Adaptive Tracking of a Single-Rigid-Body Character in Various Environments

ANONYMOUS AUTHOR(S)
SUBMISSION ID: PAPERS_523S1

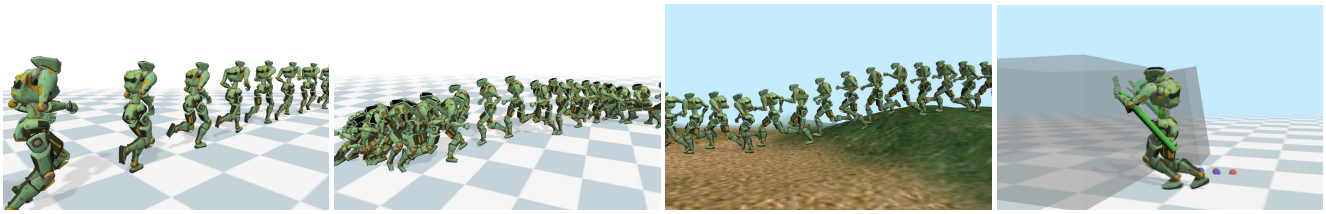


Fig. 1. Our character can perform tasks in various environments using a policy learned from a short reference motion clip on flat ground. The results suggest that our controller can perform a range of activities, such as transitioning, climbing on rough terrain, and pushing objects, without additional training.

Since the introduction of DeepMimic [Peng et al. 2018], subsequent research has focused on expanding the repertoire of simulated motions across various scenarios. In this study, we propose an alternative approach for this goal, a deep reinforcement learning method based on the simulation of a single-rigid-body character. Using the centroidal dynamics model (CDM) to express the full-body character as a single rigid body (SRB) and training a policy to track a reference motion, we can obtain a policy that is capable of adapting to various unobserved environmental changes and controller transitions without requiring any additional learning. Due to the reduced dimension of state and action space, the learning process is sample-efficient. The final full-body motion is kinematically generated in a physically plausible way, based on the state of the simulated SRB character. The SRB simulation is formulated as a quadratic programming (QP) problem, and the policy outputs an action that allows the SRB character to follow the reference motion. We demonstrate that our policy, efficiently trained within 30 minutes on an ultraportable laptop, has the ability to cope with environments that have not been experienced during learning, such as running on uneven terrain or pushing a box, and transitions between learned policies, without any additional learning.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

ACM Reference Format:
Anonymous Author(s). 2023. Adaptive Tracking of a Single-Rigid-Body Character in Various Environments. *ACM Trans. Graph.* 1, 1 (May 2023), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Approaches based on deep reinforcement learning (DRL) are producing significant results in the control of physically simulated characters in recent years. DeepMimic, regarded as one of the pioneering studies, demonstrated exceptional motion quality for complex motions through motion capture data imitation [Peng et al. 2018]. Expanding the repertoire of motions in diverse scenarios beyond mere imitation has been a focus of subsequent research. Numerous efforts have been made in this direction, such as employing large motion datasets [Bergamin et al. 2019] and generative models [Yao et al. 2022], or training controllers to fulfill a broad range of requirements without relying on specific reference motions [Lee et al. 2021]. However, these approaches often required longer learning time due to the need to experience and adapt to various changes in the environment or tasks.

In this study, we propose an alternative and orthogonal approach to the goal of going beyond examples. We express the character as a single rigid body (SRB) by using a simplified physical model called the centroidal dynamics, and train a policy for the SRB character to track a reference motion. Then the learned policy is capable of adapting to various unobserved changes in the environment and controller transitions without any additional learning. Additionally, due to the greatly reduced volume of state and action space, the learning process is sample-efficient. The physical simulation and policy learning are conducted on the SRB character, while final full-body motion is kinematically generated from the simulated SRB states. The policy obtains inputs on the state of the SRB character and produces actions to follow the given reference motion. In the simulation stage, a quadratic programming (QP) solver calculates the contact forces that best achieves the desired acceleration computed from the action and the reference motion. This is then used to update the state of the SRB character.

Our approach defines reinforcement learning (RL) tasks in a less specific (excluding full-body details) and more general form (based on the first principle that character movement is caused by contact forces), thereby allowing the agent to be less dependent on a particular situation of the full-body character. As a result, our method robustly performs transitions by switching or blending of learned policies, and shows the ability to cope with environments that have not been experienced during learning, such as running on uneven terrain, pushing a box, or balancing against external forces, without any additional learning or parameterization. Furthermore, our method is sample-efficient enough to obtain such a adaptive tracking policy in 30 minutes on an ultraportable laptop.

0730-0301/2023/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Earlier physics-based motion generation research manually created a locomotion control algorithm based on error feedback [Coros et al. 2010; Lee et al. 2010; Yin et al. 2007]. Parameter optimization and simple balancing rules were used to generate a wide range of robust motion repertoire [Agrawal et al. 2013; Ha and Liu 2014; Wang et al. 2012]. Controllers based on QP leveraging the equations of motion were developed to enhance robustness [Abe et al. 2007; Da Silva et al. 2008; Kwon and Hodgins 2017]. Non-linear trajectory optimization was used to synthesize physically probable motions for various tasks by simultaneously considering multiple frames, as opposed to the single frame optimization used in QP [Mordatch et al. 2012; Wampler et al. 2014; Ye and Liu 2010]. Model-predictive control (MPC) methodologies performed online optimization, allowing the construction of complex motions in unknown environments [Hämäläinen et al. 2015; Macchietto et al. 2009; Tassa et al. 2012]. However, these MPC techniques that utilized the full-body physical model state space had limitations in terms of motion quality and long-term planning. MPC techniques with simplified models allow interactive controls in more complicated environments, focusing on locomotion tasks [Kwon et al. 2020; Winkler et al. 2018]. Some robotics studies have leveraged simplified models to enhance their locomotion control strategies [Tsounis et al. 2020; Viereck and Righetti 2021].

