Shooting for Contact: Contact-Implicit Multiple Shooting for Dynamic Motion Retargeting

Sergio A. Esteban[1], Jason H.K. Siu[1], Derrick Mach[1], Junheng Li[1],
Vince Kurtz[2], Joel W. Burdick[1], and Aaron D. Ames[1]

***Abstract*— Motion retargeting approaches often prioritize kinematic similarity over whole-body dynamics, contact consistency, and actuation limits, yielding references that are difficult for reinforcement learning (RL) policies to reproduce, particularly for contact-rich behaviors. We present a contact-implicit, direct simulation-based multiple shooting (DSMS) framework that transforms kinematically feasible references into dynamically feasible whole-body trajectories. By embedding a differentiable simulator within a nonlinear program, DSMS resolves contact, friction, impacts, self-collision, and joint limits internally while enforcing tracking, actuation, and task constraints *without* prescribing a contact schedule or introducing explicit contact constraints. Compared with existing retargeting methods, DSMS accelerates motion-imitation RL training and yields policies with high success rates and low tracking error. We further demonstrate zero-shot sim-to-real transfer on the Unitree G1 through command-conditioned contact-rich crawling and a highly dynamic 180-degree jump-turn. Additional Material: `shooting-for-contact.github.io`.**

I. Introduction

Reference motions derived from human demonstrations, reduced-order models, and animation provide powerful priors for learning and control [1]. Recent reference-tracking reinforcement learning (RL) and imitation learning (IL) have demonstrated complex locomotion and whole-body behaviors from motion-capture, animation, and video-generated trajectories [2]–[4]. However, the quality of the learned controller depends strongly on the reference motion [5], [6]. Most retargeting pipelines prioritize kinematic correspondence, mapping human pose or key-body poses onto the robot while accounting for differences in morphology [7], [8]. Although this method can produce visually plausible motions, the resulting trajectories are often not dynamically consistent with robot physics, contacts, or actuation limits—limiting their applicability across diverse motions.

This dynamics mismatch is particularly problematic for contact-rich behaviors, such as crawling, parkour jumping, and rolling, during which the robot may use different body parts for support and balance [9]. A kinematically retargeted motion may contain foot sliding, unrealistic ground penetrations, self-collisions, or actions that require infeasible torques and impulses. RL can sometimes compensate for and robustify these defects, but it must then simultaneously discover a feasible realization of the motion and learn to stabilize it. These characteristics make reward tuning difficult: strong tracking rewards pull the policy toward an infeasible reference, while weaker rewards allow the learned behavior to deviate from the intended motion.

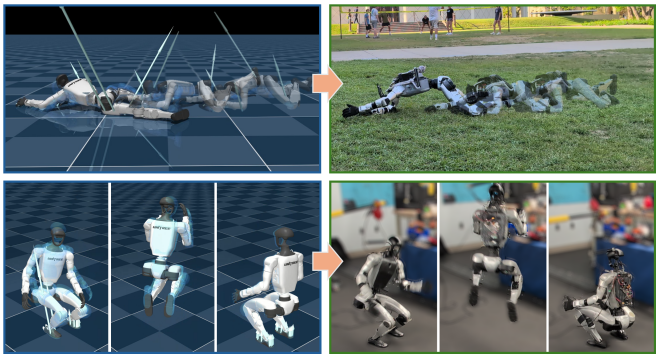


Fig. 1: DSMS trajectory optimization solutions in simulation (left) and their corresponding zero-shot hardware realizations on the Unitree G1 (right), shown for velocity-conditioned contact-rich crawling (top) and a 180° jump-turn (bottom).

We propose a method for generating *dynamically feasible* references for RL. Before policy training, trajectory optimization transforms the original motion into a dynamically consistent reference for the target robot. For contact-rich behaviors, this optimization must also reason over contacts that may not be known *a priori*, as well as handling hard constraints such as friction cone and actuation limits. The resulting policy can then focus on simply stabilizing around a physically meaningful nominal trajectory rather than discovering a trajectory from a dynamically inconsistent reference.

A. Related Works

Generating meaningful robot references is challenging because contact-rich humanoid motion is high-dimensional and hybrid. Recent reference-based RL has enabled humanoid robots to reproduce diverse locomotion and whole-body behaviors from human motion data [2], [3], [10]. However, such motions are largely generated through kinematic retargeting and geometric correspondence [11]. Other methods improve this stage by allowing interaction-preserving constraints or learned motion refinement [5], [12], [13]. Nevertheless, morphology and actuation differences can produce artifacts such as foot sliding and penetrations. Retargeting quality has consequently been shown to strongly affect the robustness and convergence of downstream tracking policies [12], [14].

[1]The authors are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA. {sesteban,jasonsiu,dmach,junhengl,ames}@caltech.edu and jwb@robotics.caltech.edu.

[2]The author is with the School of Computing, DePaul University, Chicago, IL, USA. vkurtz1@depaul.edu.

*This work was supported by Technology Innovation Institute (TII).

Several works explicitly improve the physical feasibility of reference motions. Bi-level motion imitation alternates between adapting the reference and optimizing the tracking policy [15]. OmniTrack uses privileged policy rollouts to produce physics-consistent references [16]. DynaRetarget [17] and DDR [18] employ shooting-based trajectory optimization or sampling-based model-predictive control (MPC) within simulations.

Nonlinear programs based on direct collocation [19] and multiple shooting [20] have shown promise for stabilizing highly dynamic and underactuated systems. However, in their standard forms, these transcriptions do not resolve contact implicitly and instead require prescribed contact modes or explicit contact constraints.

Contact-implicit trajectory optimization (CITO) provides an appealing basis for ensuring dynamic feasibility [21], [22]. The optimizer is free to make or break additional contacts as needed to achieve the task, ensuring full dynamic feasibility while staying close to a reference motion. However, many CITO methods require explicit contact pairs, complementarity relaxations, or contact-force variables [1], [23]. Additionally, many CITO methods aim primarily at MPC, where full convergence and tight constraints are compromised for speed [23], [24]. CITO commonly represents unilateral contact through complementarity constraints and smooth contact models [25]–[27]. Related shooting-based methods require differentiable simulations for whole-body MPC, while derivative-free methods such as predictive sampling provide a simple alternative for contact-rich behavior synthesis [28], [29]. More broadly, differentiable simulators obtain useful sensitivities through regularized contact, implicit differentiation of hard-contact solves, or analytical differentiation of collision and friction [30]–[33].

