Interactive Design of Stylized Walking Gaits for Robotic Characters

MICHAEL A. HOPKINS*, Disney Research, USA
GEORG WIEDEBACH*, Disney Research, USA
KYLE CESARE, Walt Disney Imagineering R&D, USA
JARED BISHOP, Walt Disney Imagineering R&D, USA
ESPEN KNOOP, Disney Research, Switzerland
MORITZ BÄCHER, Disney Research, Switzerland

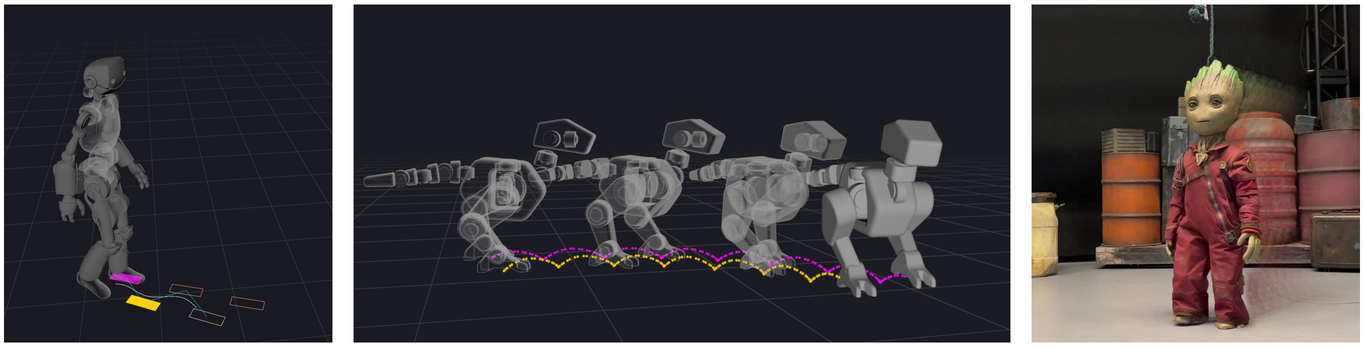

Fig. 1. **Stylized gaits designed with our interactive procedural animation technique.** An adult-sized humanoid robot walks with a casual style in sim (left). Controller reference trajectories for planned Center of Mass (white) and Center of Pressure (teal) are rendered on the ground plane along with anticipated footholds. A small-scale, dino-inspired character walks and turns in sim (center). The physical Groot robot walks along a user-defined path (right).

Procedural animation has seen widespread use in the design of expressive walking gaits for virtual characters. While similar tools could breathe life into robotic characters, existing techniques are largely unaware of the kinematic and dynamic constraints imposed by physical robots. In this paper, we propose a system for the artist-directed authoring of stylized bipedal walking gaits, tailored for execution on robotic characters. The artist interfaces with an interactive editing tool that generates the desired character motion in real-time, either on the physical or simulated robot, using a model-based control stack. Each walking style is encoded as a set of sample parameters which are translated into whole-body reference trajectories using the proposed procedural animation technique. In order to generalize the stylized gait over a continuous range of input velocities, we employ a phase-space blending strategy that interpolates a set of example walk cycles authored by the animator while preserving contact constraints. To demonstrate the utility of our approach, we animate gaits for a custom, free-walking robotic character, and show, with two additional in-simulation examples, how our procedural animation technique generalizes to bipeds with different degrees of freedom, proportions, and mass distributions.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; **Physical simulation**.

*Both authors contributed equally to this research.

Additional Key Words and Phrases: procedural animation, optimal control, robotic characters

ACM Reference Format:
Michael A. Hopkins, Georg Wiedebach, Kyle Cesare, Jared Bishop, Espen Knoop, and Moritz Bächer. 2024. Interactive Design of Stylized Walking Gaits for Robotic Characters. In . ACM, New York, NY, USA, Article 137, 15 pages. https://doi.org/10.1145/3658227

1 INTRODUCTION

While humanoid robotics is an evolving field, a number of groups have demonstrated stable, capable walking on a variety of hardware platforms. Despite this progress, stylized walking remains a challenging problem for physical robots. This is largely due to a lack of authoring tools that provide artists with fine-grain control of a robot's motion while respecting the inherent limitations of the physical system.

