SPARK: Skeleton-Parameter Aligned Retargeting on Humanoid Robots with Kinodynamic Trajectory Optimization

Anonymous Authors

***Abstract*— Human motion provides rich priors for training general-purpose humanoid control policies, but raw demonstrations are often incompatible with a robot's kinematics and dynamics, limiting their direct use. We present a two-stage pipeline for generating natural and dynamically feasible motion references from task-space human data. First, we convert human motion into a unified robot description format (URDF)-based skeleton representation and calibrate it to the target humanoid's dimensions. By aligning the underlying skeleton structure rather than heuristically modifying task-space targets, this step significantly reduces inverse kinematics error and tuning effort. Second, we refine the retargeted trajectories through progressive kinodynamic trajectory optimization (TO), solved in three stages: kinematic TO, inverse dynamics, and full kinodynamic TO, each warm-started from the previous solution. The final result yields dynamically consistent state trajectories and joint torque profiles, providing high-quality references for learning-based controllers. Together, skeleton calibration and kinodynamic TO enable the generation of natural, physically consistent motion references across diverse humanoid platforms.**

I. Introduction

Humanoid robots promise to operate in human-centered environments, performing tasks that require rich whole-body coordination, dexterous contact interaction, and agile locomotion. Human motion datasets [1] provide a rich source of coordinated behaviors that can serve as motion priors for training humanoid control policies. Recent retargeting methods [2], [3] have shown that human demonstrations can be converted into natural-looking humanoid motions by mapping human keyframes to robot inverse kinematics (IK) targets, which can then be robustly tracked by controllers trained with reinforcement learning (RL) [4], [5], [6]. However, most existing retargeting tools model human-robot morphology mismatch using root-link-centered task-space scalings and per-keyframe local offsets [3] that are inconsistent with the underlying skeletal structure, often requiring substantial IK objective tuning. Moreover, most retargeting pipelines are purely kinematic [2], [3], [7], so the resulting references can be dynamically infeasible. Using such references for RL forces the policy to both track the motion and compensate for physical inconsistencies, increasing the training difficulty.

To address these challenges, we propose a two-stage pipeline that improves both kinematic and dynamic fidelity. First, instead of root-to-keyframe scaling and keyframe local offsets, we generate a human skeletal URDF from task-space motion and calibrate it to the target robot dimensions. Unlike prior human skeleton calibration strategies [2] which are often limited to SMPL [8], our URDF calibration applies to any human motion format with an implicit skeleton, improving IK quality while reducing tuning efforts.

Fig. 1: Experimental results of highly dynamic side flip tracking on Unitree G1.

Second, we formulate a kinodynamic TO problem to recover dynamically feasible motion references from the calibrated kinematic sequence. Rather than solving this highly nonlinear and nonconvex kinodynamic TO problem [9] in one shot, we tackle it progressively in three stages: kinematic TO, inverse dynamics, and full kinodynamic TO. Each stage uses the optimal solution of the previous stage as an initial guess, gradually introducing kinematic and dynamic consistency. The resulting trajectories provide not only dynamically valid state trajectories but also corresponding torque profiles, which serve as strong supervision for downstream RL-based tracking policies. This is especially beneficial when training highly dynamic motions like the side flip in Fig. 1.

Our approach bridges human demonstration, model-based TO, and learning-based control. By disentangling kinematic alignment, dynamic feasibility recovery, and robust control policy learning, we improve both motion quality and training efficiency. In summary, our main contributions are as follows.

- A general algorithm that generates and calibrates a human URDF from task-space motion, enabling physically interpretable human-to-robot geometric alignment and improved IK retargeting quality.
- A progressive kinodynamic TO framework that gradually refines retargeted motions into kinematically and dynamically feasible humanoid trajectories.
- An integrated pipeline that produces high-quality motion and torque references for RL training, bridging kinematic retargeting, dynamics enforcement through TO, and motion tracking controller training through RL.

II. Related Work

A. Human Motion Retargeting

Retargeting human motions to humanoid robots is essential for leveraging human motion datasets in robot learning and control. A central challenge arises from the morphological differences between humans and robots, including discrepancies in link lengths and kinematic structures.

One line of work addresses this issue through simplified task space modifications that are incompatible with the underlying skeleton structure. GMR [3], for example, applies root-to-keyframe scalings and per-keyframe local offsets to human motion in task space before using it as IK targets. While effective within a limited range of configurations, this approach is inherently local. For example, the dimension mismatch caused by a taller and wider shoulder and a shorter arm should be corrected by making the spine and shoulder links longer and the arm links shorter. If following GMR's method and correcting it by root-to-arm scaling, even on the same robot, different joint configurations will result in a different optimal scaling factor, making it not calibratable. GMR solved this issue by adding keyframe orientation tracking when the position references are off, which increases the need to calibrate nontrivial keyframe orientation offsets from human to robot, and adds IK tuning efforts.

An alternative strategy is to calibrate the underlying human skeletal model to better match the robot dimensions. PHC [2] optimizes the shape parameters of the human model SMPL to align it with the dimensions of the robot, thus producing task-space references that are more consistent with the target robot. However, the SMPL [8] shape parameters are derived from principal component analysis (PCA) on the shapes of human bodies. Although they are well-suited for representing realistic human body variations, calibrating them to robot dimensions, which are often outside the human distribution, can introduce undesirable artifacts, such as asymmetric limb proportions, spine distortions, or inconsistent foot geometry.

In contrast, our method directly calibrates the human URDF generated from task space human motion, which are not only more physically interpretable and accurate but also more generalizable to diverse human motion formats with an implicit skeleton structure other than SMPL.

B. Model-Based Trajectory Optimization for Humanoids

Model-based trajectory optimization (TO) serves as a cornerstone for generating dynamically feasible motions in high-dimensional humanoid systems. These methods formulate motion generation as a constrained optimization problem, enforcing physically grounded dynamics constraints ranging from simplified centroidal dynamics [10] to full order Lagrangain dynamics [9]. With careful engineering, such optimization problems can be used to control humanoid robots in real time using solving methods such as Differential Dynamic Programming [11], [12] and Sequential Quadratic Programming [9], [13].

Despite their success, classical model-based TO pipelines typically rely on carefully hand-crafted reference trajectories. These references include but are not limited to predefined base link pose and velocity trajectories [9] and foot swing trajectories and contact schedules [12]. Designing such references is non-trivial in that it requires substantial domain expertise, manual tuning for each robot morphology, and often task-specific adjustments. Moreover, because these references are hard coded rather than optimized, the resulting motions may appear overly regularized or conservative, lacking the variability, expressiveness, and subtle coordination patterns observed in natural human movements. As a result, even when dynamically feasible, the generated behaviors can be perceptually unnatural.