B. Contributions

This work introduces a **direct simulation-based multi-shooting (DSMS)** nonlinear program for dynamics-aware retargeting of contact-rich humanoid motions. We use the discrete transition map of a differentiable simulator as the whole-body dynamics model, where contact reasoning, friction, self-collision, and actuation limits are internally handled by the high-fidelity simulator, eliminating the need for a traditional contact-implicit optimization problem setup.

The main contributions of the paper are as follows:

- A general direct simulation-based multi-shooting framework for whole-body dynamics and contact-implicit motion retargeting.
- A unified pipeline that transforms kinematically feasible trajectories into whole-body dynamically feasible references and that synthesizes command-conditioned periodic gait libraries from a few short motion clips.
- For motion-imitation RL, a numerical study of how references generated with reduced-order, kinodynamic, and our full-order optimization affect downstream policy training and tracking performance.
- Hardware demonstrations of (1) twist command-conditioned contact-rich humanoid crawling under constrained spaces and rough terrain, and (2) a highly dynamic 180° jump-turn motion.

II. Preliminaries

A. Whole-Body Dynamics

We consider legged robots that achieve locomotion by intermittently making and breaking contact with their environment. In particular, we are interested in contact-rich behaviors in which the robot may use *any* part of its body to generate motion.

We define the state as $\mathbf{x} := (\mathbf{q}, \mathbf{v}) \in \mathsf{T}\mathcal{Q}$, where $\mathbf{q} \in \mathcal{Q} \subseteq \mathbb{R}^{n_q}$ denotes the generalized configuration and $\mathbf{v} \in \mathsf{T}_{\mathbf{q}}\mathcal{Q} \subseteq \mathbb{R}^{n_v}$ the generalized velocity. The whole-body (WB) continuous-time dynamics can be written as

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{H}(\mathbf{q}, \mathbf{v}) = \mathbf{B}\boldsymbol{\tau} + \mathbf{J}_c^\top(\mathbf{q})\boldsymbol{\lambda}, \tag{1}$$

where $\mathbf{M} : \mathcal{Q} \to \mathbb{R}^{n_v \times n_v}$ is the mass–inertia matrix, $\mathbf{H} : \mathsf{T}\mathcal{Q} \to \mathbb{R}^{n_v}$ collects Coriolis, centrifugal, and gravitational terms, $\mathbf{B} \in \mathbb{R}^{n_v \times n_u}$ maps actuator torques $\boldsymbol{\tau} \in \mathbb{R}^{n_u}$ into generalized coordinates, and $\boldsymbol{\lambda} \in \mathbb{R}^{n_c}$ denotes the stacked contact-force coordinates acting through the contact Jacobian $\mathbf{J}_c : \mathcal{Q} \to \mathbb{R}^{n_c \times n_v}$. Here, n_c is the total number of scalar contact-force coordinates associated with all active contacts.

B. Differentiable Contact Dynamics

Rigid-body dynamics simulators compute contact forces $\boldsymbol{\lambda}$ and advance the system dynamics (1) with a variety of methods, including compliant models [30], complementarity problems [34], convex optimization [35], and more [36]. Most simulators provide a discrete-time forward map

$$\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i). \tag{2}$$

where $\mathbf{x}_i$ and $\mathbf{u}_i$ are the state and actuator command at the i^{th} simulation time step. A selected actuator interface maps $\mathbf{u}_i$ to the joint torque $\boldsymbol{\tau}_i$ applied in (1), as detailed in Sec. III-A. Depending on the simulator, this map may also capture sticking and sliding contacts, impacts, joint-limit activation, and other constraints within a single discrete transition.

We require that these forward dynamics are differentiable, that is, the partial derivatives

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}, \quad \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \tag{3}$$

are available. In this work, we use MuJoCo [30], which leverages a convex differentiable contact formulation [37] to provide these gradients via finite differences. However, our framework is compatible with any differentiable simulator.

III. Trajectory Optimization

Our goal is to synthesize dynamically feasible whole-body trajectories that track a desired reference. Importantly, contact is not modeled *explicitly* in the optimization problem: no contact schedule or complementarity constraints are prescribed. Instead, contact behavior is resolved *implicitly* through the simulator dynamics (2).

The tracking terms are defined relative to a reference trajectory $\mathbf{X}^{\mathrm{ref}} := (\mathbf{x}_0^{\mathrm{ref}}, \ldots, \mathbf{x}_N^{\mathrm{ref}})$ that need not satisfy the

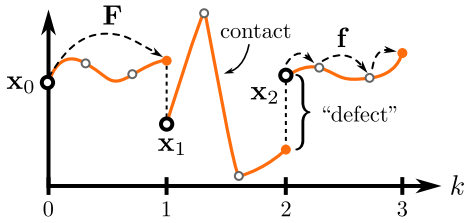


Fig. 2: Proposed multi-shooting trajectory optimization.

whole-body dynamics (1). The reference may come from retargeted human motion-capture data, animation data, or a reduced-order model (ROM), whose trajectories are dynamically consistent only within the reduced representation.

A. Direct Simulation-based Multiple Shooting (DSMS)

We employ direct multiple shooting to transcribe the trajectory optimization problem into a finite-dimensional nonlinear program (NLP). This transcription divides the horizon into N intervals, treats the shooting-node states as decision variables, and enforces continuity between independently integrated intervals through defect constraints. Shorter rollouts and intermediate states reduce long-horizon sensitivity and improve numerical robustness and convergence [38]. This is particularly valuable for highly dynamic, underactuated motions, whose open-loop behavior can be sensitive to early control decisions.

We discretize the horizon $[0, T]$ into N shooting intervals on the uniform grid

$$t_k \coloneqq k\,\Delta t, \qquad k = 0, \dots, N, \qquad \Delta t \coloneqq \frac{T}{N}.$$

Let $\mathbf{X} \coloneqq (\mathbf{x}_0, \dots, \mathbf{x}_N)$ collect the state variables at the shooting nodes and let $\mathbf{U} \coloneqq (\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_M)$ collect M control points, with $\boldsymbol{\xi}_j \in \mathbb{R}^{n_u}$. These control points parameterize the time-varying command signal through the map $\mathbf{s} : [0, T] \times \mathbb{R}^{Mn_u} \to \mathbb{R}^{n_u}$, defined by

$$\mathbf{u}(t) \coloneqq \mathbf{s}(t, \mathbf{U}) = \sum_{j=1}^{M} \beta_j(t)\boldsymbol{\xi}_j. \tag{4}$$