The field of physics-based character animation shares a common goal of enabling characters to perform expressive motions that obey the laws of physics. While recent imitation learning approaches have led to impressive results, the current focus is rather on skill than style, and existing techniques do not allow artists to directly author content. Moreover, physical robots are subject to actuator limitations and model complexities commonly unaddressed by these techniques. As a result, the expressivity and believability of animations that we can achieve on physical robots often lags behind associated results.

Procedural animation of robotic characters is challenging for two reasons: First, walking gaits must satisfy the kinematic and dynamic constraints inherent to whole-body locomotion. Second, walking gaits vary as a function of the character's velocity, and an authored style must generalize to a continuous range of velocities for forward and reverse walking, turning, strafing, etc.

To address these challenges, we aim to provide animators with an easy-to-use gait editor that limits the achievable design space to motions that satisfy critical feasibility constraints, while simultaneously exposing a highly expressive set of animation parameters that allow artists to design a wide range of periodic gaits, including asymmetric walk cycles (e.g., a stylized limp for an injured character). To this end, we propose an interactive, procedural animation technique that interfaces directly with a model-based control stack, enabling expressive walking for simulated and physical robotic hardware. Compared to other procedural animation approaches, it is tailored to the specific needs of designing stylized gaits for bipedal robotic characters. Using a custom, constraint-aware interpolation scheme, our system produces omnidirectional walking styles over a range of velocities from a small sample set of authored walk cycles.

As we demonstrate with three robotic characters, artists can author gaits for bipeds that walk along an arbitrary path while expressing a custom style, e.g., a happy or a sneaky walk, taking velocity commands from a joystick as input (Fig. 1). The three characters vary in proportions, mass distribution, and number of degrees of freedom. We interface with an existing robotic character [Panzarino 2021] to demonstrate direct authoring on hardware, and show how quickly an artist can design character-specific stylized cycles with additional in-simulation results on two realistic robot models.

The primary contribution of our work is the interactive authoring workflow enabled by the tightly-integrated gait editor, procedural animation, and control strategy. In contrast to related work, our technique is entirely real-time and does not require offline trajectory optimization or a training step. As a result the proposed system can be used to author content directly on the robot, enabling interactive iteration of a target walking style. The authors are not aware of an existing alternative that permits real-time editing of stylized gaits on physical robots with the level of detail presented by the proposed system, which supports diverse gait timings, upper-body motion, heel/toe behavior, and style variation over a continuous range of velocities. This is only made possible by combining diverse techniques from the fields of procedural animation and humanoid robotics.

2 RELATED WORK

Computer Animation. Skeletal animation, keyframing, and procedural animation are widely used techniques to author artistic motions for virtual characters [Multon et al. 1999]. Data-driven approaches have also been used to generate stylized motions, given a functional target and style reference [Grochow et al. 2004]. Prior work on the procedural blending of gait references from motion capture has shown that it is possible to blend gaits with different timings and walking paths [Holden et al. 2017; Kovar and Gleicher 2003]. Our work employs similar concepts to blend diverse walk cycles, e.g. mapping time to a monotonically-increasing parameter; however, while our method belongs to the category of procedural animation techniques, standard methods are not directly applicable in the robotics domain, as they don't account for the physics of the robot.

Physics-based Characters. Physics-based character animation takes into account the dynamics of a character, mapping high-level control inputs to low-level actuation torques [Hertzmann and Zordan 2011]. Pioneering work focused on model-based approaches, crafting custom controllers for walking and other tasks [Coros et al. 2010; Hodgins et al. 1995; Lee et al. 2010]. Work by Karim et al. [2013] demonstrated a pipeline for generating stylized procedural motions for legged creatures in a simulated environment, and Zhao et al. [2005] developed an interactive user interface for prototyping sports animation.