In contrast, our work starts from natural human motion references and refines them through model-based kinodynamic TO. We treat human demonstrations as rich priors and use TO to enforce robot-specific dynamics, contact feasibility, and actuation limits. This shifts the role of trajectory optimization from motion design to motion refinement, enabling dynamically consistent humanoid behaviors that retain the structure and naturalness of human movement while satisfying the physical constraints.

C. Reinforcement Learning for Robust Motion Tracking

Reinforcement learning (RL) has become a dominant paradigm for humanoid control due to its ability to learn feedback policies that are robust to modeling errors, disturbances, and contact uncertainties. However, RL-based control faces two persistent challenges: policies trained purely from task rewards may converge to unnatural motions, and training high-dimensional humanoid policies typically requires large amounts of data and computation [4].

To improve training efficiency and motion quality, prior works commonly adopt one of the following two strategies. A first approach uses model-based TO to synthesize dynamically feasible reference trajectories, and subsequently trains RL policies to track them [14], [15]. Although this reduces exploration difficulty and accelerates convergence, the reference motions remain shaped by handcrafted objectives, contact schedules, and heuristic design choices, inheriting the same naturalness limitations as pure TO methods. A second approach directly feeds kinematically retargeted human motions into RL as tracking references [16], [17], [18]. This significantly enhances motion naturalness by leveraging human demonstrations, but because kinematic retargeting does not enforce dynamics constraints, the RL policy must implicitly correct dynamics infeasibility. Consequently, motion correction and policy learning become entangled, increasing the difficulty of training.

In contrast, our approach bridges these paradigms. We first retarget natural human motion to the robot kinematics and then refine it through kinodynamic TO to enforce dynamics consistency. The optimized motions are subsequently used as references for RL training. This preserves the structural naturalness of human demonstrations while ensuring physical plausibility prior to policy learning. Moreover, our kinodynamic TO provides joint torque references that are unavailable in purely kinematic retargeting pipelines, offering richer

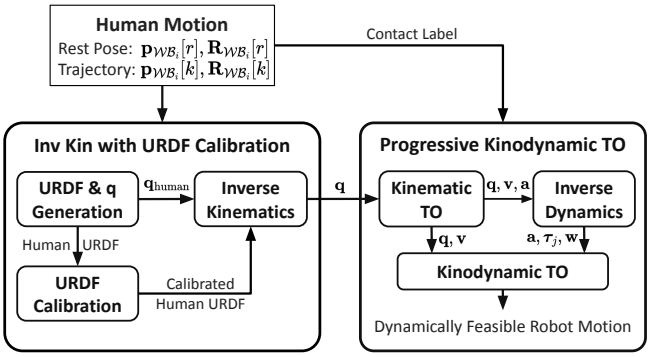

Fig. 2: Retargeting framework.

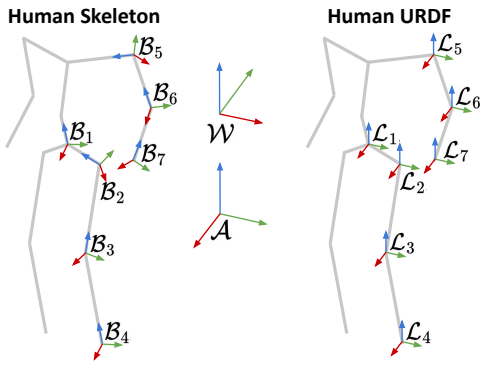

Fig. 3: Coordinate frame definitions of human skeleton and the corresponding human URDF in rest pose. While the human bone frames $\mathcal{B}_i$ are often aligned to the specific bones, we align the URDF link frames $\mathcal{L}_i$ to a user-defined aligned frame $\mathcal{A}$ with the same orientation as the robot URDF root link to facilitate URDF calibration.

supervision and further reducing learning complexity.

III. Inverse Kinematics with URDF Calibration

Fig. 2 provides an overview of our Skeleton-Parameter Aligned Retargeting framework with Kinodynamic trajectory optimization (SPARK), which consists of an upstream inverse-kinematics (IK) stage and a downstream trajectory-optimization (TO) stage. In the IK stage, we convert task space human motion into a human URDF and a human generalized coordinate trajectory (Sec. III-A). We then calibrate the human URDF to the target robot dimensions (Sec. III-B), and replay the generalized coordinates on the calibrated URDF to generate task space references for IK (Sec. III-C), producing a robot motion reference in robot generalized coordinates. In the TO stage, we refine this trajectory by removing kinematic artifacts via kinematic TO (KTO; Sec. IV-B) and dynamic artifacts via inverse dynamics (ID; Sec. IV-C) and kinodynamic TO (KDTO; Sec. IV-D).

A. Human URDF and Generalized Coordinates Generation

Human URDF Generation: Human skeletons and humanoid robots typically differ in both the number of links and their dimensions. To enable reliable retargeting, we first convert a human skeleton motion sequence into a human URDF and a generalized-coordinate trajectory, and then calibrate the human URDF to match the target robot dimensions. Replaying the generalized coordinates on the calibrated URDF yields high-quality task-space references (Sec. III-C) that are consistent with the robot morphology.

The conversion workflow is illustrated in Fig. 3. A human skeleton motion sequence provides the pose of each bone frame $\mathcal{B}_i$ ($i \in \{1, \dots, N_{\text{frame}}\}$) with respect to the world frame $\mathcal{W}$. In the URDF, we create one link $\mathcal{L}_i$ per bone $\mathcal{B}_i$ and construct the model from a rest pose specified by the motion file. The rest pose is typically a standing configuration with vertical legs and an upright torso; arm configurations may vary but should remain left–right symmetric. If a suitable rest pose is missing, we define one prior to URDF construction.

