-size humanoid and achieve faster and more robust convergence compared with other methods. Finally, we release the code-base, which is written entirely in C++, as an open source package https://github.com/roahmlab/RAPTOR.

II. Preliminary

This section formulates the bipedal robot dynamics as well as certain assumptions that are used to formulate a trajectory optimization problem to generate safe trajectories. To represent the reference trajectories of the robot, this paper uses Bezier curves, $b : [0, 1] \to \mathbb{R}$, which are defined as:

$$b(t) = \sum_{v=0}^{V} b_v P_v(t), \tag{1}$$

where $b_v \in \mathbb{R}$ are called the *Bezier Coefficients* and $P_v : [0, 1] \to \mathbb{R}$ is the *Bernstein Basis Polynomial* [19, (1)] for each $v \in \{0, \ldots, V\}$.

The robot has n joints in total, with n_a actuated joints (motors) and n_u unactuated (passive) joints, where $n_a + n_u = n$. We denote the set of indices for all actuated joints as $\mathcal{A} \subset \mathbb{N}$ and the set of indices for all unactuated joints as $\mathcal{U} \subset \mathbb{N}$. Thus, $\mathcal{A} \cap \mathcal{U} = \emptyset$ and $\mathcal{A} \cup \mathcal{U} = \{1, \ldots, n\}$.

We focus on robots with flat feet in this paper:

Assumption 1. *The lower surface of the feet of the biped are flat rectangles.*

Motivated by [20, Section 1.2], the bipedal robot in this paper is modeled as a hybrid system that alternates between single-support and double-support phases depending on the number of legs in contact with the ground. In the single-support phase, the leg in contact with the ground is called the *stance leg*, and the other leg is called the *swing leg*.

1) Single-Support Dynamics: Suppose the duration of a single-support phase is T. We define the trajectory of the robot as $q : [0, T] \to \mathcal{Q}$ where $\mathcal{Q} \subset \mathbb{R}^n$ is the configuration space of all joints of the robot. We define $q_u(t) \in \mathbb{R}^{n_u}$ as the collection of all unactuated entries of $q(t)$ and $q_a(t) \in \mathbb{R}^{n_a}$ as the collection of all actuated entries of $q(t)$. The constraint function is denoted by $c : \mathcal{Q} \to \mathbb{R}^{n_c}$ where $n_c \geq 0$ is the number of constraints. The constraints describe the closed-loop kinematic chain constraints and the requirement that the stance foot is fixed at a specific position and orientation. The constraints are satisfied when $c(q(t)) = \mathbb{0}_{n_c}$. The Jacobian of c at $q(t)$ is denoted by $J(q(t)) \in \mathbb{R}^{n_c \times n}$.

The robot's dynamics during the single-support phase for all $t \in [0, T]$ can be described as [21, Section 8.1]:

$$\begin{aligned} H(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + g(q(t)) = \\ = \tau(t) + J^T(q(t))\lambda(t), \end{aligned} \tag{2}$$

$$c(q(t)) = \mathbb{0}_{n_c} \tag{3}$$

$$\dot{c}(q(t)) = J(q(t))\dot{q}(t) = \mathbb{0}_{n_c} \tag{4}$$

$$\ddot{c}(q(t)) = J(q(t))\ddot{q}(t) + \dot{J}(q(t))\dot{q}(t) = \mathbb{0}_{n_c}, \tag{5}$$

where $H(q(t)) \in \mathbb{R}^{n \times n}$ is the positive definite inertia matrix, $C(q(t), \dot{q}(t)) \in \mathbb{R}^{n \times n}$ is the Coriolis matrix, $g(q(t)) \in \mathbb{R}^n$ is the gravitational force vector, and $\lambda(t) \in \mathbb{R}^{n_c}$ is the reaction forces/wrenches to maintain the constraints at time t. The torque applied on each joint at time t, $\tau(t) \in \mathbb{R}^n$, is given by

$$\tau(t) = Bu(t), \tag{6}$$

where $u(t) \in \mathbb{R}^{n_a}$ is the control input and $B \in \mathbb{R}^{n \times n_a}$ is the transmission matrix, which follows the property that

$$B_u = \mathbb{0}_{n_u \times n_a}, \quad B_a = \mathbb{1}_{n_a \times n_a}, \tag{7}$$

where B_a and B_u is the collection of all actuated rows and all unactuated rows of B, respectively.