More recently, learning-based approaches have started to dominate. These methods often train policies to imitate desired motion using Reinforcement Learning (RL) [Lee et al. 2021; Peng et al. 2017; Won et al. 2022]. While early methods require reward engineering, recent techniques combine RL with adversarial approaches to extract skills from unstructured motion data [Peng et al. 2022; Yao et al. 2022a]. Applying such techniques to animate robots is a promising research direction but challenging for interactive and precise full-body authoring of content directly on hardware as addressed in this paper.

Animation in Robotics. While at first glance similar to physics-based character animation, the animation of physical robots is brutally unforgiving with regards to modeling assumptions and sim-to-real gaps. Specifically, motions that are executed on a physical robot are required to be robust to changes in the environment (e.g. uneven ground, varying friction), noisy and imperfect state and ground contact estimation, and must also respect the physical limitations of the robot (e.g. actuator speed and torque curves).

Robustness to disturbances is traditionally achieved through closed-loop model predictive control [Neunert et al. 2016; Wieber 2006] or through high-level feedback such as gait timing and foothold adjustments [Griffin et al. 2017; Kryczka et al. 2015]. The vast majority of legged robotics research has focused on functional locomotion, where the robot is given high-level velocity commands and the gait style is hand-crafted for efficiency and robustness [Apgar et al. 2018; Nishiwaki et al. 2002; Xi and Remy 2014; Zhou et al. 2022]. In contrast, we seek to produce walking gaits that are stylized and not purely functional. There has been effort targeted at producing robotic gaits that mimic human walking from motion capture using model-based control [Ames 2014; Dariush et al. 2008]. While the resulting gaits are not purely functional, they do not provide the artistic control that we seek.

Recent work has used RL approaches to produce expressive and functional walking motions for legged robots. Using domain randomization, these systems can be made robust to modeling uncertainties while exhibiting a variety of gaits [Siekmann et al. 2021], and in some cases allowing for online adjustment of functional gait parameters such as step height and speed [Margolis and Agrawal 2022]. Others can extrapolate walking styles from motion capture or video performances [Bohez et al. 2022; Nakaoka et al. 2007; Peng et al. 2020; Sok et al. 2007; Yao et al. 2022b]. While these techniques yield impressive results, they require training time and, therefore, cannot provide the animator with immediate feedback. This makes them unsuited for an interactive workflow as we target here.

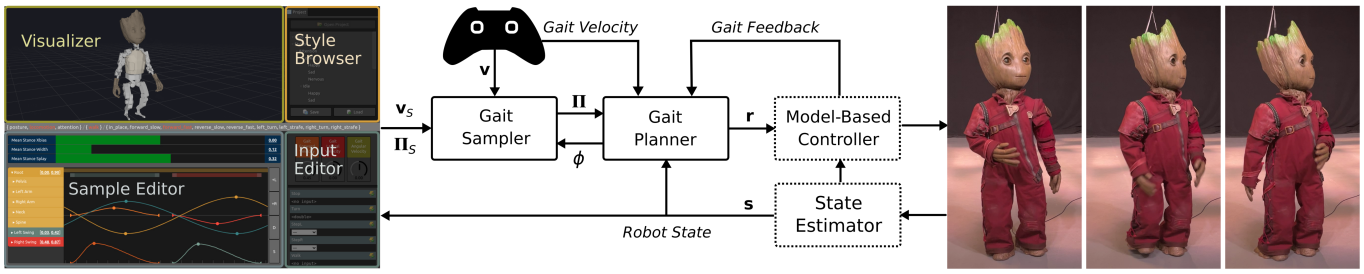

Fig. 2. **Walking gait design and execution.** An artist authors walking styles by defining sample parameters at key walking velocities using a graphical animation editor (left). The sample parameters for each style are translated into dynamically feasible whole-body motion using a procedural animation technique that runs on the physical robot (right). The robotic character walks in the desired style, tracking the commanded gait velocity from a joystick input.