RL has been widely used to develop controllers for simulated characters. Earlier studies showed the advantage of creating controllers with minimal manual effort by designing simple rewards [Coros et al. 2009; Peng et al. 2015]. DRL expanded the capabilities to tackle various tasks [Brockman et al. 2016; Duan et al. 2016; Liu and Hodgins 2017; Peng et al. 2016; Rajeswaran et al. 2017]. Some studies employed simplified models or multi-level learning for efficient long-term planning [Peng et al. 2017; Reda et al. 2022]. Notably, DeepMimic has achieved impressive motion quality for numerous reference motions [Peng et al. 2018], but faced limitations in controller performance beyond reference motions. Recent research has employed large motion datasets and encoding techniques to diversify the motions generated by RL-based controllers and to improve their generalization capability [Bergamin et al. 2019; Chentanez et al. 2018; Park et al. 2019; Peng et al. 2022; Won et al. 2022; Yao et al. 2022]. Some studies trained controllers to satisfy a wide range of requirements without using corresponding reference motions, for example, transitions between different actions or changes in jump height [Lee et al. 2022, 2021; Peng et al. 2021; Xie et al. 2020; Yin et al. 2021]. Since these full-body simulation-based RL approaches require observation of various changes in environments and intent during the learning process, they often require a longer time for learning. In our method, a policy trained to track only a single reference motion within a relatively short time can respond to various environmental changes without experiencing such cases at training time, due to the flexibility of SRB-based modeling. However, not based on full-body dynamics, our approach does not directly simulate each part of the body. As a result, it has the drawback of not being able to represent contact-rich physical interactions occurring at various parts of the actual full body. Contact forces can be applied only to the manually-classified contact points of the SRB character.

There have been numerous studies that use a large amount of motion data to train models and generate realistic real-time full-body motions purely kinematically [Cho et al. 2021; Holden et al. 2017; Ling et al. 2020; Starke et al. 2019; Zhang et al. 2018]. Our approach can also be viewed as a kinematic controller, as it generates the final full-body motions kinematically. However, unlike these methods that fail to exhibit physical interactions in response to environmental changes, our approach performs accurate physics simulation at the level of the SRB. This allows us to create physically plausible full-body motions by reflecting the changes in the state of the SRB due to variations in the physical environment.

Among various previous studies, the following two studies are the closest to our study. The research by Xie et al. [Xie et al. 2022] has many commonalities with our work in its use of the SRB model, QP, and RL. However, unlike [Xie et al. 2022] which adopted to generate four-legged locomotion by using manually set gait parameters such as foot phase offsets, we focus on creating dynamic and natural motions of a two-legged character, walking, and various other motions, by tracking motion capture data. Whereas a simple Raibert-style heuristic was used to determine foot placement in [Xie et al. 2022], our policies are trained to output desired foot landing positions to ensure balancing.

[Kwon et al. 2020] is closed to our study in terms of using the SRB model for a two-legged character, but has the following key differences: i) [Kwon et al. 2020] generates motions through per-segment trajectory optimization, which takes too much time for smooth real-time performance. Our RL-based system can generate motion at a much faster speed than real-time. ii) To mitigate the runtime performance issue, [Kwon et al. 2020] proposed a supervised learning network that takes pendulum and footstep plans and generates full-body motion. However, it may fail to produce adequate motion in scenarios deviating significantly from the training data, such as unexpected external forces. Our method can train an adaptive controller without additional learning, even in significantly different scenarios from the reference motion used in training. iii) In [Kwon et al. 2020], motion generation occurs per-segment, with planners generating trajectories at half-cycle intervals for a short future horizon. Consequently, if an unexpected external force is applied during runtime, the character may respond with a half-cycle delay. Our policy produces the desired contact position every frame, enabling an immediate response to external forces.

3 SRB CHARACTER AND FRAMES

SRB character is a simplified representation of the key physical characteristics of an articulated character, consisting of a box-shaped rigid body (the gray box in Figure 3) that approximates the inertia of the full-body character, as well as four contact points (the blue spheres) attached on either side of the two feet (the black spheres indicating their centers). The orientation of the box is defined as the orientation of its center of mass, which implies the overall orientation of the character. The mass and inertia of the box are set as the mass (60 kg) of the reference character and its

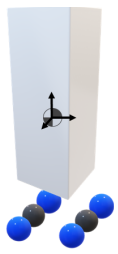


Fig. 3. SRB character

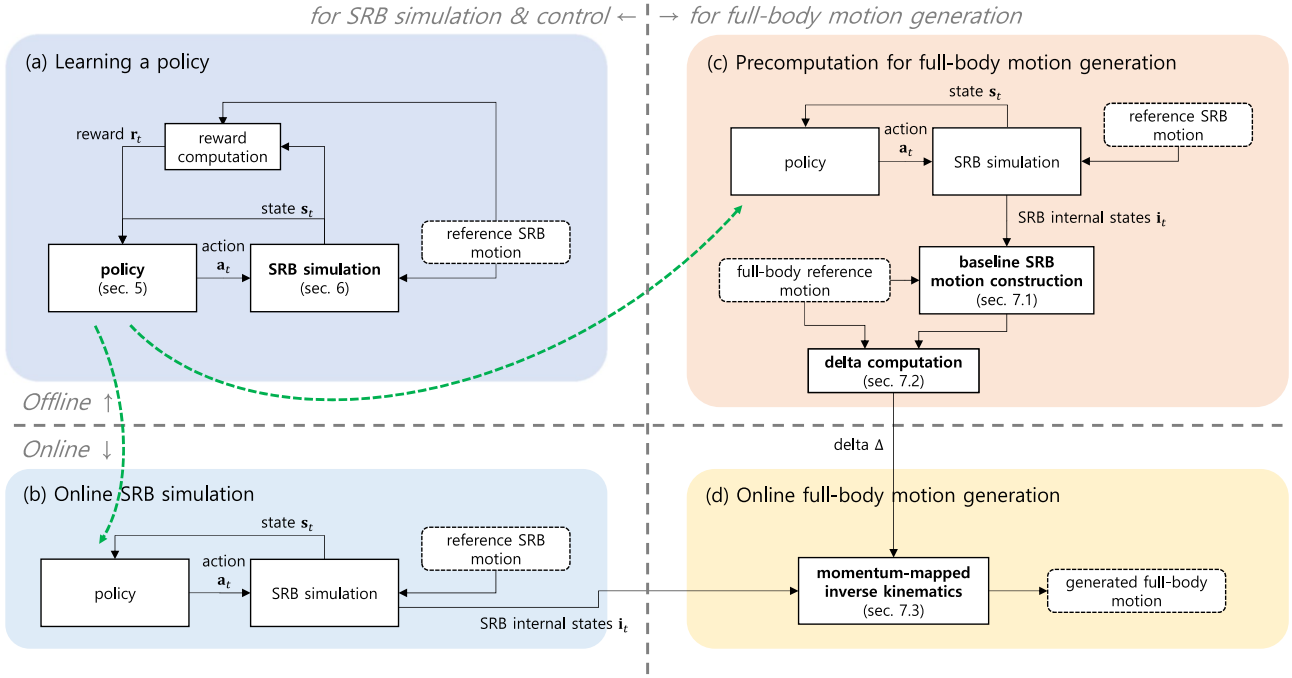


Fig. 2. System overview.

composite rigid body inertia calculated from the default posture (at attention posture).