The conditions for maintaining surface contact between the stance foot and the ground are given in [22, (6)-(8)]. In particular, the reaction force/wrench to maintain all constraints at time t, $\lambda(t) \in \mathbb{R}^{n_c}$, includes force/wrench on unactuated joints to maintain closed-loop constraints, or force/wrench from the ground to maintain contacts between the stance foot and the ground. Let $\lambda_{st}(t)$ be the 6-dimensional vector that is a collection of all entries that correspond to the contact constraints in $\lambda(t)$ (i.e., the contact wrench). The corresponding contact wrenches consist of three constraint forces, $(\lambda_{st}^{fx}(t), \lambda_{st}^{fy}(t), \lambda_{st}^{fz}(t))$, and three constraint moments, $(\lambda_{st}^{mx}(t), \lambda_{st}^{my}(t), \lambda_{st}^{mz}(t))$, respectively. To ensure that contact is maintained, the following conditions on contact wrenches must be satisfied:

$$0 \leq \lambda_{st}^{fz}(t) \tag{8a}$$

$$\sqrt{(\lambda_{st}^{fx}(t))^2 + (\lambda_{st}^{fy}(t))^2} \leq \mu \lambda_{st}^{fz}(t) \tag{8b}$$

$$\lambda_{st}^{mz}(t) \leq \gamma \lambda_{st}^{fz}(t) \tag{8c}$$

$$-\frac{1}{2} l_a \lambda_{st}^{fz}(t) \leq \lambda_{st}^{mx}(t) \leq \frac{1}{2} l_a \lambda_{st}^{fz}(t) \tag{8d}$$

$$-\frac{1}{2} l_b \lambda_{st}^{fz}(t) \leq \lambda_{st}^{my}(t) \leq \frac{1}{2} l_b \lambda_{st}^{fz}(t), \tag{8e}$$

where μ is the translational friction coefficient of the ground, γ is the torsional friction coefficient of the ground, l_a and l_b are the length and width of the lower surface of the stance foot. Note these conditions correspond to the following condition: the ground reaction force should be non-negative (8a), the stance foot should not slide on the ground in translation direction (8b), or in rotation direction (8c), and the robot should not roll over the edge of the stance foot (8d), (8e).

2) Double Support Dynamics: To simplify the exposition and formulation of the real-time optimization problem, we follow the assumption made in [20, Section 3.2, HGW3]:

Assumption 2. *The double support phase is instantaneous, and the associated impact due to ground contact is modeled as a rigid contact.*

We describe the instantaneous change in the robot model caused by the impact using the notion of a guard and reset map as in the definition of hybrid systems [23, Definition 7]. The force of ground contact imposes a holonomic constraint on the position of the stance foot that enables one to construct a reset map [20, Section 3.4.2]:

$$(q^+, \dot{q}^+, \lambda_r) = \Delta(q^-, \dot{q}^-), \tag{9}$$

where Δ describes the relationship between the pre-impact joint angles, q^-, and velocities, $\dot{q}^-$, and post-impact joint angles, q^+, velocities, $\dot{q}^+$, and reaction force/wrench λ_r. For brevity, we do not include an explicit formula for Δ in this paper, but it can be found in [20, (3.20)]. Note that this instantaneous change only affects the velocities of the robot joints (i.e., $q^+ = q^-$).

3) Fully-Actuated Representation: When the number of constraints n_c is equal to the number of unactuated dimensions n_u, the system is fully actuated. This case usually holds for bipedal robots with actuated ankles and flat feet in the single-support phase [20, Section 10]. More details on the discussion on Digit and other similar robots can be found in Section IV. The fully-actuated representation facilitates the control and optimization of such robots because it enables us to describe the unactuated joints trajectory as a function of the actuated joints trajectory.

Before formally describing this theorem, we make the following assumption:

Assumption 3. *During the single-support phase, for $\forall q_a(t) \in \mathcal{Q}_a$, there exists one and only one $q_u(t) \in \mathcal{Q}_u$ such that $c(q(t)) = \mathbb{0}_{n_c}$.*

This assumption ensures that the constraint Jacobian $J_u(q(t)) \in \mathbb{R}^{n_c \times n_u}$, which is the collection of unactuated columns of the constraint Jacobian $J(q(t))$ and a square matrix when $n_u = n_c$, is always invertible according to the inverse function theorem [24, Theorem 5.2.1]. Note in the experimental section, we numerically evaluate whether this assumption is satisfied. We denote the q_u that satisfies the constraints as a function of q_a:

$$q_u = \Gamma(q_a). \tag{10}$$

Under this assumption, one can prove that the control input can be uniquely computed as a function of the actuated joint position, velocity, and acceleration. The proof can be found in our online supplementary material [25, Appendix I].