Here, $\{\beta_j\}_{j=1}^{M}$ is a prescribed set of basis functions, such as a zero-order hold or piecewise-linear basis. The signal $\mathbf{u}(t)$ denotes the actuator command, whereas $\boldsymbol{\tau}(t)$ denotes the joint torque entering (1). Depending on the actuator interface, $\mathbf{u}(t)$ represents either a direct torque command or a desired joint position tracked by a low-level PD controller.[1] At shooting node k, the command is $\mathbf{u}_k \coloneqq \mathbf{u}(t_k)$. Using shooting-node states $\mathbf{X}$ and spline control points $\mathbf{U}$, we transcribe the optimal control problem via direct multiple shooting as an NLP:

$$\min_{\mathbf{X},\mathbf{U}} \quad J(\mathbf{X}, \mathbf{U}) \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k), \qquad k = 0, \dots, N-1, \tag{5b}$$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \le \mathbf{0}, \qquad k = 0, \dots, N, \tag{5c}$$

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \qquad k = 0, \dots, N. \tag{5d}$$

[1]Under torque control, the applied torque is $\boldsymbol{\tau}(t) = \mathbf{u}(t)$ and under position control, the applied torque is $\boldsymbol{\tau}(t) = \mathbf{K}_p\big(\mathbf{u}(t) - \mathbf{q}^{\mathrm{int}}(t)\big) - \mathbf{K}_d\mathbf{v}^{\mathrm{int}}(t)$.

Here, the objective (5a) encodes tracking and regularization, while the path limits and boundary or task constraints are collected in (5c) and (5d). The flow map $\mathbf{F}$ in (5b) integrates (2) from $\mathbf{x}_k$ under $\mathbf{u}(t)$ during $[t_k, t_{k+1})$, with the *defect* enforcing continuity between shooting intervals. Internally, $\mathbf{F}$ chains S substeps of $\mathbf{f}$ with $\Delta t_{\mathrm{sim}} = \Delta t/S$. Fine simulator steps resolve stiff contact dynamics, while state variables only at the coarser shooting nodes keep the NLP small. This is illustrated in Fig. 2. Because the simulator directly defines the flow between shooting nodes, we refer to this formulation as *direct simulation-based multiple shooting (DSMS)*.

B. Cost Function

The cost $J(\mathbf{X}, \mathbf{U})$ in (5a) is a weighted sum of state and body tracking, together with input regularizers penalizing torque effort and command rate.

With state error $\mathbf{e}_k \coloneqq \mathbf{x}_k \ominus \mathbf{x}_k^{\mathrm{ref}}$, the state tracking term[2] penalizes the tangent space deviation from the reference,

$$\ell_{\mathbf{x}}(\mathbf{x}_k, k) = \mathbf{e}_k^{\top}\mathbf{Q}_{\mathbf{x}}\,\mathbf{e}_k, \tag{6}$$

with diagonal $\mathbf{Q}_{\mathbf{x}} \succeq \mathbf{0}$ weighting the configuration and velocity errors.

In the style of the motion tracking objectives used in [2], [3], we track the pose and twist of a set of key bodies $\mathcal{B}$, such as the feet, hands, and pelvis. For body $b \in \mathcal{B}$, let $\mathbf{y}_k^b$ stack its pose and twist,

$$\mathbf{y}_k^b \coloneqq \big(\underbrace{\mathbf{p}_k^b,\ \boldsymbol{q}_k^b}_{\text{pose}},\ \underbrace{\mathbf{v}_k^b,\ \boldsymbol{\omega}_k^b}_{\text{twist}}\big), \tag{7}$$

where these quantities are obtained by forward kinematics from $\mathbf{x}_k$, and let $\mathbf{y}_k^{b,\mathrm{ref}}$ denote the same quantities on $\mathbf{x}_k^{\mathrm{ref}}$. With body error $\mathbf{e}_k^b \coloneqq \mathbf{y}_k^b \ominus \mathbf{y}_k^{b,\mathrm{ref}}$, the body tracking term is

$$\ell_{\mathbf{y}}(\mathbf{x}_k, k) = \sum_{b \in \mathcal{B}} \big(\mathbf{e}_k^b\big)^{\top}\mathbf{Q}_{\mathbf{y}}\mathbf{e}_k^b, \tag{8}$$

with diagonal $\mathbf{Q}_{\mathbf{y}} \succeq \mathbf{0}$ weighting each body's pose and twist errors. Each body is tracked in either the world frame or the frame of an anchor body such as the torso.

The terminal state and body tracking terms retain the same form but with their weights scaled by $\gamma > 1$:

$$\ell_{\mathbf{x}}^f(\mathbf{x}_N) = \gamma\,\ell_{\mathbf{x}}(\mathbf{x}_N, N), \tag{9a}$$

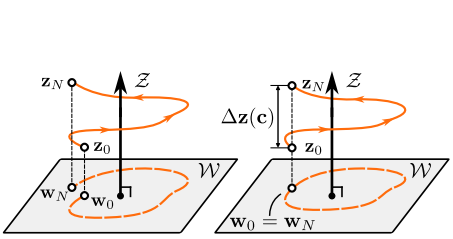
$$\ell_{\mathbf{y}}^f(\mathbf{x}_N) = \gamma\,\ell_{\mathbf{y}}(\mathbf{x}_N, N). \tag{9b}$$

Because the state and body errors are defined in different spaces—a generalized state space and a body-output space—the two terms are complementary: body tracking enforces task-relevant motion, while state tracking resolves kinematic redundancy and regularizes the whole-body posture.

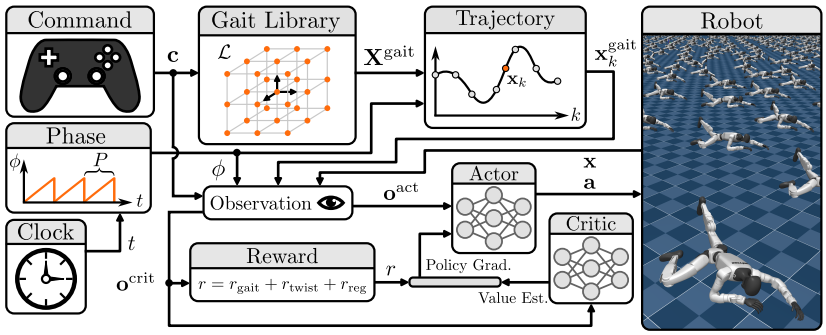
We regularize actuation with two terms. The first penalizes torque effort on the realized torque $\boldsymbol{\tau}_k$,

$$\ell_{\boldsymbol{\tau}}(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\tau}_k^{\top}\mathbf{R}_{\boldsymbol{\tau}}\,\boldsymbol{\tau}_k, \tag{10}$$

[2]Subtraction on a manifold is denoted by $\ominus$, accounting for the quaternion component of the state.