Graphical Gait Editing. Yin et al. [2007] demonstrated stylized gaits in simulation using a method where control parameters are authored using a custom GUI or extracted via motion capture. Boston Dynamics have published online videos of highly-dynamic choreographed robotic locomotion. In a blogpost [Ackerman 2021] it was disclosed that they use a custom software, "Choreographer", which allows for dance sequences to be authored. However, their exact approach remains undisclosed and proprietary. Similar to the presented work, Nakaoka et al [2010] developed an authoring tool for bipedal motion design, taking as input a set of desired tasks and producing a dynamically-feasible reference trajectory for the physical robot. While this approach supported stylized stepping behaviors, each step was individually keyframed. Here, we focus on procedurally-generated, cyclic walking gaits that vary smoothly over a continuous range of velocities.

3 OVERVIEW

Fig. 2 includes a high-level block diagram of our proposed procedural animation technique. Our objective is to design a system that enables a bipedal robot to walk in a rich variety of artist-directed styles while tracking an arbitrary planar velocity command, $\mathbf{v}$ (see Sec. 4.1, Path Frame). Inspired by animation and game authoring tools, our framework includes an interactive gait editor that allows an artist to design target gait cycles for robotic characters at key walking velocities. The engine interpolates these gait "samples" using a phase-space blending strategy that produces smooth variation of the target style over a continuous range of input velocities. Each sample, S, is defined by a set of parameters, $\mathbf{\Pi}_S$, which prescribe the desired kinematic motion of the robot over a two-step cycle at a given velocity, $\mathbf{v}_S$. These include step timings, animation curves, and stance attributes. The target walk is executed by a real-time gait planner, that is parameterized on the blended sample parameters, $\mathbf{\Pi}$, and tracked using the model-based control stack described in Sec. 7.

A unique aspect of our gait planner, when compared to other robotic systems, is the high-level of expressiveness afforded by a rich set of gait parameters. As described in Sec. 4.3, the gait parameter vector, $\mathbf{\Pi}$, encodes the salient features of the desired walking motion, for example the timing and shape of each swing foot trajectory. The parameter space is designed such that it is impossible to violate certain kinematic constraints imposed by the planner. This is accomplished by enforcing suitable limits and boundary constraints on animation curves and through an appropriate choice of variables. In general, the parameter vector is expected to vary smoothly over a range of input velocities; however, the planner is designed to permit discrete parameter changes in the case of live editing.

The proposed gait editor renders a 3D view of the character tracking the authored walk style using a dynamic simulation of the robot hardware running the full model-based control stack. By leveraging real-time simulation and control during the design phase, our system provides instantaneous feedback to the artist regarding the expected hardware performance, allowing the author to test stylized walking gaits using a gamepad to command the robot's velocity. Additionally, this proposed editor can be used to update and refine gaits directly on the robot hardware, which runs an identical control stack.

The remainder of the paper is organized as follows: Sec. 4 provides an overview of the proposed Gait Planner, including a detailed description of the selected animation parameters and state machine used to produce kinematic reference trajectories. Sec. 5 describes the Gait Sampler, which selects and interpolates authored walk cycles as a function of velocity, and Sec. 6 discusses a custom, interactive gait editor used to author gaits via the proposed animation method. For tracking the resultant gaits on hardware, Sec. 7 describes the selected Model-Based Controller and corresponding State Estimator which estimates the robot's state from sensor measurements. Finally, Secs. 8 and 9 present simulation and hardware results on multiple robotic characters, along with our conclusions.

4 GAIT PLANNER

The proposed animation engine targets the general class of bipedal robots that can be modeled as articulated rigid body systems. Assuming N joint coordinates and 6 floating-base (or root) coordinates, the robot configuration is defined as $\mathbf{q} \in \mathbb{R}^{N+6}$ and the full state is given by $\mathbf{s} = [\mathbf{q}^\top \ \dot{\mathbf{q}}^\top]^\top$.

Referring to Fig. 2, we aim to design a procedural Gait Planner, g, that maps a time-varying, planar input velocity, $\mathbf{v}(t)$, to a vector of Cartesian and joint-space reference trajectories, $\mathbf{r}(t)$, that will enable omnidirectional walking on the target robot when resolved by a whole-body controller. The input/output behavior of such a planner can be formalized using the following mapping function,

$$\mathbf{r}(t) = g(\mathbf{v}, \mathbf{s}, \mathbf{\Pi}, t). \tag{1}$$

Here we assume the gait planner is provided the current desired velocity command and estimated robot state at each time step. The planner is further parameterized by a vector of animation parameters, $\mathbf{\Pi}$, that encode the desired posture, step timing, swing foot motion, heel-toe behavior, arm motion, and other salient features of an expressive walking gait.