The coordinate-frame-related notations in this paper are as follows: For any three frames $\mathcal{A}, \mathcal{B}, \mathcal{C}$, ${}^{\mathcal{C}}\mathbf{p}_{\mathcal{AB}}$ denotes the position vector from the origin of frame $\mathcal{A}$ to the origin of frame $\mathcal{B}$, expressed in frame $\mathcal{C}$. $\mathbf{R}_{\mathcal{AB}}$ denotes the rotation matrix that takes in a vector $\mathbf{v}$ expressed in $\mathcal{B}$ and returns the same vector expressed in $\mathcal{A}$, i.e. ${}^{\mathcal{A}}\mathbf{v} = \mathbf{R}_{\mathcal{AB}}{}^{\mathcal{B}}\mathbf{v}$. It is also the relative rotation of frame $\mathcal{B}$ with respect to frame $\mathcal{A}$. $\boldsymbol{\xi}_{\mathcal{AB}}$ denotes the quaternion corresponding to $\mathbf{R}_{\mathcal{AB}}$.

In the rest pose r, we set each link origin to match the corresponding bone origin, i.e., $\mathbf{p}_{\mathcal{WL}_i}[r] = \mathbf{p}_{\mathcal{WB}_i}[r]$. We then rotate each link frame to coincide with the orientation of a user-defined aligned frame $\mathcal{A}$, which has the same orientation as the robot URDF root link to facilitate the URDF calibration in Sec. III-B. This alignment simplifies both URDF generation and calibration: the local rest-pose relative translations ${}^{\mathcal{L}_i}\mathbf{p}_{\mathcal{L}_i\mathcal{L}_j}[r]$ become identical to the global relative translations expressed in $\mathcal{A}$, i.e., ${}^{\mathcal{L}_i}\mathbf{p}_{\mathcal{L}_i\mathcal{L}_j}[r] = {}^{\mathcal{A}}\mathbf{p}_{\mathcal{L}_i\mathcal{L}_j}[r]$. Consequently, constructing the URDF reduces to computing these relative translations from the world-frame rest-pose bone positions ${}^{\mathcal{W}}\mathbf{p}_{\mathcal{WB}_i}[r]$ using

$$\begin{aligned} {}^{\mathcal{L}_i}\mathbf{p}_{\mathcal{L}_i\mathcal{L}_j}[r] &= {}^{\mathcal{A}}\mathbf{p}_{\mathcal{L}_i\mathcal{L}_j}[r] \\ &= \mathbf{R}_{\mathcal{WA}}^T \left({}^{\mathcal{W}}\mathbf{p}_{\mathcal{WL}_j}[r] - {}^{\mathcal{W}}\mathbf{p}_{\mathcal{WL}_i}[r]\right) \\ &= \mathbf{R}_{\mathcal{WA}}^T \left({}^{\mathcal{W}}\mathbf{p}_{\mathcal{WB}_j}[r] - {}^{\mathcal{W}}\mathbf{p}_{\mathcal{WB}_i}[r]\right). \end{aligned} \tag{1}$$

Human Generalized Coordinate Generation: We define the generalized coordinates of the human URDF at timestep k as $\mathbf{q}_{\text{human}}[k] = \left[{}^{\mathcal{W}}\mathbf{p}_{\mathcal{WL}_1}[k]; \boldsymbol{\xi}_{\mathcal{WL}_1}[k]; \phi_1[k]; \dots; \phi_{N_j}[k]\right]$, where ${}^{\mathcal{W}}\mathbf{p}_{\mathcal{WL}_1}$ is the root-link position, $\boldsymbol{\xi}_{\mathcal{WL}_1}$ is the root-link quaternion, and ϕ_n ($n \in \{1, \dots, N_j\}$) are the Euler angles of the n-th spherical joint. These quantities are computed as

$${}^{\mathcal{W}}\mathbf{p}_{\mathcal{WL}_1}[k] = s_{\text{root}}{}^{\mathcal{W}}\mathbf{p}_{\mathcal{WB}_1}[k], \tag{2a}$$

$$\begin{aligned} \boldsymbol{\xi}_{\mathcal{WL}_1}[k] &= \text{quat}\left(\mathbf{R}_{\mathcal{WL}_1}[k]\right) \\ &= \text{quat}\left(\mathbf{R}_{\mathcal{WB}_1}[k]\mathbf{R}_{\mathcal{B}_1\mathcal{L}_1}\right), \end{aligned} \tag{2b}$$

$$\begin{aligned} \phi_n[k] &= \text{euler}\left(\mathbf{R}_{\mathcal{L}_{i_n}\mathcal{L}_{j_n}}[k]\right) \\ &= \text{euler}\left(\mathbf{R}_{\mathcal{B}_{i_n}\mathcal{L}_{i_n}}^T \mathbf{R}_{\mathcal{B}_{i_n}\mathcal{B}_{j_n}}[k]\mathbf{R}_{\mathcal{B}_{j_n}\mathcal{L}_{j_n}}\right), \end{aligned} \tag{2c}$$

where s_{root} is the root position scaling factor to be calibrated

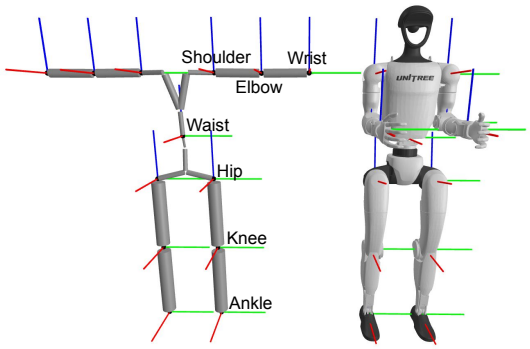

Fig. 4: Rest pose key frame calibration of human URDF to humanoid robot dimensions. We match the link lengths for arms regardless of their absolute positions, and the absolute positions for the remaining key frames. We also calibrate the orientation offsets for end effectors if the robot has enough degree of freedom to track end effector orientations.

in Sec. III-B, quat$(\cdot)$ converts rotation matrix to quaternion, and euler$(\cdot)$ converts rotation matrix to euler angles. $\mathbf{R}_{\mathcal{B}_i\mathcal{L}_i}$ is the relative rotation of the i-th link frame with respect to the corresponding bone frame, which can be computed under rest pose as

$$\mathbf{R}_{\mathcal{B}_i\mathcal{L}_i} = \mathbf{R}_{\mathcal{B}_i\mathcal{A}}[r] = \mathbf{R}^T_{\mathcal{W}\mathcal{B}_i}[r]\mathbf{R}_{\mathcal{W}\mathcal{A}}. \tag{3}$$

B. Human URDF Calibration

We calibrate the human URDF to the target humanoid dimensions by matching keyframes in the respective rest poses. Fig. 4 shows an example of the rest pose configurations and key frame definitions when calibrating human URDF to Unitree G1. The robot rest pose (typically the neutral configuration with zero joint angles) should be a standing pose compatible with the human-URDF rest pose convention. The arm configuration may differ, since we only calibrate arm lengths. We decompose calibration into five groups:

End Effector Calibration: We calibrate the local position and orientation offsets of end effectors such as wrists and ankles from the human URDF to the robot URDF so that their absolute poses can be tracked. If the robot wrist has fewer than three rotational degrees of freedom, its orientation cannot be tracked exactly; in this case, we omit the wrist-orientation calibration. In our IK formulation, end effectors are the only frames with orientation targets, since orientation offsets for intermediate links are generally ambiguous.