Reference SRB motion is a reference motion expressed by the SRB character, whose center of mass position and orientation are set to those of the pelvis in the full-body reference motion. The contact timing and position of the feet of the reference SRB motion are set identically to those of the full-body reference motion. The simulated SRB character is controlled to contact the ground at the same time as the reference contact timing.

SRB frames. This paper uses the following three types of reference frames to express various states values:

- **SRB frame** has the SRB character's center of mass as its origin and is attached to the SRB character.
- **Forward-facing SRB frame** has the same origin as the SRB character frame, with its z-axis representing the horizontally projected z-axis of the SRB character frame (forward-facing direction), and its y-axis aligns with the global vertical axis.
- **Projected SRB frame** is obtained by vertically projecting the origin of the forward-facing SRB frame onto the ground or terrain.

Frames corresponding to the reference SRB motion shall be referred to by prefixing "reference" (e.g., "reference SRB frame").

4 OVERVIEW

Figure 2 outlines the overview of the system presented in this paper.

First, a policy is learned for the simulated SRB character to track a reference SRB motion (Figure 2 (a)). In the SRB simulation, the desired acceleration of the SRB character is determined by both the action from the policy and the reference SRB motion. Then, a quadratic programming (QP) problem is solved to find the actual implementable acceleration that can closely match the desired acceleration based on the given internal state of the SRB character. The internal state of the SRB character is updated by integrating the obtained acceleration. The reward is calculated by comparing the updated SRB internal state and the reference SRB motion.

The learned policy can be used to simulate the SRB character at runtime (Figure 2 (b)). It allows not only tracking a single reference motion but also transitions between different motions through policy switching or blending, without requiring additional learning.

To precompute the data required for full-body motion reconstruction (Figure 2 (c)), the baseline SRB motion, which is the tracking result of the reference SRB motion by the learned policy, is first constructed. Then the difference (delta Δ) between baseline SRB motion and the full-body reference motion is calculated.

The simulated SRB motion generated at runtime is converted into full-body motion by momentum-mapped inverse kinematics (Figure 2 (d)). The precomputed delta Δ is combined to produce a more realistic full-body motion by adding the detailed full-body data that were not expressed by the SRB motion.

5 POLICY REPRESENTATION

State $\mathbf{s} \in \mathbb{R}^{21}$ is composed of the following components.

- **Center of mass state** $\mathbf{s}_c \in \mathbb{R}^{11}$ consists of the height, orientation (in unit quaternion), and generalized velocity (angular and linear velocities) of the center of mass of the SRB character, which are all expressed in the projected SRB frame.

- **Foot state** $\mathbf{s}_f^j \in \mathbb{R}^4 (j \in \{\text{left}, \text{right}\})$ consists of the center positions and rotations of each foot of the SRB character in the horizontal plane. These values are expressed in the forward-facing SRB frame.
- **Motion phase** $\psi(t) \in [0, 2\pi]$ indicates what point of the reference motion the current motion of the SRB character corresponds. This is actually stored in state $\mathbf{s}$ as a 2D-vector $(\sin(\psi(t)), \cos(\psi(t)))$.

Action $\mathbf{a} \in \mathbb{R}^{10}$ consists of the following components.

- **Desired foot landing position** $\mathbf{a}_s^j \in \mathbb{R}^2 (j \in \{\text{left}, \text{right}\})$ in the forward-facing SRB frame. Note that the vertical position is not included.
- **Desired velocity of center of mass** $\mathbf{a}_v \in \mathbb{R}^6$ refers to the desired relative linear and angular velocities of the center of mass of the SRB character against that of the reference SRB motion. This is expressed with respect to the SRB frame.

Reward r_t at each step t is configured as below:

$$r_t = w^s r_t^s - w^m r_t^m. \tag{1}$$

Here, r_t^s is the alive reward given when the episode does not end and r_t^m is the mimic reward term that guides the SRB character to closely follow the reference SRB motion.

The mimic reward term r_t^m is calculated as follows:

$$r_t^m = w^p r_t^p + w^e r_t^e. \tag{2}$$

r_t^p is a posture reward term, which gives higher rewards when the states of the centers of mass of both SRB character and reference SRB motion are alike, which is calculated as follows:

$$\begin{aligned} r_t^p =& w^{p_1} \|\Delta \mathbf{p}_t - \Delta \hat{\mathbf{p}}_{\psi(t)}\| + w^{p_2} \|\Delta \mathbf{R}_t \ominus \Delta \hat{\mathbf{R}}_{\psi(t)}\| \\ &+ w^{p_3} \|\mathbf{p}_t - \hat{\mathbf{p}}_{\psi(t)}\| + w^{p_4} \|\mathbf{R}_t \ominus \hat{\mathbf{R}}_{\psi(t)}\|. \end{aligned} \tag{3}$$

Here, $\Delta \mathbf{p}_t$ and $\Delta \mathbf{R}_t$ respectively refer to the changes in the center of mass' position and orientation of the SRB character between time t and $t+1$, while $\Delta \hat{\mathbf{p}}_{\psi(t)}$ and $\Delta \hat{\mathbf{R}}_{\psi(t)}$ refer to those changes in the reference SRB motion between the corresponding time points. $\mathbf{p}_t$ and $\mathbf{R}_t$ respectively refer to the position and orientation of the SRB character's center of mass at time t, while $\hat{\mathbf{p}}_{\psi(t)}$ and $\hat{\mathbf{R}}_{\psi(t)}$ refer to those of the reference SRB motion. $\|\mathbf{R}_1 \ominus \mathbf{R}_2\|$ refers to the minimum angle of rotation between the two rotation matrices. To compare the difference in the character height and the leaning angle of the body, the above terms are expressed in the projected frame of the respective character, that is, the projected SRB frame or the reference projected SRB frame.