In the presented system, the planner output vector includes the desired pelvis (root) height, pelvis rotation, left and right foot poses, and upper body joint coordinates (e.g. arm, neck, spine, tail angles), i.e.

$$\mathbf{r}(t) = \begin{bmatrix} z_{\text{pelvis}} & \boldsymbol{\theta}_{\text{pelvis}}^{\top} & \mathbf{p}_{\text{foot,l}}^{\top} & \mathbf{p}_{\text{foot,r}}^{\top} & \mathbf{q}_{\text{upper}}^{\top} \end{bmatrix}^{\top}. \quad (2)$$

Here we use the symbols, z and $\boldsymbol{\theta}$, to denote a z-axis translation and quaternion rotation, while, $\mathbf{p} = [\mathbf{x}^{\top} \ \boldsymbol{\theta}^{\top}]^{\top}$, denotes a 6-DoF pose vector with translation, $\mathbf{x} \in \mathbb{R}^3$. The left and right foot frames are defined at the geometric center of each sole, while the pelvis frame is assumed to be attached to the root link using an x-forward, z-up convention. All reference poses are defined with respect to an inertial origin frame which also follows a z-up convention.

While the planner computes the desired upper-body state, $\mathbf{q}_{\text{upper}}$, in joint-space, the lower-body reference motion is prescribed in Cartesian coordinates for compatibility with task-space control strategies commonly employed among walking robots. This task-space representation also draws parallels to character animation rigs, which often employ pelvis (root) and foot IK solvers to animate the legs during walking.

Note that the reference does not specify a desired horizontal pelvis translation, $[x_{\text{pelvis}} \ y_{\text{pelvis}}]^{\top}$. The pelvis translation is strongly correlated with the center of mass (CoM) dynamics of the robot, which are critical for balance. As described in Sec. 7, the proposed method relies on a model predictive controller to stabilize the CoM dynamics subject to the physical contact constraints. The optimized CoM translation is appended to the vector of planner outputs, $\mathbf{r}(t)$, which are tracked using an inverse dynamics-based whole-body controller. In general, we have found that automating the horizontal motion of the pelvis in this manner has little impact on the perceived style of a gait, and typically leads to more natural looking walking compared to manual curve editing.

The following subsections detail the implementation of the proposed planner. We begin with adopted conventions, followed by a description of the nominal motion planning algorithm enabling omnidirectional walking, and conclude with an overview of the animation parameters used to author stylized gaits.

4.1 Conventions

Walk Cycle. A bipedal walking gait is typically defined by alternating steps segmented by a double support phase, i.e. a time interval where both feet are in contact. This excludes flight phases as in jogging or running gaits. Fig. 3 illustrates the concept of a two-step walk cycle. The robot is assumed to progress through two complete steps, with periodic motion of the feet, pelvis, and upper body, e.g. arms, neck, and spine. In the proposed framework, the walk cycle begins with a left liftoff (LLO) event as the character transitions from double support to the left swing phase. This is followed by a left touchdown (LTD), right liftoff (RLO), and right touchdown (RTD) event, each corresponding with the start of a new contact phase. When walking at a constant velocity, the gait is assumed to be periodic, and the cycle ends at the following LLO event when the robot returns to the initial configuration having translated some distance determined by the cycle period, T_S, and gait velocity. In the proposed planner, the associated event times, t_{LLO}, t_{LTD}, t_{RLO}, and t_{RTD}, are derived from a subset of animation parameters which determine the start and end of each swing interval as described in Sec. 4.3, Time Intervals.