Arm Calibration: We select three keyframes on each arm (shoulder, elbow, and wrist) and calibrate the shoulder–elbow and elbow–wrist link lengths. Because human and robot rest poses can differ substantially in arm configuration, we do not calibrate the full 3D relative translations. Instead, we estimate a scalar length scaling factor s_{length} for each segment through the following

$$s_{\text{length}} = \|\mathbf{p}_{\text{robot}}\|_2 \,/\, \|\mathbf{p}_{\text{human}}\|_2\,, \tag{4}$$

where $\|\mathbf{p}_{\text{robot}}\|_2$ is the arm length of the robot and $\|\mathbf{p}_{\text{human}}\|_2$ is the arm length of the human.

Leg Calibration: We choose three keyframes on each leg (hip, knee, and ankle) and calibrate the absolute hip–knee and knee–ankle translations in the human URDF to match the robot URDF.

Main Body Calibration: We select five keyframes on the main body (one waist frame, two shoulder frames, and two hip frames). Because the torso contains additional intermediate links, we correct their dimensions using an affine transformation. In the Unitree G1 example shown in Fig. 4, where the robot URDF root frame and all human URDF frames have their x axes pointing forward and z axes pointing upward in the rest pose, this affine transformation includes an xz shear and independent scalings along y and z. The orientation synchronization was done by the aligned frame $\mathcal{A}$ introduced in Sec. III-A. To calibrate the rest pose transformations of all links between $\mathcal{L}_i$ and $\mathcal{L}_j$ in the human URDF, we find their corresponding links $\mathcal{R}_i, \mathcal{R}_j$ in the robot URDF, and calibrate the affine transformation $\mathbf{A}$ using

$$\mathbf{p}_{\mathcal{R}_i\mathcal{R}_j} = \mathbf{A}\mathbf{p}_{\mathcal{L}_i\mathcal{L}_j} = \begin{bmatrix} 1 & 0 & k_x \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \mathbf{p}_{\mathcal{L}_i\mathcal{L}_j}, \tag{5}$$

where k_x is the xz shear factor and s_y, s_z are the scaling factors along the y and z directions. We then apply the calibrated transform to any intermediate link relative translation $\mathbf{p}_{\mathcal{L}_m\mathcal{L}_n}$, where $\mathcal{L}_m$ and $\mathcal{L}_n$ are links between $\mathcal{L}_i$ and $\mathcal{L}_j$, using

$$\mathbf{p}_{\mathcal{L}_m\mathcal{L}_n,\text{corrected}} = \mathbf{A}\mathbf{p}_{\mathcal{L}_m\mathcal{L}_n}. \tag{6}$$

Setting $\mathcal{L}_i$ to the waist frame and $\mathcal{L}_j$ to a shoulder frame calibrates all links between waist and shoulders. Likewise, setting $\mathcal{L}_i$ to the waist frame and $\mathcal{L}_j$ to a hip frame calibrates links between waist and hips.

Root Position Scale Calibration: After calibrating leg and main body dimensions, we need to calibrate the root position scaling factor s_{root} in (2a) so that stance legs are still sticking on ground after leg dimensions have changed. The calibration can be done using

$$s_{\text{root}} = \left\|\mathbf{p}^{\text{corrected}}_{\mathcal{L}_{\text{root}}\mathcal{L}_{\text{ankle}}}\right\|_2 / \left\|\mathbf{p}_{\mathcal{L}_{\text{root}}\mathcal{L}_{\text{ankle}}}\right\|_2, \tag{7}$$

where $\mathbf{p}^{\text{corrected}}_{\mathcal{L}_{\text{root}}\mathcal{L}_{\text{ankle}}}$ and $\mathbf{p}_{\mathcal{L}_{\text{root}}\mathcal{L}_{\text{ankle}}}$ are the corrected and original root-to-ankle vector respectively.

C. Inverse Kinematics (IK)

After URDF calibration (Sec. III-B), we replay the human generalized-coordinate trajectory in (2) on the calibrated human URDF to obtain task-space references for all keyframes. Because calibration aligns the reference geometry with the target robot morphology, these task-space references can be tracked reliably. We solve IK for the generalized coordiantes $\mathbf{q}$ of the humanoid robot at each timestep via

$$\begin{aligned} \min_{\mathbf{q}} \; & \sum_{i=1}^{N_{\text{EE}}} \|\delta\mathbf{R}(\mathbf{q})\|^2_{\mathbf{W}_\mathbf{R}} + \sum_{i=1}^{N_{\text{frame}}} \|\delta\mathbf{p}(\mathbf{q})\|^2_{\mathbf{W}_\mathbf{p}} \\ & + \|\mathbf{q}_j - \mathbf{q}_{j,\text{prev}}\|^2_{\mathbf{W}_{\mathbf{q}_j}} \\ \text{s.t. } & \mathbf{q}_{j,\min} \le \mathbf{q}_j \le \mathbf{q}_{j,\max} \\ & \mathbf{v}_{j,\min}\Delta t \le \mathbf{q}_j - \mathbf{q}_{j,\text{prev}} \le \mathbf{v}_{j,\max}\Delta t, \end{aligned} \tag{8}$$

where $\|\mathbf{e}\|_{\mathbf{W}}^2 := \mathbf{e}^T\mathbf{W}\mathbf{e}$ denotes a $\mathbf{W}$-weighted squared ℓ_2 norm. The objective minimizes position errors $\delta\mathbf{p}$ over all N_{frame} keyframes and orientation errors $\delta\mathbf{R}$ only over the N_{EE} end effectors. We omit orientation targets for intermediate links, since their orientations are determined by the connected keyframe positions. This avoids calibrating ambiguous intermediate-link orientation offsets. If a robot wrist has fewer than three rotational degrees of freedom, we also drop wrist-orientation tracking. Finally, we enforce joint position and velocity limits and regularize towards the previous solution $\mathbf{q}_{\text{prev}}$ to reduce jitter near singularities.