(a) Limit cycle closure.



(b) Learning architecture.

Fig. 3: (a) A gait reference is closed into a command-constrained limit cycle: the planar pose advances by $\Delta\mathbf{z}(\mathbf{c})$, while $\mathbf{w}_0 = \mathbf{w}_N$ enforces limit-cycle closure. (b) The commanded twist selects a phase-aligned gait from the library, and an asymmetric actor–critic trains a reference-free policy from proprioception, command, and phase.

with diagonal $\mathbf{R}_{\boldsymbol{\tau}} \succeq \mathbf{0}$. We penalize the realized torque rather than the raw command $\mathbf{u}_k$ because the meaning of $\mathbf{u}_k$ depends on the actuator interface. The second term penalizes the command difference $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$, promoting smooth commands and suppressing chatter,

$$\ell_{\Delta\mathbf{u}}(\mathbf{U}) = \sum_{k=1}^{N-1} \Delta\mathbf{u}_k^\top \mathbf{R}_{\Delta\mathbf{u}}\, \Delta\mathbf{u}_k, \tag{11}$$

where diagonal $\mathbf{R}_{\Delta\mathbf{u}} \succeq \mathbf{0}$ and $\mathbf{u}_k = \mathbf{s}(t_k, \mathbf{U})$.

Collecting the tracking and regularization terms, the objective $J(\mathbf{X}, \mathbf{U})$ in (5a) is

$$\begin{aligned} J(\mathbf{X}, \mathbf{U}) = \Delta t \sum_{k=0}^{N-1} \Big[\ell_{\mathbf{x}}(\mathbf{x}_k, k) + \ell_{\mathbf{y}}(\mathbf{x}_k, k) + \ell_{\boldsymbol{\tau}}(\mathbf{x}_k, \mathbf{u}_k)\Big] \\ + \ell^f_{\mathbf{x}}(\mathbf{x}_N) + \ell^f_{\mathbf{y}}(\mathbf{x}_N) + \Delta t\, \ell_{\Delta\mathbf{u}}(\mathbf{U}), \end{aligned} \tag{12}$$

where the running terms are scaled by the timestep Δt so that their sum approximates the integral cost.

C. Receding Horizon Control

Full-horizon trajectory optimization solves the complete reference trajectory from a fixed initial state in a single solve. For highly dynamic humanoid motions, such as backflips and forward rolls, we instead apply (5) in a receding-horizon fashion, using repeated replanning to stabilize the rollout. Each solve spans a moving reference window of H intervals and duration $T_H = H\Delta t$, shorter than the full trajectory.

At the j-th replanning time $\bar{t}_j$, we initialize the horizon from the current simulated state and shift the reference accordingly:

$$\mathbf{x}_0 = \mathbf{x}(\bar{t}_j), \qquad \mathbf{x}_k^{\text{ref}} = \mathbf{x}^{\text{ref}}(\bar{t}_j + k\Delta t), \tag{13}$$

for $k = 0, \ldots, H$. We then solve the corresponding H-interval instance of (5), execute the first Δt_{ctrl} seconds of the optimized command, advance the simulator, and repeat with the shifted window. Concatenating the executed segments produces a long rollout whose transitions are generated directly by the simulator and therefore satisfy (1) by construction, rather than only up to the defect tolerance in (5b). Example motions are shown in Fig. 5.

D. Gait Synthesis

To demonstrate that the proposed optimization can enforce constraints while preserving dynamic feasibility in contact-rich motion, we use human motion-capture data of crawling. Crawling involves frequent contacts with the hands, elbows, knees, and feet, including sliding contacts. We synthesize a library of periodic gaits, each realizing a prescribed average base twist $\mathbf{c} := (v^x, v^y, \omega^z) \in \mathbb{R}^3$. To cover the command space, we use three motion-capture references: a forward crawl, a backward crawl, and an in-place turn, together with a static, zero-twist prone idle reference. Their corresponding twist grids tile the full command space.

Starting from a nearly periodic gait reference with period P, we solve the NLP (5) over a single gait cycle, using the equality constraints (5d) to enforce limit cycle closure. To formulate these constraints, we split the shooting-node state as $\mathbf{x}_k = (\mathbf{z}_k,\ \mathbf{w}_k) \in \mathcal{Z} \times \mathcal{W}$. The first component is the planar base pose,

$$\mathbf{z}_k := \left(\mathbf{p}_k^{xy},\ \psi_k\right) \in \mathcal{Z}, \tag{14}$$

containing the base horizontal position $\mathbf{p}_k^{xy}$ and heading ψ_k, with $\mathcal{Z} \cong SE(2)$. The second component is the remaining state,

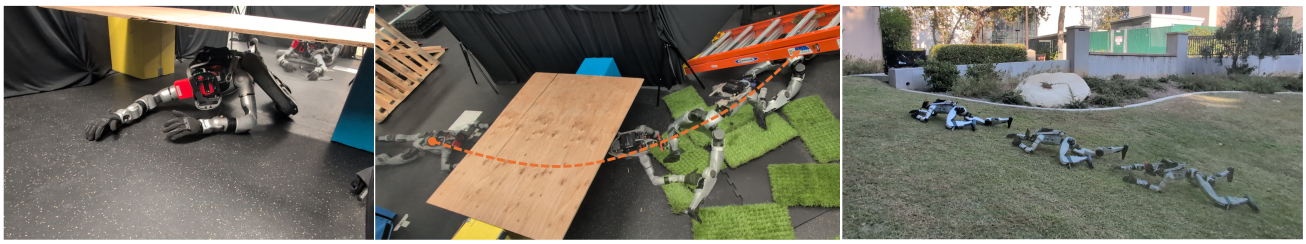
$$\mathbf{w}_k := \left(p_k^z,\ \boldsymbol{\theta}_k,\ \mathbf{q}_k^{\text{jnt}},\ \mathbf{v}_k\right) \in \mathcal{W}, \tag{15}$$

containing the base height p_k^z, yaw-invariant tilt $\boldsymbol{\theta}_k$, joint positions $\mathbf{q}_k^{\text{jnt}}$, and generalized velocities $\mathbf{v}_k$.