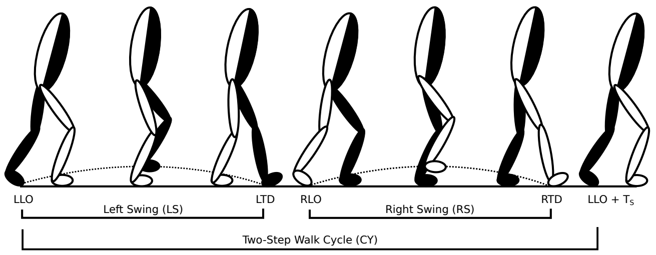

Fig. 3. **Two-step walk cycle**. The gait transitions through four contact events during each cycle: left liftoff (LLO), left touchdown (LTD), right liftoff (RLO), and right touchdown (RTD). For the purposes of this paper, the two-step cycle is defined such that it starts and ends at LLO.

Path Frame. While it is common to animate periodic walk cycles at constant speed, the goal of the proposed planner is to track an arbitrary, time-varying input velocity, enabling a robot to move freely about an environment. By convention, the commanded velocity vector, $\mathbf{v} = [\,\dot{x} \ \dot{y} \ \dot{\theta}\,]^{\top}$, is expressed in body coordinates, where $\dot{x}$, $\dot{y}$, and $\dot{\theta}$ represent the desired forward, lateral, and angular velocity in the horizontal plane, respectively. The integral of the input velocity represents the desired walking path of the biped. Accordingly, we define a *path frame*, whose coordinates are given by the integral of the input velocity. Fig. 4 includes an illustration of two walking paths, at both a constant and variable velocity. By our convention, the x-axis of the path frame is aligned with the front of the robot and the y-axis to the left. The path frame coordinates are then given by the planar pose, $\mathbf{p}^{\mathbf{v}} = [\,x \ y \ \theta\,]^{\top}$, where the non-bolded θ represents the robot's z-axis rotation, or yaw, relative to the origin frame.

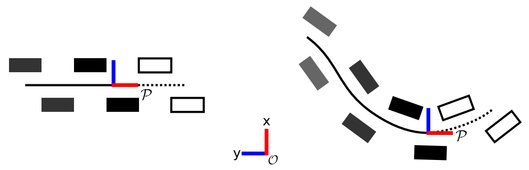

Fig. 4. **Example walking paths with sampled footholds.** The path frame, $\mathcal{P}$, with x-axis in red and y-axis in blue, is defined as the integral of the gait velocity. The path frame history (solid line) illustrates the past walking path relative to the inertial origin frame, $\mathcal{O}$, while the extrapolated trajectory (dotted line) defines the predicted walking path. The planner derives the upcoming footholds (white) by sampling the predicted path frame at the midpoint of each upcoming foot contact interval. The achieved footholds (black) are fixed in place following each step at the corresponding touchdown event.

4.2 Motion Planning

To plan the reference motion for stylized gaits, the proposed method partitions the computed trajectories into two components:

- a nominal component of motion derived from the desired input velocity, $\mathbf{v}$, satisfying the basic kinematic and timing constraints associated with the walking task
- a stylized component of motion derived from the vector of animation parameters, $\mathbf{\Pi}$, that encodes the gait expression

Applying this decomposition to the reference states, $\mathbf{r}(t)$, as defined in Eq. 2 we have

$$\begin{aligned} z_{\text{pelvis}} &= z^{\mathbf{v}}_{\text{pelvis}} + z^{\mathbf{\Pi}}_{\text{pelvis}} \\ \boldsymbol{\theta}_{\text{pelvis}} &= \boldsymbol{\theta}^{\mathbf{v}}_{\text{pelvis}} \boldsymbol{\theta}^{\mathbf{\Pi}}_{\text{pelvis}} \\ \mathbf{p}_{\text{foot},\mu} &= \mathbf{p}^{\mathbf{v}}_{\text{foot},\mu} \oplus \mathbf{p}^{\mathbf{\Pi}}_{\text{foot},\mu} \; \forall \mu \in \{l, r\} \\ \mathbf{q}_{\text{upper}} &= \mathbf{q}^{\mathbf{v}}_{\text{upper}} + \mathbf{q}^{\mathbf{\Pi}}_{\text{upper}}, \end{aligned} \tag{3}$$