Theorem 4. *Given actuated joint position $q_a(t)$, velocity $\dot{q}_a(t)$, and acceleration $\ddot{q}_a(t)$, the velocity and acceleration of all joints are given as*

$$\dot{q}(t) = G(q(t))\dot{q}_a(t) \tag{11}$$

$$\ddot{q}(t) = G(q(t))\ddot{q}_a(t) + \dot{G}(q(t))\dot{q}_a(t). \tag{12}$$

The reaction force $\lambda(t)$ and the control input $u(t)$ are then uniquely given as

$$\lambda(t) = (J_u^{-1}(q(t)))^T \tilde{\tau}_u(t) \tag{13}$$

$$u(t) = \tilde{\tau}_a(t) - J_a(q(t))^T \lambda(t) = G^T(q(t))\tilde{\tau}(t), \tag{14}$$

where $\tilde{\tau}(t)$ is the full inverse dynamics vector:

$$\tilde{\tau}(t) = H(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + g(q(t)). \tag{15}$$

$\tilde{\tau}_u(t)$ and $\tilde{\tau}_a(t)$ are the collection of unactuated and actuated entries of $\tilde{\tau}(t)$, respectively, $J_u(q(t))$ and $J_a(q(t))$ are the collection of unactuated and actuated columns of the constraint Jacobian $J(q(t))$, respectively, $G(q(t)) \in \mathbb{R}^{n \times n_a}$ is the projection matrix from actuated space to joint space with its unactuated rows defined as

$$G_u(q(t)) = -J_u^{-1}(q(t))J_a(q(t)), \tag{16}$$

and its actuated rows defined as

$$G_a(q(t)) = \mathbb{1}_{n_a \times n_a}. \tag{17}$$

As we show in the next section, this theorem allows us to formulate the trajectory optimization problem with decision variables only corresponding to the actuated joints. In particular, we can apply this theorem to recover the remaining joints, reaction force, and the control input as a function of just the actuated joints.

III. Multiple-step Periodic Gait Generation

This section describes the optimization formulation for offline generating a library of desired gaits.

The duration of all walking steps to be a fixed number $T \in \mathbb{R}_+$. The trajectory of the actuated joint $j \in \mathcal{A}$ for one walking step is a V-degree Bezier curve:

$$q_j(t) = \sum_{v=0}^{V} b_{v,j} P_v\left(\frac{t}{T}\right), \tag{18}$$

where $t \in [0, T]$. For an L-step gait optimization, the variables at a walking step $l \in \{1, \dots, L\}$ are denoted by $b_{v,j}^{(l)}$. The following variables are decision variables in the optimization problem: $\{b_{v,j}^{(l)\in\mathbb{R}}\}_{v\in\{0,\dots,m\}, j\in\mathcal{A}, l\in\{1,\dots,L\}}$, $\{\dot{q}_r^{(l)} \in \mathbb{R}^n\}_{l\in\{1,\dots,L\}}$, and $\{\lambda_r^{(l)} \in \mathbb{R}^{n_u}\}_{l\in\{1,\dots,L\}}$, where $\dot{q}_r^{(l)}$ denotes the joint velocity after the reset map at the end of each step l, and $\lambda_r^{(l)}$ denotes the reaction force during the reset map at the end of each step l that is required to maintain the corresponding constraints. The variable y describes the concatenation of all of the variables above.

To ensure that the computed trajectory is dynamically feasible, the optimization problem checks that certain constraints are satisfied by the trajectory on a finite number of time nodes. Motivated by [26], we choose the *Chebyshev Nodes*, $t_i \in [0, T]$ for $i \in \{1, \dots, N\}$ as the set of time nodes along which constraints must be satisfied where $N \geq 3$ is the total number of the nodes. This is then used to formulate the optimization problem to design desired gaits:

$$\min_{y} \quad \mathcal{J}(y) \tag{19a}$$
$$\text{s.t.} \quad \text{Joint Limits} \tag{19b}$$
$$\text{Actuator Limits} \tag{19c}$$
$$\text{Maintaining Contact} \tag{19d}$$
$$\text{Torso Constraints} \tag{19e}$$
$$\text{Swing Foot Constraints} \tag{19f}$$
$$\text{Reset Map Constraints} \tag{19g}$$

The remainder of this subsection describes the cost and each of the constraints. Before proceeding, recall that one can use the decision variables of the optimization problem to compute the control input [27, Theorem 2].