The planar pose advances by the commanded displacement, $\mathbf{z}_N = \mathbf{z}_0 \oplus \Delta\mathbf{z}(\mathbf{c})$,[3] which enforces a prescribed average base twist $\mathbf{c}$. The remaining state satisfies $\mathbf{w}_0 = \mathbf{w}_N$, returning the height, tilt, joints, and velocities to their initial values and enforcing periodicity. Together, these constraints yield the limit cycle shown in Fig. 3a. Because closure determines the cycle only up to a planar rigid transformation, the tracking costs (6) and (8) anchor it to the gait reference, fully determining the remaining degrees of freedom.

In addition to the cost (12), we add a no-slip term that penalizes the horizontal velocity of each contact link while it is planted. Because this term is a soft penalty rather than

[3]Addition on a manifold is denoted by $\oplus$ and accounts for the $SE(2)$ component of the state.



(a) (b) (c)

Fig. 4: Experimental demonstrations of (a) crawling forward and backward under height-constrained space, (b) crawling under a height-constrained course with rough patches, and (c) crawling uphill on grass.

a hard constraint, it discourages unnecessary slipping while permitting the sliding contacts required by crawling, without excluding dynamically feasible solutions.

Repeating this procedure for a range of commanded twists yields a library of command–gait pairs, in which each gait trajectory $\mathbf{X}_i^{\text{gait}}$ is dynamically feasible under its corresponding command $\mathbf{c}_i$:

$$\mathcal{L} = \{(\mathbf{c}_i, \mathbf{X}_i^{\text{gait}})\}_{i=1}^{N_{\text{gait}}}. \tag{16}$$

IV. Reinforcement Learning

A. Motion Imitation

We use a motion-imitation framework to RL train policies that reproduce generated reference motions [2]. Given a reference trajectory from Sec. III-C, the policy learns to reproduce it using per-frame body pose and twist rewards, reference-state initialization, and adaptive frame sampling.

We use simulation deployment to evaluate how readily and accurately each reference can be learned and reproduced: a policy is trained on each reference and evaluated by how reliably and accurately it reproduces the motion during simulation rollouts. We use these results to compare dynamic fidelity and retargeting methods in Sec. V-D.

B. Locomotion Controller

We train a single command-conditioned RL policy to track the gait library $\mathcal{L}$. Given a commanded twist $\mathbf{c}$, the nearest library trajectory provides the imitation target, allowing the discrete library to serve as a continuous velocity-command interface, as shown in Fig. 3b.

We use an asymmetric actor-critic with a reference-free actor that observes signals available at deployment:

$$\mathbf{o}^{\text{act}} = \big(\hat{\mathbf{g}},\ \boldsymbol{\omega}^{\text{base}},\ \mathbf{q}^{\text{jnt}},\ \mathbf{v}^{\text{jnt}},\ \mathbf{a}^{\text{prev}}, \mathbf{c},\ \sin(\phi),\ \cos(\phi)\big). \tag{17}$$

These comprise projected gravity $\hat{\mathbf{g}}$, base angular velocity $\boldsymbol{\omega}^{\text{base}}$, joint positions and velocities $\mathbf{q}^{\text{jnt}}, \mathbf{v}^{\text{jnt}}$, previous action $\mathbf{a}^{\text{prev}}$, commanded twist $\mathbf{c}$, and a sinusoidal encoding of the gait phase ϕ. During training, the critic additionally observes the base linear velocity and gait reference:

$$\mathbf{o}^{\text{crit}} = \big(\mathbf{o}^{\text{act}},\ \mathbf{v}^{\text{base}},\ \mathbf{x}^{\text{gait}}(\mathbf{c}, \phi)\big). \tag{18}$$

Here, $\mathbf{x}^{\text{gait}}(\mathbf{c}, \phi)$ is the reference state for command $\mathbf{c}$ at phase ϕ. The actor therefore requires only proprioception, command, and phase at inference.

The action $\mathbf{a}$ specifies joint-position offsets applied through a PD controller. We use the prone idle posture as the nominal pose, so zero action is valid and the policy learns only residual corrections.

The reward consists of gait tracking, command tracking, and regularization:

$$r = r_{\text{gait}} + r_{\text{twist}} + r_{\text{reg}}. \tag{19}$$

Following [2], r_{gait} tracks key-body poses and velocities, together with the anchor position and heading. The anchor target is propagated egocentrically by the reference twist and re-based at each command change to preserve translation and turning across gait cycles. Finally, r_{twist} tracks the commanded base twist, while r_{reg} penalizes action rate, joint-limit violations, and self-collisions.

All trajectories in $\mathcal{L}$ share period P, so a phase clock ϕ indexes the current imitation target. During an episode, resampling $\mathbf{c}$ selects the nearest gait without resetting the phase or robot state, while $\mathbf{c} = \mathbf{0}$ selects the idle reference. At each switch, we blend the outgoing and incoming references at the shared phase while updating the command immediately, enabling smooth transitions.

V. Results

A. Implementation Details

We solve (5) using `cyipopt`, a Python interface to the interior-point optimizer IPOPT [39]. At each iteration, the resulting sparse KKT system is factorized with the HSL `ma57` solver , which exploits the sparsity of the multiple shooting Jacobian to reduce computational and memory costs. Because MuJoCo does not provide the second-order dynamics derivatives needed to build the Lagrangian's second-order terms,[4] we use the L-BFGS approximation to construct curvature information from first-order gradients (3). Despite the stiffness of the contact dynamics and the abrupt curvature changes induced by contact creation and separation, this approximation performs surprisingly well in practice.

We train RL policies for the Unitree G1 in mjlab [40] with reference clips from the BONES-SEED human motion dataset [41]. Policies are optimized with PPO via `rsl_rl`, using $(512, 256, 128)$ actor and critic MLPs. Simulated dynamics run at 200 Hz and the policy at 50 Hz, with 10 second episodes. For sim-to-real transfer, we randomize base pushes, center-of-mass offsets, joint-encoder biases, and contact friction.

[4]Second-order derivatives are available through automatic differentiation in MJX, MuJoCo's JAX-based reimplementation.