r_t^e is an end-effector reward, which has a higher value when the support foot positions of both the SRB character and reference SRB motion are alike. This reward is only applied to the foot in contact with the ground in the reference SRB motion and is calculated as follows:

$$r_t^e = \sum_j \sum_k \|\mathbf{c}_t^{jk} - \hat{\mathbf{c}}_{\psi(t)}^{jk}\|^2, \tag{4}$$

where $\mathbf{c}_t^{jk}$ refers to the position of the k^{th} contact point ($k \in \{\text{toe}, \text{heel}\}$) of the j^{th} foot ($j \in \{\text{left}, \text{right}\}$) of the SRB character at time t with respect to the SRB frame, while $\hat{\mathbf{c}}_{\psi(t)}^{jk}$ refers to that of the corresponding contact point of the reference SRB motion at the corresponding time, expressed with respect to the reference SRB frame. Note that the swing foot is excluded from this term because the SRB character does not directly depict the full-body character's actual swing motion of the foot.

6 SRB SIMULATION

This section will explain the internal state of the SRB character at time t, where the input $\mathbf{s}_t$ for the RL policy is extracted, followed by the simulation process by QP that computes the internal state at time $t+1$.

6.1 Internal state of SRB character

The internal state of the SRB character consists of the center of mass' position and orientation, their time derivatives, and the position and orientation of the two feet.

Center of mass state. The global position $\mathbf{p}_t^{\{g\}}$ and global orientation $\mathbf{R}_t^{\{g\}}$ of the SRB character's center of mass is used to calculate the SRB frame $\mathbf{T}_t \in \text{SE}(3)$ at the center of mass as follows:

$$\mathbf{T}_t = \text{F}\left(\mathbf{R}_t^{\{g\}}, \mathbf{p}_t^{\{g\}}\right), \tag{5}$$

where F is a function that outputs the rigid-body transformation from the position and rotation information.

The time derivatives of the center of mass position and orientation are expressed as the following generalized velocity $\dot{\mathbf{q}}_t \in \text{se}(3)$, which indicates the spatial velocity in body frame:

$$[\dot{\mathbf{q}}_t] = \mathbf{T}_t^{-1} \cdot \dot{\mathbf{T}}_t, \tag{6}$$

where $\dot{\mathbf{T}}_t$ is the time derivative of $\mathbf{T}_t$, and $[\cdot]$ is the operator that converts se(3) in the 6-vector coordinates into a 4x4 matrix format.

Foot state consists of the global position $\mathbf{f}_t^j$ and orientation $\mathbf{F}_t^j$ of each foot. Their specific meaning varies depending on whether the j^{th} foot is in the swing state, determined by the touch down and off timings of that foot in the reference SRB motion. $\mathbf{f}_t^j$ is a horizontal position and $\mathbf{F}_t^j$ is expressed as only the rotational component against the vertical axis.

When the j^{th} foot is in a swing state, $\mathbf{f}_t^j$ moves continuously to the desired foot landing position $\mathbf{a}_s^j$. This $\mathbf{a}_s^j$ is not directly suitable as a basis for reproducing the continuous movement of the swing foot since it changes discontinuously over time. Therefore, we use a LQR filter presented in [Hwang et al. 2017], to compute continuous $\mathbf{f}_t^j$ from the discontinuous $\mathbf{a}_s^j$. Similarly, by the LQR filter, $\mathbf{F}_t^j$ moves continuously to the desired foot landing orientation which is directly obtained from the reference SRB motion. Further details of LQR filtering are described in the supplementary material.

When the j^{th} foot is in a contact state, $\mathbf{f}_t^j$ refers to the actual global position of the foot, which remains fixed for the contact duration. The moment the j^{th} foot changes its state from swing to contact, the position of the foot and its rotation against the vertical axis (at the very last moment in the swing phase) become $\mathbf{f}_t^j$ and $\mathbf{F}_t^j$ for the contact duration. Action $\mathbf{a}_s^j$, which is output from the policy for the foot in the contact state, is ignored and not used.

6.2 QP simulation

The QP simulator takes the SRB character's center of mass state $\mathbf{T}_t$, $\dot{\mathbf{q}}_t$, its relative desired velocity $\mathbf{a}_v$, and contact point $\mathbf{c}_t$ as inputs to calculate its internal state in the next time step. To this end, the desired acceleration is calculated first based on the reference SRB motion and $\mathbf{a}_v$. The actual achievable acceleration that maximally satisfies this desired acceleration and the corresponding contact force are calculated from the QP formulated as:

$$\min_{\ddot{\mathbf{q}},\lambda} Q\left(\ddot{\mathbf{q}},\lambda\right) \tag{7}$$

such that

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{b} = \mathbf{J}_f^T\mathbf{F}_e + \mathbf{J}_c^T\mathbf{B}\lambda, \tag{8}$$

$$\lambda \geq \mathbf{0}. \tag{9}$$

Here, $\ddot{\mathbf{q}}$ refers to the generalized center of mass acceleration, λ : the coefficient for the friction cone basis, $\mathbf{F}_e$: the external force applied on the SRB character, $\mathbf{J}_c$: the Jacobian matrices related to the velocity of the contact point $\mathbf{c}_t$, $\mathbf{J}_f$: Jacobian matrices related to the velocity of the point where the external force is applied, and $\mathbf{B}$: the friction cone basis vectors. $\mathbf{F}_e$ is set to $\mathbf{0}$ within the RL process, but is given non-zero values when external forces are applied during the runtime simulations. In short, the acceleration and contact force that minimizes the objective function Q are obtained while satisfying the equation of motion constraints and the constraints on the linearized basis of contact force [Ellis et al. 2007; Kwon and Hodgins 2017].

The objective function Q is defined as follows:

$$Q = \left\|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d\right\|^2 + w_\lambda \left\|\lambda\right\|^2. \tag{10}$$

Here, $\ddot{\mathbf{q}}_d$ refers to the desired acceleration. w_λ refers to the weight for the contact force term, which was set to 0.001 to ensure robust control during our experiments.