1) Cost (19a)*:* Inspired by [8, (6)], the cost function aims to minimize the 2-norm of the control input consumed, the initial velocity, and the initial acceleration:

$$\mathcal{J}(y) = \sum_{l=1}^{L}\Big(\frac{w_1}{N}\sum_{i=1}^{N}\left\|u_i^{(l)}\right\|_2 + \\ + w_2\left\|\dot{q}_a^{(l)}(0)\right\|_2 + w_3\left\|\ddot{q}_a^{(l)}(0)\right\|_2\Big), \tag{20}$$

where w_1, w_2 and w_3 are user-defined positive constant scalars.

2) Joints Limits (19b) *& Actuator Limits* (19c)*:* All joints $q^{(l)}(t_i)$ belong to the configuration space $\mathcal{Q}$ for all $i \in \{1,\dots,N\}$ and $l \in \{1,\dots,L\}$. The control input $u^{(l)}(t_i)$ must be within the robot torque limits for all $i \in \{1,\dots,N\}$ and $l \in \{1,\dots,L\}$.

3) Maintaining Contact (19d)*:* The contact constraints are formulated for $\lambda^{(l)}(t_i)$ and $\lambda_r^{(l)}$ for all $i \in \{1,\dots,N\}$ and $l \in \{1,\dots,L\}$ using (8) and can be computed using the decision variables by applying (13).

4) Torso Constraints (19e)*:* Inspired by [22, (50)], the torso constraints require that the absolute value of the roll and pitch angle of the torso are less than or equal to 3° and the height of the torso larger than a specific value to avoid falling.

5) Swing Foot Constraints (19f)*:* The swing foot constraints require that: the swing foot stays on the ground at the beginning and end of each step, the swing foot yaw angle is equal to a user specified desired angle at the beginning and end of each trajectory, the swing foot height is greater than or equal to some user specified height during the middle of the step, and that the swing foot position is equal to some user specified desired position at the beginning and at the end of every step. More information on the user-defined values can be found in Section IV.

6) Reset Map Constraints (19g)*:* Reset map constraints require that the previous walking step and the next walking step are "aligned" according to (9):

$$(q^{(l)}(x_N), \dot{q}_r^{(l)}, \lambda_r^{(l)}) = \Delta(q^{(l)}(x_N), \dot{q}^{(l)}(x_N)) \tag{21a}$$
$$q_a^{(l+1)}(x_1) = q_a^{(l)}(x_N) \tag{21b}$$
$$\dot{q}_a^{(l+1)}(x_1) = \dot{q}_{ra}^{(l)} \tag{21c}$$

for all $l \in \{1,\dots,L-1\}$. (21b) ensures that the actuated joint positions are continuous between steps. (21a) and (21c) ensure that the actuated joint velocities satisfy the reset map equation (9) between steps.

For periodic gaits, we generally assume that the number of walking steps is even, so that the stance legs at the first step and the last step are different. The corresponding reset map constraints can be formulated as below:

$$(q^{(L)}(x_N), \dot{q}_r^{(L)}, \lambda_r^{(L)}) = \Delta(q^{(L)}(x_N), \dot{q}^{(L)}(x_N)) \tag{22a}$$
$$q_a^{(1)}(x_1) = q_a^{(L)}(x_N) \tag{22b}$$
$$\dot{q}_a^{(1)}(x_1) = \dot{q}_{ra}^{(L)} \tag{22c}$$

If the number of walking steps is odd, in other words, the stance legs at the first step and the last step are the same, the constraints need to be rewritten so that joint positions and velocities at the beginning of the first step and the end of the last step are symmetric. As a result, this only holds for symmetric walking behaviors, such as walking straight forward.