IV. Progressive Kinodynamic TO

A. General TO Formulation

Given the IK retargeting result $\mathbf{q}_{\text{IK}}^*$ and the contact labels from the human motion, we further apply kinodynamic trajectory optimization (KDTO) to obtain humanoid motions that are feasible both kinematically and dynamically. Directly solving KDTO is difficult due to the problem's high dimensionality and nonlinearity, so we adopt a progressive pipeline with three stages: (i) kinematic TO (KTO; Sec. IV-B) to correct contact kinematics and self-collisions, (ii) per-timestep inverse dynamics (ID; Sec. IV-C) to produce a good initialization for second-order dynamic quantities, and (iii) a final KDTO stage (Sec. IV-D) that jointly optimizes kinematics and dynamics. Both KTO and KDTO can be written in the following general form:

$$\begin{aligned}
\min_{\mathbf{x}[0:N],\mathbf{u}[0:N-1]} \quad & l_{\text{track}}(\mathbf{x}[0:N]) + l_{\text{reg}}(\mathbf{x}[0:N],\mathbf{u}[0:N-1]) \\
\text{s.t. } & \mathbf{x}[k+1] = \mathbf{f}\left(\mathbf{x}[k],\mathbf{u}[k]\right) && \text{(9a)} \\
& \mathbf{g}(\mathbf{x}[k]) \le \mathbf{0} \\
& \mathbf{h}\left(\mathbf{x}[k],\mathbf{u}[k]\right) \le \mathbf{0} && \text{(9b)} \\
& \mathbf{x}_{\min} \le \mathbf{x}[k] \le \mathbf{x}_{\max} \\
& \mathbf{u}_{\min} \le \mathbf{u}[k] \le \mathbf{u}_{\max}. && \text{(9c)}
\end{aligned}$$

The exact definitions of the decision variables, costs and constraints differ between KTO (Sec. IV-B) and KDTO (Sec. IV-D), and are defined in the corresponding subsections. In general, the decision variables are the system states $\mathbf{x}$ and inputs $\mathbf{u}$. The objective consists of a motion-tracking term $l_{\text{track}}(\mathbf{x}[0:N])$ and a smoothing regularizer $l_{\text{reg}}(\mathbf{x}[0:N],\mathbf{u}[0:N-1])$. The constraints include the system dynamics (9a), general inequality constraints (9b), and box constraints (9c) on states and inputs. Constraints involving time index $k+1$ apply for all $k \in \{0,\dots,N-1\}$; all others apply for $k \in \{0,\dots,N\}$.

B. Kinematic TO (KTO)

Decision Variables: For KTO, the system state is $\mathbf{x} = [\mathbf{q};\mathbf{v}]$, comprising the robot generalized coordinates $\mathbf{q}$ and generalized velocities $\mathbf{v}$, and the system input is the generalized acceleration $\mathbf{u} = \mathbf{a}$. We initialize KTO using the IK solution: $\mathbf{q}_{\text{guess}} = \mathbf{q}_{\text{IK}}^*$. We then obtain $\mathbf{v}_{\text{guess}}$ and $\mathbf{a}_{\text{guess}}$ via finite differences.

Cost Function: The KTO tracking cost $l_{\text{track}}^{\text{KTO}}$ penalizes task-space pose errors $\delta\mathbf{p}_i, \delta\mathbf{R}_i$ and task-space velocity errors $\delta\mathbf{v}_i, \delta\boldsymbol{\omega}_i$ for all N_{frame} keyframes used in IK, with targets computed from the IK solution $\mathbf{q}_{\text{IK}}^*$ and $\mathbf{v}_{\text{IK}}^*$. The regularization term $l_{\text{reg}}^{\text{KTO}}$ penalizes generalized accelerations $\mathbf{a}$ to smooth the trajectory.

$$\begin{aligned}
l_{\text{track}}^{\text{KTO}} = \sum_{k=0}^{N}\sum_{i=0}^{N_{\text{frame}}} & \|\delta\mathbf{p}_i(\mathbf{q}[k])\|_{\mathbf{W_p}}^2 + \|\delta\mathbf{R}_i(\mathbf{q}[k])\|_{\mathbf{W_R}}^2 + \\
& \|\delta\mathbf{v}_i(\mathbf{q}[k],\mathbf{v}[k])\|_{\mathbf{W_v}}^2 + \|\delta\boldsymbol{\omega}_i(\mathbf{q}[k],\mathbf{v}[k])\|_{\mathbf{W}_\omega}^2 && \text{(10a)} \\
l_{\text{reg}}^{\text{KTO}} = \sum_{k=0}^{N-1} & \|\mathbf{a}[k]\|_{\mathbf{W_a}}^2 && \text{(10b)}
\end{aligned}$$

Double-Integrator Dynamics Constraint: The KTO dynamics use a semi-implicit double integrator that maps generalized accelerations $\mathbf{a}$ to states $(\mathbf{q},\mathbf{v})$:

$$\begin{aligned}
\mathbf{q}[k+1] &= \mathbf{q}[k] \oplus \mathbf{v}[k+1]\,\Delta t \\
\mathbf{v}[k+1] &= \mathbf{v}[k] + \mathbf{a}[k]\,\Delta t,
\end{aligned} \tag{11}$$

where $\oplus$ denotes the additive group operation (to handle configuration manifolds).

Kinematic Contact Constraints: A key role of KTO is to enforce consistent contact kinematics. We impose three sets of constraints:

$$\begin{aligned}
p_{\text{swing},z} &\ge 0, && \text{(12a)} \\
p_{\text{first contact},z} &= 0 \\
R_{\text{first contact},zz} &= 1-\epsilon, && \text{(12b)} \\
\mathbf{v}_{\text{contact}} &= \mathbf{0} \\
\boldsymbol{\omega}_{\text{contact}} &= \mathbf{0}. && \text{(12c)}
\end{aligned}$$

The swing-foot-above-ground constraint (12a) enforces a nonnegative swing-foot height $p_{\text{swing},z}$. The first-contact-on-ground constraint (12b) sets the contact-foot height to zero at the first contact timestep and approximately aligns the foot with the ground by constraining $R_{\text{first contact},zz} = 1-\epsilon$, where ϵ is a small slack to avoid numerical infeasibility (we use $\epsilon = 10^{-4}$). Finally, the holonomic contact constraint (12c) enforces zero linear and angular velocities $\mathbf{v}_{\text{contact}}, \boldsymbol{\omega}_{\text{contact}}$ at the contact foot. Together, these constraints keep the swing foot above the ground and establish holonomic surface contact for the stance foot.