Action $\mathbf{a}_v$ is used in the following to calculate the desired velocity $\dot{\mathbf{q}}_d$, through which the desired acceleration $\ddot{\mathbf{q}}_d$ is obtained.

$$\dot{\mathbf{q}}_d = \hat{\dot{\mathbf{q}}}_{\psi(t)} + \mathbf{a}_v, \tag{11}$$

where $\hat{\dot{\mathbf{q}}}_{\psi(t)}$ indicates the linear and angular velocities of the reference SRB motion's center of mass, and action $\mathbf{a}_v$ refers to the desired relative linear and angular velocities of the SRB character's center of mass against the reference SRB motion's center of mass.

The desired acceleration $\ddot{\mathbf{q}}_d$ is calculated as follows by taking the difference between the desired and current position and velocity.

$$\ddot{\mathbf{q}}_d = a\log\left(\mathbf{T}_t^{-1}\hat{\mathbf{T}}_{\psi(t)}\right) + b\left(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}_t\right), \tag{12}$$

where $\mathbf{T}_t$ is the current SRB frame, while $\hat{\mathbf{T}}_{\psi(t)}$ is the reference SRB frame at the corresponding phase. $\dot{\mathbf{q}}_t$ is the current generalized velocity, while $\dot{\mathbf{q}}_d$ is the desired generalized velocity. The log function converts a rigid body transformation matrix $\in \mathrm{SE}(3)$ into a generalized velocity $\in \mathrm{se}(3)$. In our experiments, PD gains a and b were set at 120 and 35, respectively.

By integrating the calculated $\ddot{\mathbf{q}}$, $\mathbf{T}_t$ and $\dot{\mathbf{q}}_t$ are updated, and the foot state $\mathbf{f}_t^j$ and $\mathbf{F}_t^j$ are then updated as described in Section 6.1. The next time step's state $\mathbf{s}_{t+1}$ can be derived from these values. The simulation process through the QP solver helps the SRB character effectively find the contact force that best achieves the desired acceleration at each moment, helping to learn a robust policy.

6.3 Motion phase adjustment

At each SRB simulation timestep, the rate at which motion phase $\psi(t)$ changes is adjusted in two aspects. First, the phase change rate is increased with increasing locomotion speed to prevent excessively long strides, which is inspired by [Kwon and Hodgins 2017]. Second, when the character experiences a large unexpected force, it may deviate from the specified contact timings in the reference SRB motion, resulting in unstable control. To address this, the phase rate is decreased to delay touchdown when it is expected to happen before the specified time, and increased to allow for an earlier touchdown when it is expected to happen later. Further details of these adjustments are provided in the supplementary material.

7 FULL-BODY MOTION GENERATION

To generate a full-body motion from a simulated SRB character motion, we create a baseline SRB motion and calculate the kinematic and dynamic differences compared to the full-body reference motion. During real-time simulation, these precomputed differences are applied to the simulated SRB motion, obtaining target values for momentum-mapped inverse kinematics (MMIK) to generate the full-body motion by solving MMIK.

7.1 Baseline SRB motion construction

After obtaining a trained policy, it is used to simulate the SRB character to generate multiple simulated cycles, serving as the baseline SRB motion. Then the COM trajectory and the feet states of the baseline SRB motion are transformed offline. This transformation aligns the horizontal start and end points of the COM trajectory with those of the reference COM trajectory per motion cycle (Figure 6), while ensuring no inclination of the trajectory by considering only the rotation against the y-axis.

7.2 Delta computation

The delta between the baseline SRB motion and the full-body reference motion is computed to incorporate the reference full-body details into a reconstructed full-body motion during runtime. The average differences in the center of mass frame, foot contact points, centroidal velocity between them are calculated for each different motion phase and stored as delta Δ.

7.3 Momentum-mapped inverse kinematics

During runtime, the simulated SRB motion is transformed into the full-body motion using momentum-mapped inverse kinematics (MMIK) presented in [Kwon et al. 2020]. We improve the previous form of MMIK to use the additional input of the precomputed delta Δ. The inputs are the full-body reference motion, Δ, and simulated SRB states and it outputs the full-body pose closest to the given full-body reference pose, while satisfying the footstep position, COM configuration, and its time derivative of the error-adjusted simulated SRB motion using Δ. Further details for computing Δ and MMIK are described in the supplementary material.

8 EXPERIMENTAL RESULTS

All experiments and policy training were conducted on a ultra-portable laptop that runs on 16GB RAM, M1 CPU. All motions were

simulated at 60 Hz. Most controllers converged within 30 minutes, around 3M time steps, which is equivalent to about 14 hours of simulated time. The most challenging *Sprint jumps* controller converged within 1-hour, and even after just 30 minutes, a policy performing the motion reliably was obtained. Our approach enables the character to perform a range of reference motions, including those involving drastic changes in direction, such as *Sprint* or *Sharp turns*, or those requiring strong force (Figure 7).

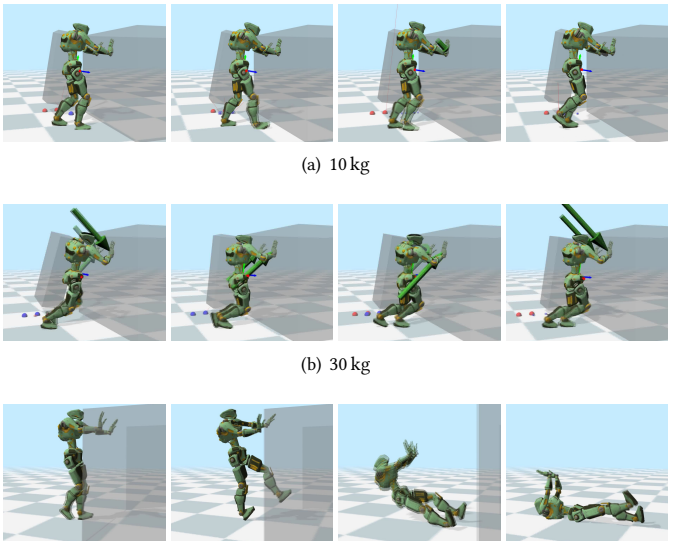
We demonstrate how effectively our controllers respond to diverse, unobserved changes. Note that all controllers used in the experiments were trained simply to mimic reference motions on flat ground (except Section 8.4), without any additional training for adapting to those changes. In some experiments, we conducted comparisons with DeepMimic [Peng et al. 2018]. DeepMimic has the advantage of simulating full-body dynamics, but it is designed to imitate detailed poses of a full-body reference motion, limiting its adaptability to unfamiliar situations. As a result, DeepMimic exhibited less adaptability compared to our method. However, it is important to note that DeepMimic and our method belong to different categories and have different purposes. These comparative experiments were not intended to claim that our method is always superior to DeepMimic, but rather to showcase the advantages of our simplified physics-based approach compared to widely used full-body-based methods. Further implementation and training details, and additional experimental results such as sample efficiency are provided in the supplementary material.