IV. Simulation Experiments

This section summarizes the experimental evaluation of our method on Digit and its comparison to Aligator [12]. Note we have only compared to Aligator rather than other optimization-based trajectory design algorithms because Aligator (1) is able to natively use Pinocchio [28] which efficiently evaluate derivatives of a robot's dynamics, (2) has made its code publicly available, and (3) it is able to enforce closed-loop kinematic constraints. Note, we also performed the same comparison on another humanoid Talos [29] that was originally included in the official examples of aligator and does not contain any closed-loop kinematics chains. The results can be found in our online supplementary material [25]. We have also shown that our method works for other fully-actuated systems, such as robotic manipulators. More results can be found on our project website https://roahmlab.github.io/RAPTOR/.

A. Digit Overview

Digit is a humanoid robot with 42 degrees of freedom, developed by Agility Robotics (including floating-base). Digit includes 4 actuated revolute joints per arm and 14 revolute joints per leg, of which 6 are actuated and 8 are passive. Each leg has 3 closed kinematic chains. The upper leg chain consists of 1 actuated joint (knee) and 4 passive joints (shin, heel-spring, achilles-rod, tarsus). The lower leg chains include 2 actuated joints (toe-A, toe-B) and 4 passive joints (toe-pitch, toe-roll, toe-B-rod, toe-B-rod). The kinematic structure is summarized in Figure 2.

We make the following simplifications for Digit. We focus on the leg motion of Digit in this work. In addition, note the springs on Digit's legs exhibit little movement during actual hardware experiments. In fact, the most recent bipedal robot models ([30]–[33]), including the latest version of Digit [34], do not incorporate such springs. As a result, we make the following assumption:

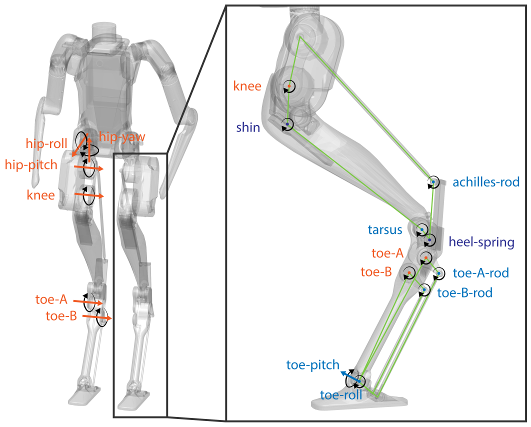

Fig. 2: This figure illustrates the closed-loop structure on the legs of Digit. The orange arrows show the rotation axes of all actuated joints (motors). The blue arrows show the rotation axes of all unactuated joints. The purple arrows show the rotation axes of all springs (shin and heel-spring), which are assumed to be fixed in this paper. The green lines show how these joints are connected to be closed-loops.

Assumption 5. *All arms on Digit are assumed to be fixed. All four springs (shin and heel-spring on both legs) on Digit are assumed to be fixed.*

By fixing all the springs, each chain in the legs becomes a four-bar linkage mechanism with 3 kinematics constraints. This results in a total of 15 joints per leg with 9 passive joints and 9 constraints. To numerically verify Assumption 3, we provide 100 different actuated joint positions within the corresponding joint limits and solve inverse kinematics for unactuated joints. The solutions are always unique. The other unactuated joints, which are the floating-base coordinates, are also unique once the unactuated joints in the closed-loop are settled. As a result, Digit can be treated as a fully actuated system during the single-support phase.

B. *Implementation Details*

1) Platform Details: Our gait optimization code is implemented in C++. The experiments are run on a desktop with an AMD Ryzen 9 5950 16-core 32-thread processor and 128 GB RAM. In our method, we use Pinocchio [28] to compute the unconstrained inverse dynamics and its gradient. We use Ipopt [35] as our nonlinear solver. The first-order gradient is provided analytically while the second-order Hessian is approximated using Ipopt's "limited-memory" option. We use linear solvers from HSL [36].

2) Handling Closed-loop Constraints: To apply Theorem 4, we must first solve for the inverse kinematics Γ. This enables the computation of the other terms in the theorem. We use the multidimensional root finding function in the GSL library [37] to find $q_u(t)$. This method is sensitive to its initial guess. To be more specific, in the offline operation, we uniformly sample 100 points within the actuated joint limits and solve the inverse kinematics problem over each of these 100 points, as how we have numerically verified Assumption 3 described at the end of Section IV-A. We fit a trigonometric series to approximate the solutions. In that way, we can compute an approximate solution rapidly by relying upon interpolation. During each iteration of the optimization, we treat this approximation as the initial guess of the GSL multidimensional root-finding function, which then converges into a more accurate solution, usually in less than 10 iterations. More details on implementation can be found in the README[1] of our code.