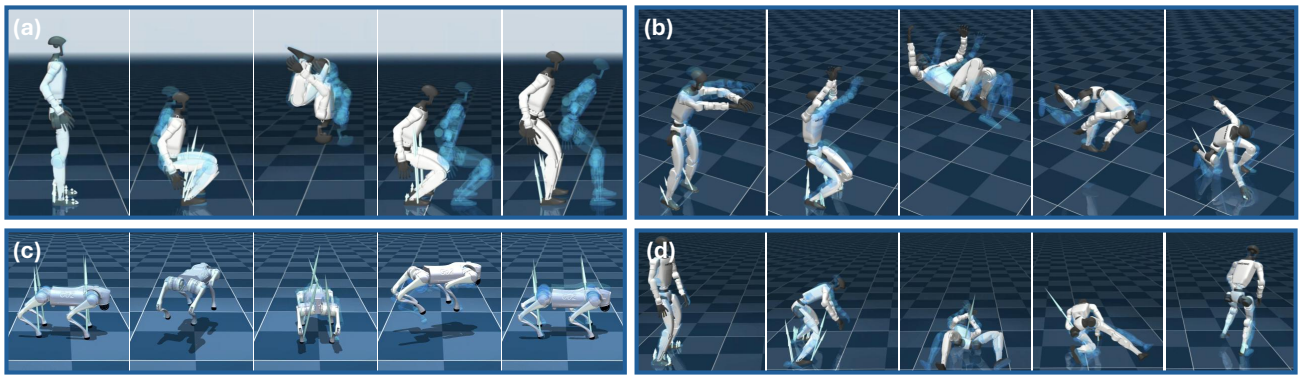


Fig. 5: Simulation snapshots for dynamic contact-rich whole-body maneuvers: (a) humanoid backflip, (b) humanoid super hero backflip, (c) quadruped jump-turn, and (d) humanoid side-rolling. Blue ghost visualizes dynamically infeasible references.

B. Hardware Deployment

Because each optimized reference is dynamically feasible by construction, the policies are trained on physically achievable imitation targets rather than purely kinematically retargeted motions. The resulting policies transfer zero-shot to the Unitree G1 hardware without real-world fine-tuning.

We first deploy a motion imitation policy trained to reproduce a $180°$ jump-turn reference from a single rigid-body (SRB) trajectory (Sec. V-D.1). The controller executes this highly dynamic maneuver on hardware, rotating the robot by approximately $180°$ during the jump and recovering to a stable landing (Fig. 1). We next deploy the RL crawling controller (Sec. IV-B) to evaluate contact-rich locomotion in the real world. The robot is steered in real time through commanded twists. Indoors, it crawls forward and backward through a height-constrained space and along a height-constrained course over rough patches, while outdoors, it crawls uphill on grass (Fig. 4).

C. Locomotion Velocity Tracking

We evaluate velocity tracking in simulation by sampling piecewise-constant twist commands and holding each command for three gait cycles. Fig. 6 compares the commanded twist with the instantaneous (pelvis) velocity and its mean over each command interval. Although the instantaneous velocities exhibit large phase-dependent oscillations due to the crawling gait, their cycle-averaged values follow the commanded trends across v^x, v^y, and ω^z, as intended.

D. Ablations and Method Comparison

We perform an ablation study and directly compare against existing methods to assess both our method's ability to produce trajectories advantageous for motion-tracking RL training and the performance of the resulting policies. All policies are trained across five random seeds using default mjlab motion-tracking settings [42], except that the termination gates and domain-randomization ranges are adjusted to match those used for hardware deployment. Policy evaluation is performed in a separate MuJoCo simulator featuring asynchronous control and sensing as well as randomized starting joint configurations. We report both RL training performance and sim-to-sim base pose tracking error[5] and joint tracking error.

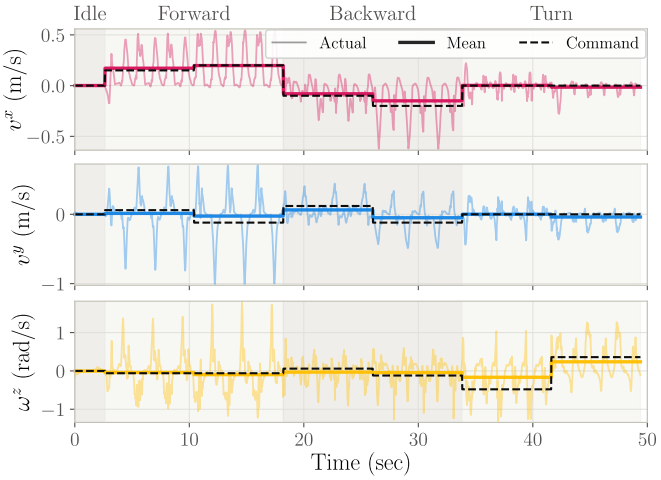


Fig. 6: Commanded (dashed), achieved body-frame instantaneous velocity (actual), and average (mean) velocities as the crawling policy tracks forward, backward, and turning commands in one continuous run.

1) Dynamic Fidelity Ablation: This ablation studies DSMS efficacy when applied to trajectories generated using ROMs of increasing levels of dynamic fidelity. We begin with a backflip motion generated with an SRB ROM. Per-frame inverse kinematics (IK) then provides a fully kinematically feasible trajectory serving as a baseline for this ablation. A kino-dynamic (KD) optimization is performed on this trajectory to further refine the motion [43]. Finally, our method can be applied at any stage to provide full dynamic feasibility, before all trajectories are passed to RL training for evaluation. The trajectory generated by SRB $\rightarrow$ **DSMS** is shown in the backflip sequence of Fig. 5a.

Fig. 7 and Table I report the policy training performance for each trajectory and the tracking accuracy of the resulting policies. The two trajectories retargeted with DSMS converge fastest—indicated by lower PPO action noise σ—and achieve

[5]Positional error component of the pose is normalized by the robot's height and the orientation error component by $180°$. Achieving low pose error is challenging due to the lack of direct control of the robot's underactuated base and thus, makes for an interesting metric of study.

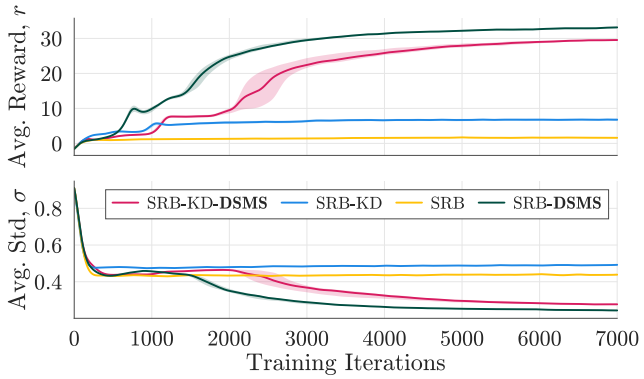


Fig. 7: Training performance of motions generated from the ROM ablation, with variance (shade) across seeds.