8.1 Controller transitions and interpolations

Transitions by switching. Our method allows for transitioning between different controllers simply by switching pre-trained policies (Figure 8). At the transition point, the policy π_A, reference SRB motion $\hat{\mathbf{m}}_A$, and precomputed delta Δ_A, of the Controller A are immediately replaced with those of the Controller B. We use a motion stitching technique that gradually reduces the difference between the reference motions and deltas at the transition point and reflects it over time. Note that the transition was successful (no falling within 20 seconds after the switch) when the difference in movement speed was within 30%, such as *Sprint* and *Sprint jumps* or *Sprint* and *Sharp turns*, or when both controllers are exceptionally robust (e.g. *Sprint* and *Run*).

Transitions by blending. Transitions between controllers with larger speed differences can be achieved through blending, where Controller A gradually transitions to Controller B by linearly increasing the interpolation weight t from 0 to 1 (for 1 sec). The actions from π_A and π_B, $\hat{\mathbf{m}}_A$ and $\hat{\mathbf{m}}_B$, Δ_A and Δ_B, as well as the change rates of $\psi_A(t)$ and $\psi_B(t)$, the remaining time in $\hat{\mathbf{m}}_A$ and $\hat{\mathbf{m}}_B$ until each j^{th} foot touches down $t_A^{d,j}$, $t_B^{d,j}$ and until touch off $t_A^{o,j}$, $t_B^{o,j}$ are interpolated. With this method, smooth transitions were achieved between controllers with significant differences in movement speeds and styles, such as *Fast walk* (1.8 m/s) and *Run* (3.6 m/s) (Figure 9).

Interpolation. The interpolation method allows for creating new interpolated controllers at constant ratios between two controllers, apart from transitioning between two controllers (Figure 10).



(a) 10 kg

(b) 30 kg

(c) 10 kg - DeepMimic

Fig. 4. Pushing a box of varying weights.

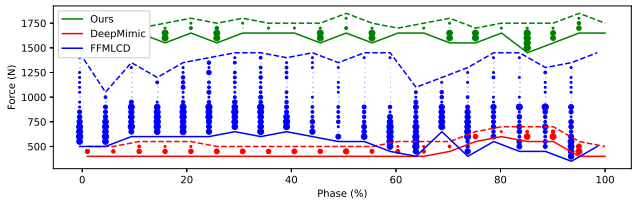
8.2 Environmental adaptations

Uneven terrain. Our controllers trained on even terrain was able to create locomotion on uneven terrain with a gradient of a maximum 30 degrees (Figure 11). This is achieved without the need for any additional modifications to the algorithm, and the results are based on the inherent adaptability of the SRB-based policy. In contrast, DeepMimic was observed to struggle even on gentle slopes, as it focuses on tracking full-body poses, resulting in the swing foot tripping.

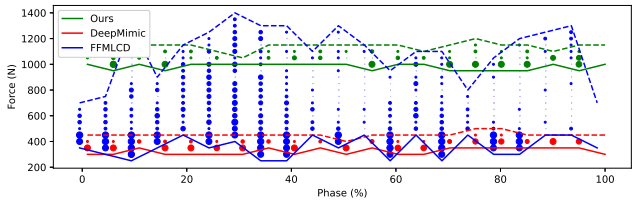
Push boxes. The policy was trained to track an edited version of the *Walk* reference motion, where both arms were raised forward. The learning environment did not include any boxes, and therefore, the policy did not receive any box information. As shown in Figure 4, the character was able to push boxes of different weights using different motions, by leaning forward to exert more force when the box is heavier. Adjusting the foot landing position slightly behind the desired landing position in the action output of the policy resulted in a stronger pushing behavior when dealing with heavy boxes. In contrast, a DeepMimic controller trained to track the same reference motion was unable to push the box. Our SRB-based model achieved comparable results to a previous study [Lee et al. 2021] without the need for a policy parameterized by box weights.

8.3 Comparision for external pushes

We compared the balance maintenance level of characters controlled by our method, FFMLCD [Kwon et al. 2020], and DeepMimic [Peng et al. 2018]. As reported in a previous study [Lee et al. 2015], the response to external force can vary depending on the push timing, so we performed push experiments throughout the entire motion phase rather than at a specific point. The experiments used *Sprint* controllers and were performed at 20 evenly spaced phase points.



(a) Push from side



(b) Push from back

Fig. 5. Plots for external pushes. Dotted line: the smallest external force at which the character always loses balance. Solid line: the largest external force at which the character maintains balance at each phase without falling. Circles: the ratio of successful balance maintenance (no falling within 20 seconds after the push) out of 10 trials at points where both balance maintenance and loss occur, with its size representing the proportion of successful balance maintenance.

External force was applied to the character's root from the side or behind for 0.2 seconds, ranging from 100 N to 2000 N in 50 N increments. Each force level was applied 10 times at each point.

As depicted in Figure 5, our method shows a higher minimum force at which the character always loses balance (dotted lines) in side pushes, while it is comparable to FFMLCD in back pushes. However, from a practical standpoint, the important factor is the force magnitude at which the character maintains balance without falling (solid lines). In this aspect, our method exhibits a force magnitude approximately three times higher than FFMLCD in both push cases. This is because our method, utilizing RL, generates desired footstep positions at each frame to maintain balance, while FFMLCD relies on non-linear optimization-based planning that may occasionally struggle to find optimal solutions in challenging situations. Moreover, FFMLCD performs this planning at each half cycle instead of each frame, which further limits its responsiveness. Due to the limited adaptability caused by full-body tracking, DeepMimic easily fell even with weaker forces. All controllers were more stable when pushed from the side compared to the back, because pushing from the back would cause the body to lean forward and the swing foot to suddenly hit the ground, resulting in a fall or excessive contact force.