C. *Gait Optimization Comparison*

We perform a comparison between RAPTOR and aligator [12] on Digit. Our implementation on aligator has also been open-sourced[2]. We perform 3 different experiments which make Digit walk forward for 0m, 0.4m, and 0.8m forward in one walking step. We assign a fixed initial configuration of the robot to aligator and enforce the same constraint for RAPTOR, so that both methods start from the same configuration at the beginning of the trajectory. We set the maximum number of iterations to 200 and the constraint violation tolerance to 1e-4 for both methods. For aligator, we adhere to the same settings as provided in the official example except the time discretization. We consider 3 different time discretizations (dt = 0.01s, 0.005s, 0.001s) for further discussion on aligator. For RAPTOR, we performed an ablation study to choose the best parameters for Ipopt. The results can be found in our online supplementary material [25, Appendix II]. We choose the degree of the Bezier curve V to 5 and the number of time nodes N to 14. We choose HSL ma57 as the linear solver and the update strategy for barrier parameter to `adaptive`. To initialize the optimization, we adhere to the same initial guess strategy as provided in the official example for aligator. For RAPTOR, we simply assign all decision variables to 0. More details on warm up strategy of RAPTOR can be found in README[1] of our code.

To validate whether solution is physically realizable, we simulate the dynamics of the robot using the integration method `solve_ivp` from [38] in Python. Since the development of the tracking controller is out of the scope of this work, we simply apply a naive control policy here:

$$u(t) = u_{open}(t) + K_P(q_d(t) - q(t)) + K_D(\dot{q}_d(t) - \dot{q}(t)), \quad (23)$$

where $u_{open}(t)$ is the open loop control input generated from the optimization, K_P and K_D are the PD gains. We abuse the notation here and denote $q(t)$ as the robot states in the simulation and $q_d(t)$ as the desired states of the optimized gait trajectory. aligator generates optimized control inputs on its corresponding time discretization, hence we compute $u_{open}(t)$ using zero-order hold (ZOH) over the control input sequence. The motivation of choosing different time discretization here is that, intuitively speaking, finer discretization should lead to better tracking performance when using ZOH. For RAPTOR, since the outputs are continuous trajectories, we can directly compute $u_{open}(t)$ at any time t using Theorem 4. We tuned the PD gains to minimize the tracking error while staying within the torque limitations and eventually choose $K_P = 80$ and $K_D = 5$.

[1]https://github.com/roahmlab/RAPTOR/tree/main/Examples/Digit
[2]https://github.com/roahmlab/aligator-roahmlab

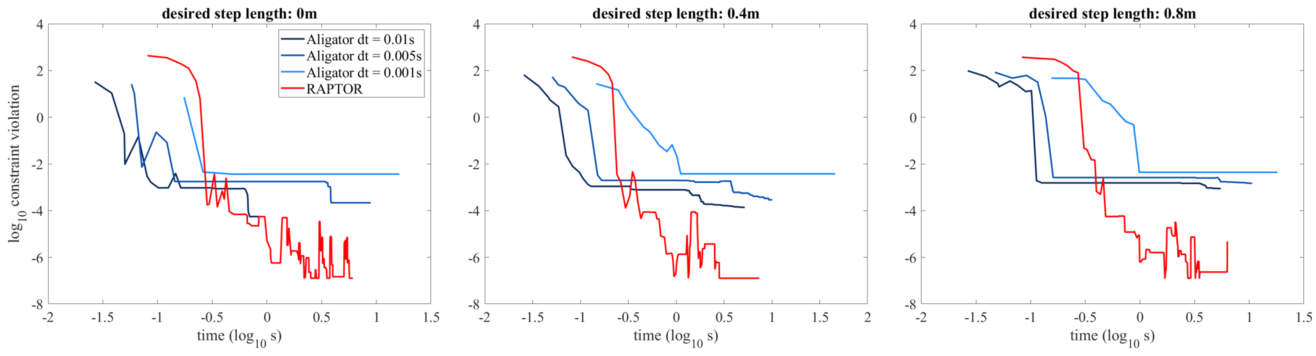

Fig. 3: Pareto curve of the constraint violation with respect to the computation time. Note that all curves do not start from 0 in time, since we need to evaluate the problem for at least one iteration to get the constraint violation. In other words, the beginning of the curves indicates the computation time of the first iteration.