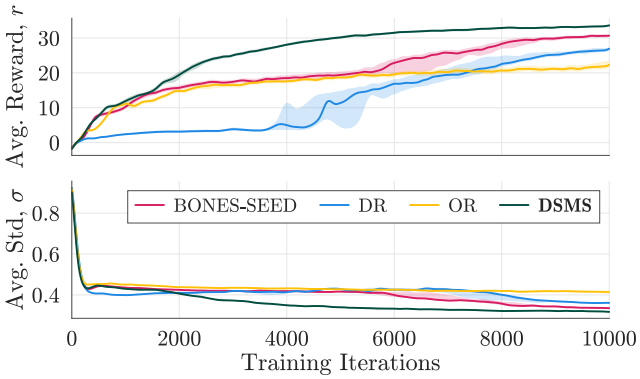


Fig. 8: Training performance of super hero backflip motion across various retargeting methods.

TABLE I: ROM ablation sim-to-sim tracking accuracy.

Method	Landed	Success	Joint Err. (rad)†	Pose Err.†
SRB	0/75	0%	–	–
SRB→KD	0/75	0%	–	–
SRB→**DSMS**	**75/75**	**100%**	$\mathbf{0.110 \pm 0.006}$	$\mathbf{0.155 \pm 0.041}$
SRB→KD→**DSMS**	72/75	96.0%	0.144 ± 0.016	0.253 ± 0.139

σ denotes the PPO policy's exploration noise scale; lower σ indicates greater policy convergence.

TABLE II: Retargeting method tracking error comparison.

Method	Landed	Success	Joint Err. (rad)†	Pose Err.†
OR	7/75	9.3%	$\mathbf{0.157 \pm 0.001}$	$\mathbf{0.095 \pm 0.014}$
BS	60/75	80.0%	0.195 ± 0.003	0.146 ± 0.027
DR	**74/75**	**98.7%**	0.237 ± 0.006	0.160 ± 0.077
DSMS	**74/75**	**98.7%**	$\mathbf{0.159 \pm 0.006}$	$\mathbf{0.102 \pm 0.046}$

† Errors only computed over successful landings.

the highest reward. Sim-to-sim deployment supports that, under standard RL with minimal tuning, policies trained on dynamically-feasible motions outperform those trained on kinematic-only references.

2) Retargeting Method Comparison: We compare against popular retargeting and optimization methods using a "super hero backflip" (Fig. 5b) from BONES-SEED (BS) as a shared reference motion. This flip features contact across the arms, knees, and feet upon landing, as well as artifacts such as unrealistic knee clipping into the ground. The BS baseline is pre-retargeted to the G1, then dynamically retargeted with DSMS and also retargeted with OmniRetarget (OR) and DynaRetarget (DR) [5], [17]. Each resulting trajectory is then used to train motion-tracking policies for evaluation, as discussed previously.

As shown by the training curves (Fig. 8), the trajectory optimized with DSMS again converges fastest, reaching an average reward of 20 in ∼2000 iterations versus the ∼5000 required by its closest competitors—a 40-minute wall-clock difference on an Nvidia RTX 4090. Given our method's ∼12-minute optimization time, this advantage is worthwhile even before accounting for the runtime of competing methods. The combined sim-to-sim tracking metrics (Table II) tell the same story: DSMS attains the highest backflip success rate, near-lowest tracking error, making it the only method to consistently perform the hero backflip with accurate tracking.

TABLE III: Qualitative feature comparison across methods.

Attribute	OmniRetarget	DynaRetarget	SPARK	Ours
Kinematic Feas.	✓	✓	✓	✓
WB Dyn. Feas.	✗	✓	✗	✓
Contact-Implicit	✓	✓	✗	✓
Arbitrary Const.	✗	✗	✓	✓
Solver	Seq. SOCP	SBTO	Unspecified	IPOPT

E. Qualitative Capability Comparison

To augment our discussion of retargeting and trajectory optimization approaches and situate DSMS among similar approaches, we present Table III.

Here, whole-body dynamic feasibility refers to a trajectory satisfying (1), while arbitrary constraints refers to a method's ability to enforce constraints of the form of (5c) and (5d). Although all four methods provide kinematic feasibility, they differ substantially in the physical and constraint-handling ability. OmniRetarget achieves excellent kinematic retargeting and often produces visually plausible trajectories, but it sacrifices dynamic feasibility for kinematic accuracy. SPARK shares this limitation, and although it can incorporate arbitrary constraints, it is not contact-implicit: it relies on holonomic constraints that require accurate contact detection for the solver to function. DynaRetarget instead samples rollouts to incorporate dynamics and handle contact implicitly, however, it's not able to accommodate arbitrary constraints.

DSMS combines whole-body dynamic feasibility with a contact-implicit formulation, enabling contact-rich motions such as rolling and crawling without extensive tuning. It also supports arbitrary equality and inequality constraints. Our ablations further show that better-modeled trajectories yield better downstream tracking controllers, underscoring the importance of this approach.

VI. Conclusion

We presented a contact-implicit multi-shooting framework for transforming kinematically feasible references into dynamically feasible whole-body humanoid motions. By using a differentiable simulator as the transition model, the framework accounts for contact, friction, impacts, self-collision, and actuation limits without requiring a prescribed contact sequence or manual constraint setup. The resulting reference trajectories improve downstream RL convergence

and tracking performance while enabling contact-rich and dynamic behaviors such as crawling, jumping, backflip, and side-rolling.

Future work will investigate alternative differentiable simulators and sampling-based optimization methods, automatic discovery of contact-rich behaviors, and extension to additional robot morphologies.

REFERENCES

[1] Z. Gu, J. Li, W. Shen, W. Yu, Z. Xie, S. McCrory, X. Cheng, A. Shamsah, R. Griffin, C. K. Liu, *et al.*, "Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning," *IEEE/ASME Transactions on Mechatronics*, vol. 31, no. 2, pp. 2300–2330, 2026.

[2] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu, "Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion," *arXiv:2508.08241*, 2025.

[3] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM TOG*, vol. 37, no. 4, pp. 1–14, 2018.

[4] A. Allshire, H. Choi, J. Zhang, D. McAllister, A. Zhang, C. M. Kim, T. Darrell, P. Abbeel, J. Malik, and A. Kanazawa, "Visual imitation enables contextual humanoid control," in *Robot Learning*.

[5] L. Yang, X. Huang, Z. Wu, A. Kanazawa, P. Abbeel, C. Sferrazza, C. K. Liu, R. Duan, and G. Shi, "Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction," *arXiv:2509.26633*, 2025.

[6] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, *et al.*, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," *arXiv:2502.01143*, 2025.