Self Collision Avoidance Constraint: We formulate self-collision avoidance constraints on a point-to-point basis. At any timestep k, for each collision pair $(\mathbf{p}_i,\mathbf{p}_j) \in \mathcal{C}$ (with i,j denoting collision-point indices), we enforce that the distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ exceeds a minimum separation $d_{(i,j),\min}$:

$$\|\mathbf{p}_i(\mathbf{q}[k]) - \mathbf{p}_j(\mathbf{q}[k])\|_2 \ge d_{(i,j),\min}. \tag{13}$$

This formulation extends naturally to sphere-to-sphere constraints by increasing $d_{(i,j),\min}$ by the sum of radii.

Joint Position and Velocity Limits: The KTO box constraints include joint-position limits and joint-velocity limits:

$$\begin{aligned}
\mathbf{q}_{j,\min} \le \mathbf{q}_j[k] \le \mathbf{q}_{j,\max}, \\
\mathbf{v}_{j,\min} \le \mathbf{v}_j[k] \le \mathbf{v}_{j,\max}.
\end{aligned} \tag{14}$$

C. Inverse Dynamics (ID)

At each timestep, we fix the generalized coordinates and velocities to the KTO optimum $(\mathbf{q}^*_{\text{KTO}}, \mathbf{v}^*_{\text{KTO}})$ and solve inverse dynamics (ID) as a single-timestep quadratic program (QP).

Decision Variable: ID optimizes the generalized acceleration $\mathbf{a}$, joint torques $\boldsymbol{\tau}_j$, and contact wrenches $\mathbf{w}$. The KTO-optimal acceleration $\mathbf{a}^*_{\text{KTO}}$ serves as both the tracking reference and the initialization for $\mathbf{a}$.

Cost Function: The ID objective tracks the KTO-optimal acceleration $\mathbf{a}^*_{\text{KTO}}$ while minimizing joint torques $\boldsymbol{\tau}_j$ and contact wrenches $\mathbf{w}$:

$$l^{\text{ID}} = \|\delta\mathbf{a}\|^2_{\mathbf{W_a}} + \|\boldsymbol{\tau}_j\|^2_{\mathbf{W}_{\boldsymbol{\tau}_j}} + \|\mathbf{w}\|^2_{\mathbf{W_w}}. \tag{15}$$

Lagrangian Dynamics Constraint: The ID dynamics constraint uses the full-order Lagrangian rigid body dynamics of a floating-base system:

$$\mathbf{M}(\mathbf{q})\mathbf{a} + \mathbf{H}(\mathbf{q}, \mathbf{v}) = \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\tau}_j\end{bmatrix} + \sum_{i=1}^{N_{\text{foot}}} \mathbf{J}_i^T(\mathbf{q})\mathbf{w}_i. \tag{16}$$

Contact Wrench Constraints: When a foot is in swing phase, its contact wrench must vanish, i.e. $\mathbf{w}_{\text{swing}} = \mathbf{0}$. When a foot is in contact phase, the 6D wrench of a foot $\mathbf{w} = [f_x; f_y; f_z; \tau_x; \tau_y; \tau_z]$ satisfies the Contact Wrench Cone (CWC) constraint [19], which can be written as:

$$\begin{aligned} |f_x| \le \mu f_z,\ |f_y| \le \mu f_z,\ f_z > 0 \\ |\tau_x| \le Y f_z,\ |\tau_y| \le X f_z \\ \tau_{z,\min} \le \tau_z \le \tau_{z,\max}, \end{aligned} \tag{17}$$

$$\begin{aligned} \tau_{z,\min} &= -\mu(X+Y)f_z + |Y f_x - \mu\tau_x| + |X f_y - \mu\tau_y|, \\ \tau_{z,\max} &= +\mu(X+Y)f_z - |Y f_x + \mu\tau_x| - |X f_y + \mu\tau_y|, \end{aligned} \tag{18}$$

where X and Y are the length and width of the foot support-box, and μ is the friction coefficient.

Contact Foot Zero Acceleration Constraint: In (12), the contact-foot velocity is constrained to zero, which implies zero contact-foot acceleration. However, because ID treats generalized acceleration $\mathbf{a}$ as a decision variable and does not explicitly include (12), the resulting solution $\mathbf{a}^*_{\text{ID}}$ may violate this zero-acceleration condition. We therefore enforce it explicitly as:

$$\mathbf{J}_{\text{contact}}(\mathbf{q})\mathbf{a} + \dot{\mathbf{J}}_{\text{contact}}(\mathbf{q}, \mathbf{v})\mathbf{v} = \mathbf{0}. \tag{19}$$

Torque Limits: The ID is subjected to joint torque limits:

$$\boldsymbol{\tau}_{j,\min} \le \boldsymbol{\tau}_j \le \boldsymbol{\tau}_{j,\max}. \tag{20}$$

D. Kinodynamic TO (KDTO)

Decision Variable: The KDTO state is $\mathbf{x} = [\mathbf{q}; \mathbf{v}; \mathbf{a}]$, which includes generalized coordinates, velocities, and accelerations, and the input is $\mathbf{u} = [\boldsymbol{\tau}_j; \mathbf{w}]$, which includes joint torques and contact wrenches. Since KDTO is highly nonlinear and nonconvex, we warm-start it using the KTO and ID solutions. Specifically, we initialize generalized coordinates and velocities with KTO: $\mathbf{q}_{\text{guess}} = \mathbf{q}^*_{\text{KTO}}$ and $\mathbf{v}_{\text{guess}} = \mathbf{v}^*_{\text{KTO}}$, and initialize accelerations, torques, and contact wrenches with ID: $\mathbf{a}_{\text{guess}} = \mathbf{a}^*_{\text{ID}}$, $\boldsymbol{\tau}_{j,\text{guess}} = \boldsymbol{\tau}^*_{j,\text{ID}}$, and $\mathbf{w}_{\text{guess}} = \mathbf{w}^*_{\text{ID}}$.