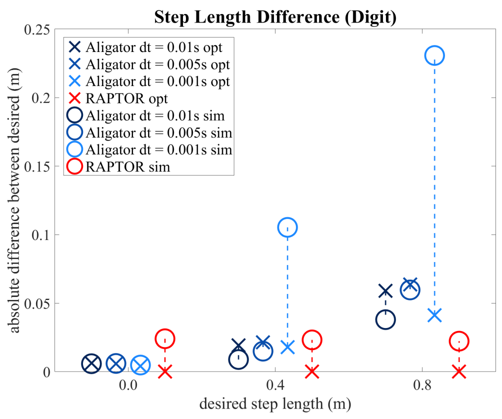

Fig. 4: The absolute value of the difference between the desired step length and the actual step length. The crosses indicate the difference on the optimized trajectory. They may be away from 0 due to optimization not converging to a feasible solution. The circles indicate the difference on the trajectory in simulation when tracking the optimized trajectory using a controller. The circles may not be aligned with the crosses due to imperfect tracking of the controller.

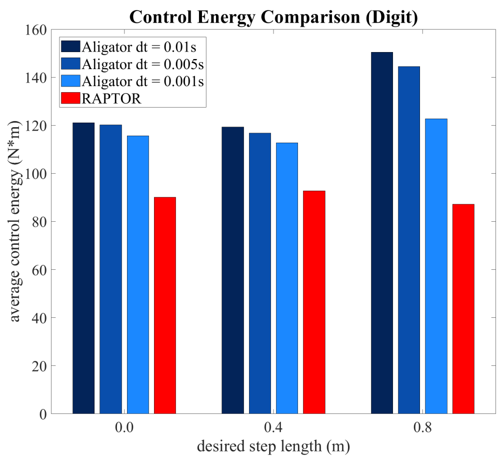

Fig. 5: Control energy consumption of 3 variants of aligator and RAPTOR in the simulation for 3 different desired step lengths

Figure 3 shows how the constraint violation converges with respect to time during optimization. Although aligator converges faster than RAPTOR in general, it struggles to converge to an acceptable solution for more challenging tasks (desired step length $\geq$ 0.4m) or for finer time discretization, while RAPTOR is always able to converge to a feasible solution below the constraint violation tolerance, usually within 1 second.

Figure 4 provides more details on the impacts of converging below the constraint violation tolerance or not. Here we consider the difference between the desired step lengths and the resultant step lengths of both the optimized trajectories and the simulation trajectories when tracking the optimized trajectories using the controller in (23). As we have discussed previously, aligator struggles to generate a feasible solution that reaches the desired step length in the optimization stage, while the solution of RAPTOR can always reach desired step lengths. For simulation trajectories, finer discretization brings better tracking performance for aligator when compared between 0.01s and 0.005s. However, aligator struggles to converge when time discretization is 0.001s, returning unoptimized control inputs that lead to larger tracking error, and as a result performs the worst in both optimization and simulation. For RAPTOR, (23) results in a larger tracking error compared to aligator. The performance is still better than that of aligator for the most challenging task when the desired step length is 0.8m. We can infer that RAPTOR could perform better in simulation if equipped with a more sophisticated controller, which we consider to be one of our future work.

We also consider the control energy consumed in the simulation, which is calculated using $\sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i^T u_i}$, where N is the number of samples recorded in the simulation and u_i is the control input at ith time sample. As shown in Figure 5, RAPTOR is always able to find the solution with the minimum control energy consumption, which is critical for humanoids with embedded mobile batteries.

V. Conclusion

This paper presents a novel trajectory optimization algorithm for full-size humanoids that can generate feasible gaits in seconds. The method outperforms existing state-of-the-art planners in terms of energy efficiency and achieving the desired behavior. Future work will involve transferring the optimized trajectory onto real-world hardware.

REFERENCES

[1] 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. [Online]. Available: https://doi.org/10.1177/0278364913506757.

[2] A. Hereid and A. D. Ames, "Frost: Fast robot optimization and simulation toolkit," IEEE/RSJ, Vancouver, BC, Canada, Sep. 2017.

[3] S. Kajita, M. Morisawa, K. Miura, *et al.*, "Biped walking stabilization based on linear inverted pendulum tracking," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 4489–4496.