[7] V. Voleti, B. Oreshkin, F. Bocquelet, F. Harvey, L.-S. Ménard, and C. Pal, "Smpl-ik: Learned morphology-aware inverse kinematics for ai driven artistic workflows," in *SIGGRAPH Asia 2022 Technical Communications*, pp. 1–7, 2022.

[8] K. Ayusawa and E. Yoshida, "Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1343–1357, 2017.

[9] S. A. Esteban, V. Kurtz, A. B. Ghansah, and A. D. Ames, "Reduced-order model guided contact-implicit model predictive control for humanoid locomotion," in *2025 IEEE ICRA*, pp. 14735–14741, IEEE, 2025.

[10] Y. Ze, S. Zhao, W. Wang, A. Kanazawa, R. Duan, P. Abbeel, G. Shi, J. Wu, and C. K. Liu, "Twist2: Scalable, portable, and holistic humanoid data collection system," *arXiv:2511.02832*, 2025.

[11] H. Wang, Q. Liao, B. Zhang, K. Ren, K. Sreenath, and X. Xiong, "Spark: Skeleton-parameter aligned retargeting on humanoid robots with kinodynamic trajectory optimization," *arXiv:2603.11480*, 2026.

[12] J. P. Araujo, Y. Ze, P. Xu, J. Wu, and C. K. Liu, "Retargeting matters: General motion retargeting for humanoid motion tracking," *arXiv:2510.02252*, 2025.

[13] Q. Zhao, K. Yang, X. Wang, S. Zhao, Y. Lu, X. Zhang, Q. Shen, X.-X. Long, and X. Cao, "Make tracking easy: Neural motion retargeting for humanoid whole-body control," *arXiv:2603.22201*, 2026.

[14] F. Liu, Z. Gu, Y. Cai, Z. Zhou, H. Jung, J. Jang, S. Zhao, S. Ha, Y. Chen, D. Xu, *et al.*, "Opt2skill: Imitating dynamically-feasible whole-body trajectories for versatile humanoid loco-manipulation," *IEEE Robotics and Automation Letters*, 2025.

[15] W. Zhao, Y. Zhao, J. Pajarinen, and M. Muehlebach, "Bi-level motion imitation for humanoid robots," in *Conference on Robot Learning*, pp. 963–981, PMLR, 2025.

[16] Y. Li, P. Zhi, Y. Wang, T. Liu, S. Yan, W. Liu, X. Wang, B. Jia, and S. Huang, "Omnitrack: General motion tracking via physics-consistent reference," *arXiv:2602.23832*, 2026.

[17] V. Dhedin, I. Taouil, S. Omar, D. Yu, K. Tao, A. Dai, and M. Khadiv, "Dynaretarget: Dynamically-feasible retargeting using sampling-based trajectory optimization," *arXiv:2602.06827*, 2026.

[18] C. Roux, L. De Matteïs, A. Jordana, V. Guillet, N. Mansard, O. Stasse, and P. Souères, "Direct dynamic retargeting for humanoid imitation learning from videos," *arXiv:2605.23762*, 2026.

[19] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *2016 IEEE ICRA*, pp. 1447–1454, 2016.

[20] Z. Olkin, W. D. Compton, R. M. Bena, and A. D. Ames, "Chasing autonomy: Dynamic retargeting and control guided rl for performant and controllable humanoid running," *arXiv:2603.25902*, 2026.

[21] 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.

[22] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *Robotics Research: The 18th International Symposium ISRR*, pp. 985–1000, Springer, 2019.

[23] S. Le Cleac'h, T. A. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, "Fast contact-implicit model predictive control," *Robotics*, vol. 40, pp. 1617–1629, 2024.

[24] V. Kurtz, A. Castro, A. Ö. Önol, and H. Lin, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," *The International Journal of Robotics Research*, vol. 45, no. 1, pp. 23–40, 2026.

[25] G. Kim, D. Kang, J.-H. Kim, and H.-W. Park, "Contact-implicit differential dynamic programming for model predictive control with relaxed complementarity constraints," in *2022 IEEE/RSJ IROS*, pp. 11978–11985, IEEE, 2022.

[26] J.-P. Sleiman, J. Carius, R. Grandia, M. Wermelinger, and M. Hutter, "Contact-implicit trajectory optimization for dynamic object manipulation," in *2019 IEEE/RSJ IROS*, pp. 6814–6821, IEEE, 2019.

[27] V. Kurtz and H. Lin, "Contact-implicit trajectory optimization with hydroelastic contact and ilqr," in *2022 IEEE/RSJ IROS*, pp. 8829–8834, IEEE, 2022.

[28] J. Z. Zhang, T. A. Howell, Z. Yi, C. Pan, G. Shi, G. Qu, T. Erez, Y. Tassa, and Z. Manchester, "Whole-body model-predictive control of legged robots with mujoco, 2025," *URL https://arxiv.org/abs/2503.04613*, 2025.

[29] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv:2212.00541*, 2022.

[30] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[31] T. A. Howell, S. Le Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable simulator for robotics," *arXiv:2203.00806*, vol. 9, no. 2, p. 4, 2022.

[32] S. Le Cleac'h, M. Schwager, Z. Manchester, V. Sindhwani, P. Florence, and S. Singh, "Single-level differentiable contact simulation," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4012–4019, 2023.

[33] Q. Le Lidec, L. Montaut, Y. de Mont-Marin, F. Schramm, and J. Carpentier, "Highly-efficient differentiable simulation for robotics,"

[34] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, no. 3, pp. 231–247, 1997.

[35] A. Castro, X. Han, and J. Masterjohn, "Irrotational contact fields," *IEEE Transactions on Robotics*, 2025.

[36] J. Carpentier, L. Montaut, and Q. L. Lidec, "From compliant to rigid contact simulation: a unified and efficient approach," *arXiv:2405.17020*, 2024.

[37] E. Todorov, "Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco," in *2014 IEEE ICRA*, pp. 6054–6061, IEEE, 2014.

[38] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2023.

[39] 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.

[40] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel, "mjlab: A lightweight framework for gpu-accelerated robot learning," 2026.

[41] BONES Studio, "BONES-SEED." [Online]. Available: https://huggingface.co/datasets/bones-studio/seed, 2025.

[42] "mjlab tracking settings." [Online]. Available: https://github.com/mujocolab/mjlab/blob/v1.5.3/src/mjlab/tasks/tracking/tracking_env_cfg.py, 2026.

[43] X. Zhang, S. Haener, V. Madabushi, and M. Tucker, "Kinodynamic motion retargeting for humanoid locomotion via multi-contact whole-body trajectory optimization," *arXiv:2603.09956*, 2026.