Cost Function: The KDTO costs share the same tracking term as KTO ($l^{\text{KDTO}}_{\text{track}} = l^{\text{KTO}}_{\text{track}}$), with a regularization term

$$l^{\text{KDTO}}_{\text{reg}} = \sum_{k=0}^{N-1} \|\delta\mathbf{a}[k]\|^2_{\mathbf{W_a}} + \|\delta\boldsymbol{\tau}_j[k]\|^2_{\mathbf{W}_{\boldsymbol{\tau}_j}} + \|\delta\mathbf{w}[k]\|^2_{\mathbf{W_w}}. \tag{21}$$

The regularization term encourages generalized accelerations $\mathbf{a}$, joint torques $\boldsymbol{\tau}_j$, and contact wrenches $\mathbf{w}$ to stay close to the ID solutions $\mathbf{a}^*_{\text{ID}}$, $\boldsymbol{\tau}^*_{j,\text{ID}}$, and $\mathbf{w}^*_{\text{ID}}$.

Constraints: The KDTO constraints combine the KTO constraints with the ID constraints. For dynamics, we enforce both the discrete-time double-integrator constraint (11) and the rigid-body dynamics constraint (16) at each timestep. For contact, we include the kinematic contact constraints (12), the zero-wrench constraint for swing foot, and the CWC constraint (17) for stance foot. Self-collision avoidance follows (13). Finally, we impose joint position and velocity limits (14) and joint-torque limits (20).

V. Results and Analyses

This section evaluates the key components of our pipeline through three sets of experiments. First, we quantify how URDF calibration improves IK retargeting accuracy on multiple humanoid platforms in Sec. V-A. Next, we study motion editing in Sec. V-B, using forward jumping as a case study, and compare KDTO against raw edits and KTO. Finally, we demonstrate the advantage of KDTO for highly dynamic motions such as the side flip shown in Fig. 1 in Sec. V-C, where recovering dynamic feasibility and optionally providing torque references accelerates the RL training process.

A. Effect of URDF Calibration on IK

As discussed in Sec. II-A, prior retargeting methods such as GMR [3] model human–robot morphology differences using task-space root-to-keyframe scalings and per-keyframe local offsets that are not consistent with the underlying skeletal structure. Calibrating these scaling and offset parameters is therefore intertwined with tuning the IK cost, which makes the approach difficult to generalize across large variations in source human body shape and motion. When applied to datasets with substantial diversity in human shapes and motions, such as AMASS [1], it can fail in corner cases such as Fig. 5, where the reference matches along one axis (width) but is incorrect along another (height).

Our calibration of URDF resolves this issue by correcting the dimensions of the actual human skeleton. Once the target robot and the human-motion format are fixed, the same calibration logic can be applied to large datasets containing human skeletons of various dimensions. Because the calibrated task space targets are structurally consistent with the robot, we do not need to impose orientation tracking on intermediate links, and the IK position and orientation tracking weights can be set uniformly to one. This substantially reduces the IK tuning effort. We benchmark the IK mean per-body position error E_{mpbpe} on the AMASS ACCAD

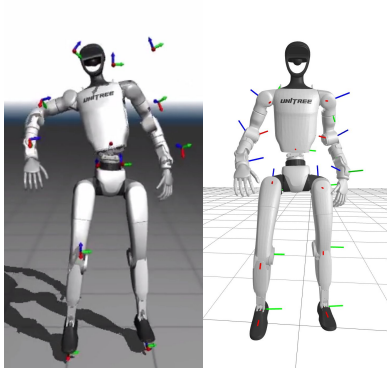

Fig. 5: IK Comparison between GMR (left) and URDF Calibration (right).

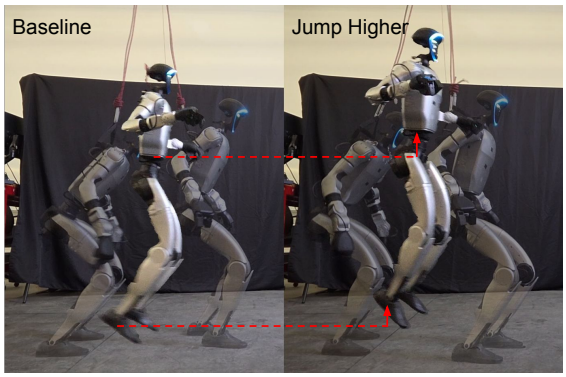

Fig. 6: Unitree G1 performing a forward jump. The left case is the baseline, and the right one is edited to jump higher.

TABLE I: IK E_{mpbpe} [cm] Comparison between GMR and Our URDF Calibration on AMASS ACCAD Dataset over Multiple Robots.

Retarget Method	G1	H1	T1	PM01	Kuavo 4Pro
GMR	9.37	12.53	6.59	9.51	19.50
URDF Calibration	**1.60**	**4.40**	**1.85**	**2.47**	**4.71**

dataset [1] when retargeting to Unitree G1 [20], H1 [21], Booster T1 [22], EngineAI PM01 [23], and Kuavo 4Pro [24]. Table I shows that URDF calibration yields significantly lower E_{mpbpe} than GMR, with improvements of 82.9% on G1, 64.9% on H1, 71.9% on T1, 74.0% on PM01, and 75.8% on Kuavo 4Pro.

B. Motion Editing

Collecting human demonstrations is costly, so editing existing motions is an appealing way to synthesize additional training data and broaden hardware coverage. However, naive edits can introduce kinematic discontinuities and dynamic infeasibility, making the resulting references difficult to track. TO can mitigate these artifacts and improve reference quality. Here we use forward jumping as a case study. We apply two kinds of edits to it: (i) increasing the jump height shown in Fig. 6, (ii) connecting it with an initial and terminal standing phase for safe hardware deployment. Fig. 7 reports RL learning curves of the mean per-body position tracking error E_{mpbpe} when using references from the raw edit, KTO, and KDTO. We train motion-tracking policies following BeyondMimic [18] in IsaacLab [4] on Unitree G1 [20].

Jump Higher: Simply scaling the vertical position to obtain a higher jump is insufficient, because a larger apex height also requires a longer flight time under the same gravity. We therefore scale the foot z-height by $4\times$ and adjust the time discretization Δt based on the resulting change in center-of-mass height. We measured that this height edit increases the maximum center-of-mass height by $1.4\times$, so we scale the time by $\sqrt{1.4}$. Instead of directly scaling Δt, which degrades the accuracy of TO integration, we interpolate the reference to $\sqrt{1.4}$ times the original frequency while keeping Δt unchanged. The right subfigure of Fig. 7 shows that

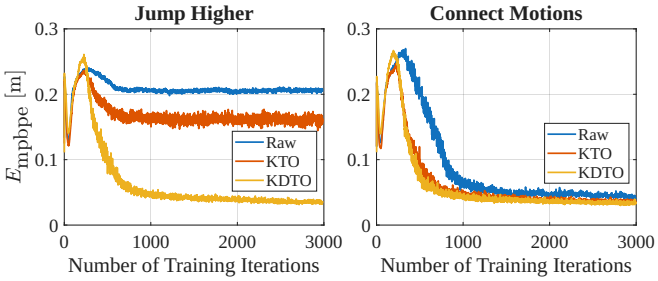

Fig. 7: Comparison of RL E_{mpbpe} learning curves of jumping motion editing when the reference motions are provided by raw edit, KTO, and KDTO.