[4] K. S. Narkhede, A. M. Kulkarni, D. A. Thanki, and I. Poulakakis, "A sequential mpc approach to reactive planning for bipedal robots using safe corridors in highly cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 831–11 838, 2022.

[5] A. Shamsah, Z. Gu, J. Warnke, S. Hutchinson, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4913–4934, 2023.

[6] Z. Li, J. Zeng, S. Chen, and K. Sreenath, "Autonomous navigation of underactuated bipedal robots in height-constrained environments," *The International Journal of Robotics Research*, vol. 42, no. 8, pp. 565–585, 2023.

[7] J. Liu, P. Zhao, Z. Gan, M. Johnson-Roberson, and R. Vasudevan, "Leveraging the template and anchor framework for safe, online robotic gait design," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 10 869–10 875.

[8] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302.

[9] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2024.

[10] C. Mastalli, R. Budhiraja, W. Merkt, *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 2536–2542.

[11] C. Mastalli, S. P. Chhatoi, T. Corbéres, S. Tonneau, and S. Vijayakumar, "Inverse-dynamics mpc via nullspace resolution," *IEEE Transactions on Robotics*, 2023.

[12] W. Jallet, A. Bambade, E. Arlaud, S. El-Kazdadi, N. Mansard, and J. Carpentier, *PROXDDP: Proximal Constrained Trajectory Optimization*, https://inria.hal.science/hal-04332348v1, 2023.

[13] V. Kurtz, A. Castro, A. Ö. Önol, and H. Lin, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," *arXiv preprint arXiv:2309.01813*, 2023.

[14] P. Zhao, S. Mohan, and R. Vasudevan, "Optimal control of polynomial hybrid systems via convex relaxations," *IEEE Transactions on Automatic Control*, vol. 65, no. 5, pp. 2062–2077, 2019.

[15] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. Van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1241–1246.

[16] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *arXiv preprint arXiv:2401.16889*, 2024.

[17] H. Duan, B. Pandit, M. S. Gadde, *et al.*, *Learning vision-based bipedal locomotion for challenging terrain*, 2023.

[18] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, eadi9579, 2024.

[19] G. Lorentz, *Bernstein Polynomials* (Mathematical expositions). University of Toronto Press, 1953. [Online]. Available: https://books.google.com/books?id=ISjvAAAAMAAJ.

[20] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[21] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[22] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for hzd gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, 2018.

[23] S. A. Burden, H. Gonzalez, R. Vasudevan, R. Bajcsy, and S. S. Sastry, "Metrization and simulation of controlled hybrid systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 9, pp. 2307–2320, 2015.

[24] J. Shurman, *Multivariable Calculus*. Reed College, 2010.

[25] *Online supplementary material for rapid and robust trajectory optimization for humanoids*, https://github.com/roahmlab/RAPTOR/blob/main/Assets/RAPTOR_Supplementary_Materials.pdf, Accessed: 2024-08-28.

[26] Y. Zhao, H.-C. Lin, and M. Tomizuka, "Efficient trajectory optimization for robot motion planning," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 260–265.

[27] B. Zhang, D. Haugk, and R. Vasudevan, "System identification for constrained robots," *arXiv preprint arXiv:2408.08830*, 2024.

[28] J. Carpentier, G. Saurel, G. Buondonno, *et al.*, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[29] P. Robotics. "Talos." (2024), [Online]. Available: https://pal-robotics.com/robot/talos/.

[30] B. Dynamics. "Atlas and beyond: The world's most dynamic robots." (2024), [Online]. Available: https://bostondynamics.com/atlas/.

[31] T. A.
bibinitperiod Robotics. "Tesla bot." (2024), [Online]. Available: https://www.tesla.com/AI.

[32] U. Robotics. "Unitree's first universal humanoid robot." (2024), [Online]. Available: https://www.unitree.com/cn/h1/.

[33] F. Intelligence. "Fourier gr1." (2024), [Online]. Available: https://fourierintelligence.com/gr1/.

[34] A. Robotics, "The next generation of digit - enabling humans to be more human," 2023. [Online]. Available: https://www.youtube.com/watch?v=rnFZAB9ogEE.

[35] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[36] J. Scott *et al.*, "Hsl@ 60: A brief history of the hsl mathematical software library," STFC, Tech. Rep., 2023.

[37] B. Gough, *GNU scientific library reference manual*. Network Theory Ltd., 2009.

[38] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.