8.4 Interactive control

We also conducted experiments to enable more useful applications by allowing users to interactively control the facing direction of the character. For this purpose, we added a scalar δ_y, representing the difference between the desired facing direction and the current SRB facing direction, to the inputs of the policy network. The policy is then trained using the following reward r_t^c:

$$r_t^c = \exp(-r_t) \cdot \exp(-w^c \cdot (\cos(\delta_y) - 1)), \tag{13}$$

where r_t represents the reward defined in Equation 1, and w^c denotes the weight.

9 DISCUSSION

In this paper, we have presented a framework that can learn an adaptive tracking policy in a sample-efficient manner using a SRB character simplifying a full-body character. The QP-based SRB simulation allows efficient learning of a policy, and the simulated SRB motion can be converted back into a full-body motion using precomputed delta and a momentum-mapped inverse kinematics solver.

Thanks to the less detailed SRB model-based learning, our method enables fast policy learning with fewer samples compared to traditional full-body methods. Moreover, the learned policy exhibits adaptability in various environments. However, it is worth noting that our method has limitations in accurately representing contact-rich scenarios, unlike full-body physics simulations. For example, it does not explicitly represent collisions between body parts and the ground when the character falls. In our method, "falling" refers to when the center of mass is too close to the ground, making the solution of MMIK meaningless. In practice, we can partially address this limitation by switching to a full-body simulation using PD-servo to generate a more natural fall when the character begins to lose balance.

In our method, each controller is trained based on a single short reference motion clip. Instead of relying on a large number of reference motions, our approach prioritizes maximizing the number of situations that the same controller can handle without additional learning. Once such an adaptability is achieved, we believe that it can be straightforwardly extended in the direction of enabling a wider range of motions by using more complex network structures supporting unorganized motion capture datasets, such as VAE or RNN [Park et al. 2019; Yao et al. 2022]. The proposed framework allows only indirect adjustment of the contact timing based on the adjustment of the speed of phase progression. This can negatively affect robustness because a limited range of contact timing adjustment interferes with more flexible adaptation of the character to external force. However, unlike DeepMimic-style controllers which use fixed contact timing, the contact timing is easily adjustable in our framework that employs virtual legs, and thus it would help to improve the robustness of the controllers by adjusting the contact timing in a similar way as tested in [Reda et al. 2022]. One of the computational bottlenecks in our learning process is solving the QP. As future work, it would be beneficial to explore the use of a GPU-based QP solver, such as the one presented in [Xie et al. 2022], to further accelerate learning.

REFERENCES

Yeuhi Abe, Marco Da Silva, and Jovan Popović. 2007. Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 249–258.

Shailen Agrawal, Shuo Shen, and Michiel van de Panne. 2013. Diverse motion variations for physics-based character animation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 37–44.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (Nov. 2019), 206:1–206:11.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-Based Motion Capture Imitation with Deep Reinforcement Learning. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '18)*. Article 1, 10 pages.

Kyungmin Cho, Chaelin Kim, Jungjin Park, Joonkyu Park, and Junyong Noh. 2021. Motion recommendation for online character control. *ACM Transactions on Graphics* 40, 6 (2021).

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust task-based control policies for physics-based characters. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009), 1–9.

Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2010. Generalized biped walking control. *ACM Transactions On Graphics (TOG)* 29, 4 (2010), 1–9.

Marco Da Silva, Yeuhi Abe, and Jovan Popović. 2008. Simulation of human motion data using short-horizon model-predictive control. In *Computer Graphics Forum*, Vol. 27. 371–380.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. 1329–1338.

Jane Ellis, Harald Winkler, Jan Corfee-Morlot, and Frédéric Gagnon-Lebrun. 2007. CDM: Taking stock and looking forward. *Energy policy* 35, 1 (2007), 15–28.

Sehoon Ha and C Karen Liu. 2014. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–11.

Perttu Hämäläinen, Joose Rajamäki, and C Karen Liu. 2015. Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–13.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Transactions on Graphics* 36, 4, Article 42 (2017), 13 pages.

Jaepyung Hwang, Kwanguk Kim, Il Hong Suh, and Taesoo Kwon. 2017. Performance-based animation using constraints for virtual object manipulation. *IEEE computer graphics and applications* 37, 4 (2017), 95–102.

Taesoo Kwon and Jessica K Hodgins. 2017. Momentum-mapped inverted pendulum models for controlling dynamic human motions. *ACM Transactions on Graphics (TOG)* 36, 1 (2017), 1–14.

Taesoo Kwon, Yoonsang Lee, and Michiel Van De Panne. 2020. Fast and flexible multilegged locomotion using learned centroidal dynamics. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 46–1.

Seyoung Lee, Jiye Lee, and Jehee Lee. 2022. Learning Virtual Chimeras by Dynamic Motion Reassembly. *ACM Transactions on Graphics* 41, 6 (2022), 182:1–182:13.

Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. 2021. Learning a family of motor skills from a single motion clip. *ACM Transactions on Graphics* 40, 4 (July 2021), 93:1–93:13.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Trans. Graph.* 29, 4 (2010), 1–8.

Yoonsang Lee, Kyungho Lee, Soon-Sun Kwon, Jiwon Jeong, Carol O'Sullivan, Moon Seok Park, and Jehee Lee. 2015. Push-recovery Stability of Biped Locomotion. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 180:1–180:9.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion VAEs. *ACM Transactions on Graphics* 39, 4 (2020).

Libin Liu and Jessica Hodgins. 2017. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 1–14.

Adriano Macchietto, Victor Zordan, and Christian R. Shelton. 2009. Momentum control for balance. *ACM Transactions on Graphics* 28, 3 (July 2009), 80:1–80:8.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics* 38, 6 (2019), 205:1–205:11.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. 2015. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. 2016. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* 36, 4 (2017).

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics* 41, 4 (July 2022), 94:1–94:17.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics* 40, 4 (2021), 144:1–144:20.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

Daniele Reda, Hung Yu Ling, and Michiel van de Panne. 2022. Learning to Brachiate via Simplified Model Imitation. In *ACM SIGGRAPH 2022 Conference Proceedings (SIGGRAPH '22)*. Article 24, 9 pages.

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural State Machine for Character-Scene Interactions. *ACM Transactions on Graphics* 38, 6, Article 209 (2019), 14 pages.

Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4906–4913.

Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. 2020. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3699–3706.

Julian Viereck and Ludovic Righetti. 2021. Learning a centroidal motion planner for legged locomotion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4905–4911.