KDTO achieves a substantially lower final tracking error than KTO and the raw edit. This highlights that when edits affect dynamics-relevant quantities such as gravitational acceleration, recovering a dynamically feasible trajectory with KDTO is important for high-quality tracking.

Connect Motions: The jumping motions cannot be deployed directly on hardware because they start and end with nonzero velocities. A simple workaround is to repeat the first and last frames so that the reference is static at the beginning and end of the execution. This edit introduces velocity discontinuities at the transitions between standing and jumping, which TO can smooth out. We apply this edit to all hardware experiments. The left subfigure of Fig. 7 shows the learning curves for this edit applied to the baseline case where the jump segment itself remains unmodified. KTO improves the convergence speed, and the gap between KTO and KDTO is small. All three variants reach similar final tracking errors, suggesting that this edit introduces only minor dynamic artifacts in our setting.

C. Highly Dynamic Motion Tracking

The advantage of recovering dynamically feasible trajectories with KDTO is most pronounced for highly dynamic motions, such as the side flip motion shown in Fig. 1. Beyond producing a kinematically consistent trajectory, KDTO also yields joint torque references that can accelerate policy learning. In Fig. 8 we compare E_{mpbpe} learning curves for side-flip tracking under three reference variants. The KTO

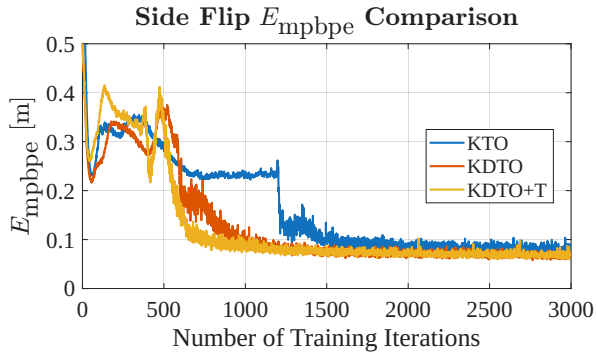

Fig. 8: Comparison of RL E_{mpbpe} learning curves of side flip tracking when the reference motions are provided by KTO, KDTO, KDTO+T.

and KDTO runs use the standard BeyondMimic [18] reward, while KDTO+T adds an exponential joint torque tracking term $r_{\boldsymbol{\tau}_j}$ tracking the optimal KDTO joint torques $\boldsymbol{\tau}^*_{j,\text{KDTO}}$, which is defined as

$$\begin{aligned} r_{\boldsymbol{\tau}_j} &= \exp\left(\tau_{j,\text{MSE}}/\sigma\right) \\ \tau_{j,\text{MSE}} &= \frac{1}{N_j}\left\|\left(\boldsymbol{\tau}_j - \boldsymbol{\tau}^*_{j,\text{KDTO}}\right) ./ \boldsymbol{\tau}_{j,\max}\right\|_2^2 . \end{aligned} \tag{22}$$

This term encourages joint torques $\boldsymbol{\tau}_j$ to track KDTO-optimized torques $\boldsymbol{\tau}_{j,\text{KDTO}}$. Here, $\tau_{j,\text{MSE}}$ is the mean-squared normalized torque error, with $./$ denoting element-wise division. N_j is the number of joints and $\boldsymbol{\tau}_{j,\max}$ is the torque limit. As shown in Fig. 8, the KTO learning curve plateaus for roughly 500 iterations before converging, as training struggles near the most challenging flip timestep when the robot is inverted. In this regime, KTO must simultaneously learn to track the difficult motion while correcting dynamics infeasibility. By contrast, KDTO exhibits a much shorter plateau because the reference motion is already dynamically feasible, and KDTO+T converges even faster thanks to the additional guidance from the torque-tracking reward.

VI. Conclusion

In this paper, we presented a pipeline for transferring task space human motion to humanoid robots with no per-motion tuning. We first introduced a URDF calibration procedure that aligns human skeletal dimensions with the target robot prior to reference generation, producing task-space targets that are kinematically consistent with the robot morphology. This structural alignment substantially reduces the need for manual IK weight adjustment and orientation-offset tuning.

We then investigated the role of dynamics consistency in tracking edited and highly dynamic motions and demonstrated that KDTO recovers dynamically feasible trajectories and provides additional joint torque references, which improve both learning efficiency and tracking performance.

References

[1] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *International Conference on Computer Vision*, Oct. 2019, pp. 5442–5451.

[2] Z. Luo, J. Cao, K. Kitani, W. Xu *et al.*, "Perpetual humanoid control for real-time simulated avatars," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 895–10 904.

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

[4] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Munoz, X. Yao, R. Zurbrügg, N. Rudin *et al.*, "Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning," *arXiv preprint arXiv:2511.04831*, 2025.

[5] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs *et al.*, "Mujoco playground," *arXiv preprint arXiv:2502.08844*, 2025.

[6] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel, "mjlab: A lightweight framework for gpu-accelerated robot learning," *arXiv preprint arXiv:2601.22074*, 2026.

[7] 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 preprint arXiv:2509.26633*, 2025.

[8] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 851–866.

[9] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, "Tailoring solution accuracy for fast whole-body model predictive control of legged robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 074–11 081, 2024.

[10] 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*. IEEE, 2014, pp. 295–302.

[11] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[12] F. Farshidian *et al.*, "OCS2: An open source library for optimal control of switched systems," [Online]. Available: https://github.com/leggedrobotics/ocs2.

[13] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006.

[14] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4975–4982, 2023.

[15] 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.

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

[17] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[18] 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 preprint arXiv:2508.08241*, 2025.

[19] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5107–5112.

[20] Unitree G1: `https://www.unitree.com/g1`.

[21] Unitree H1: `https://www.unitree.com/h1`.

[22] Booster T1: `https://www.booster.tech/booster-t1`.

[23] EngineAI PM01: `https://en.engineai.com.cn/product-pm01.html`.

[24] Kuavo 4Pro: `https://www.lejurobot.cn/zh/application/kuavo-my`.