Kevin Wampler, Zoran Popović, and Jovan Popović. 2014. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. 2018. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1560–1567.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics* 41, 4 (2022), 96:1–96:12.

Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel van de Panne. 2022. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model. In *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 523–539.

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills. *Computer Graphics Forum* 39, 8 (2020), 213–224.

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Transactions on Graphics* 41, 6 (2022), 183:1–183:16.

Yuting Ye and C. Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. *ACM Trans. Graph.* 29, 4 (2010), 1–9.

KangKang Yin, Kevin Loken, and Michiel Van de Panne. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 105–es.

Zhiqi Yin, Zeshi Yang, Michiel Van De Panne, and Kangkang Yin. 2021. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics* 40, 4 (July 2021), 91:1–91:17.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-Adaptive Neural Networks for Quadruped Motion Control. *ACM Transactions on Graphics* 37, 4 (2018).

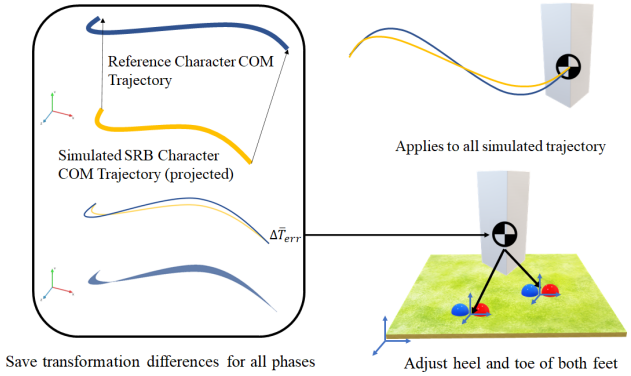
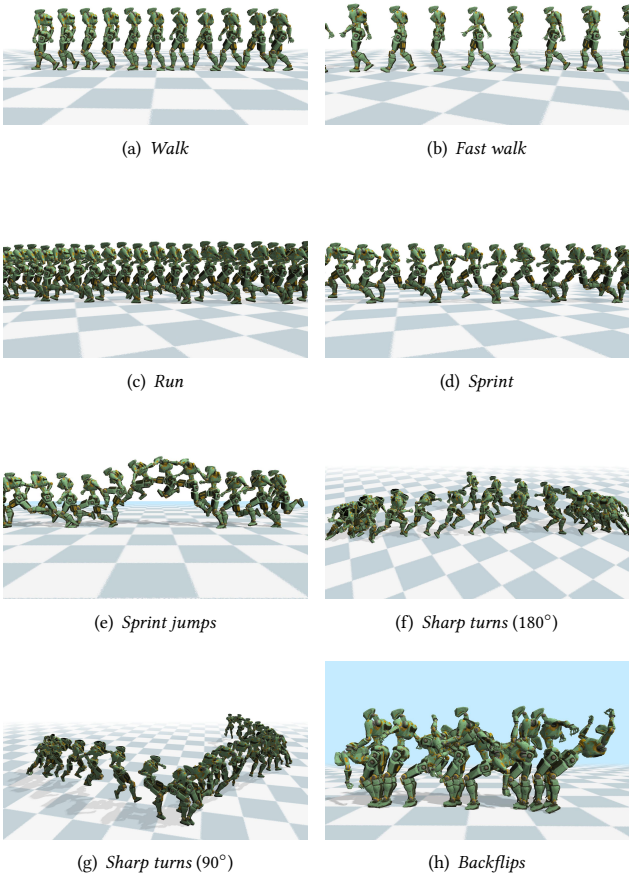


Fig. 6. Conceptual depiction of aligning COM trajectory and calculating COM frame delta.



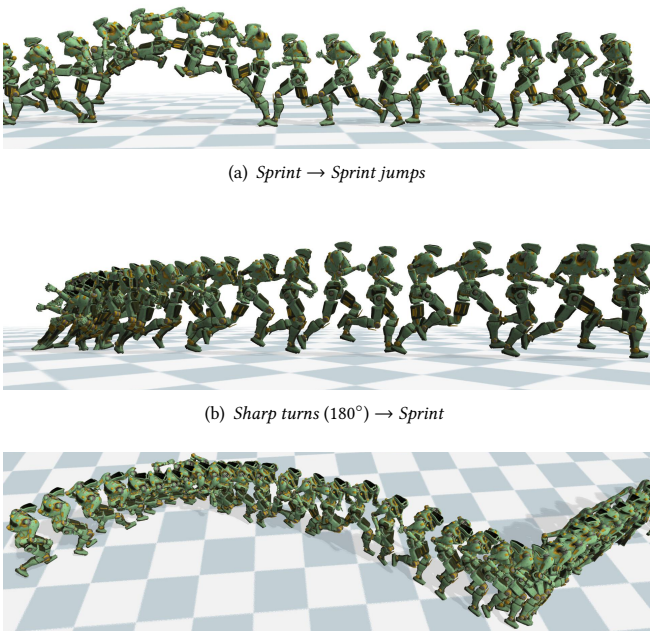
(a) *Walk* (b) *Fast walk*

(c) *Run* (d) *Sprint*

(e) *Sprint jumps* (f) *Sharp turns* (180°)

(g) *Sharp turns* (90°) (h) *Backflips*

Fig. 7. Tracking various motions.



(a) *Sprint* → *Sprint jumps*

(b) *Sharp turns* (180°) → *Sprint*

(c) *Sharp turns* (-90°) → *Sprint* → *Sharp turns* (90°)

Fig. 8. Transitions by controller switching.

Fig. 9. Blending of controllers. Bi-directional transitions of motions are made between *Fast walk* (1.8 m/s) and *Run* (3.6 m/s) through the blended controller.

(a) *Run* controller ($t = 0$, 3.6 m/s)

(b) Interpolated controller 1 ($t = 0.25$, 4.2 m/s)

(c) Interpolated controller 2 ($t = 0.5$, 4.9 m/s)

(d) Interpolated controller 3 ($t = 0.75$, 5.3 m/s)

(e) *Sprint* controller ($t = 1$, 6.0 m/s)

Fig. 10. Interpolated controllers. This figure shows the simulated motions from *Run* and *Sprint* controllers and the interpolated controllers between the two. The speed in parentheses is the moving speed of the simulated character.

(a) *Walk* on uneven terrain

(b) *Sprint* on uneven terrain

Fig. 11. Adaptation for uneven terrain.