where the superscripts, $\mathbf{v}$ and $\mathbf{\Pi}$, are used to denote the nominal and stylized components of motion, respectively. Note that each reference foot pose is given by the composition of the nominal and stylized foot pose. Here, the pose composition operator, $\oplus$, defines the rigid body transform, $\mathbf{p}_A \oplus \mathbf{p}_B = [\mathbf{x}_A + \boldsymbol{\theta}_A \mathbf{x}_B,\ \boldsymbol{\theta}_A \boldsymbol{\theta}_B]^T$. This is equivalent to expressing the stylized foot pose in a moving reference frame defined at the nominal foot coordinates.

All gait trajectories are evaluated in real-time, and the final reference state is updated at a rate of 125 Hz in our experiments. The remainder of this subsection describes the computation of the nominal reference states, while Sec. 4.3 discusses the choice of animation parameters and the resulting stylized reference.

Footstep Plan. The planner relies on a finite state machine to handle transitions between double support and swing phases. To compute the nominal left and right foot trajectories during each contact phase, we first evaluate a desired footstep plan from the predicted walking path. Given the current input velocity, the motion planner extrapolates the walking path from the current path pose along a simple arc assuming zero acceleration. As illustrated in Fig. 4, a pair of upcoming footsteps is generated for each successive two-step cycle, k, by sampling the predicted path pose at appropriate foothold times, $t^k_{\text{foothold,l}}$ and $t^k_{\text{foothold,r}}$, and applying a nominal stance transform for each foothold side such that

$$\begin{aligned} \mathbf{p}^k_{\text{foothold,l}} &= \mathbf{p}^{\mathbf{v}}\left(t^k_{\text{foothold,l}}\right) \oplus \mathbf{p}_{\text{stance,l}} \\ \mathbf{p}^k_{\text{foothold,r}} &= \mathbf{p}^{\mathbf{v}}\left(t^k_{\text{foothold,r}}\right) \oplus \mathbf{p}_{\text{stance,r}}. \end{aligned} \tag{4}$$

Here $\mathbf{p}_{\text{stance,l}}$ and $\mathbf{p}_{\text{stance,r}}$ are constant planar poses expressed in the walking path frame, representing the nominal translation and rotation of each foot during stance. The stance poses contribute to the overall style of the walk and are determined by a subset of the planner animation parameters as discussed in Sec. 4.3, Attributes.

A natural choice for t^k_{foothold} is the midpoint of the time interval when the foot will be in contact with the foothold, e.g. $t^k_{\text{foothold,l}} = (t^{k+1}_{LLO} - t^k_{LTD})/2$. During the swing phase, the upcoming foothold is re-evaluated at each time step to account for changes in the input velocity. At the time of touchdown, the target foothold is fixed in place for the new support phase. In the event that the upcoming foothold overlaps the current support foot, the target position is projected to a safe stepping region using a conservative heuristic. Given the initial pose of the target foothold as expressed in the support foot frame, we select one of three regions in front, behind, or to the side of the support foot and clip the target position to nearest point on the boundary.

Nominal Support/Swing Foot Trajectories. The nominal foot trajectories are computed to satisfy non-sliding contact constraints during walking. With each step, the swing foot breaks contact with the current foothold at liftoff and makes contact with the upcoming foothold at touchdown. During the support phase, the foot reference pose is latched to the foothold pose evaluated at the last touchdown, i.e. $\mathbf{p}^{\mathbf{v}}_{\text{foot,l}}(t) = \mathbf{p}^{k-1}_{\text{foothold,l}}$. With each step, we compute a nominal swing foot trajectory that interpolates from the latched foothold to the upcoming foothold over the duration of the swing interval. Splitting the reference trajectory into linear and angular components, we have, for the left foot example,

$$\mathbf{x}^{\mathbf{v}}_{\text{foot,l}}(t) = (1 - \alpha(t))\, \mathbf{x}^{k-1}_{\text{foothold,l}} + \alpha(t) \mathbf{x}^k_{\text{foothold,l}} \tag{5}$$

$$\boldsymbol{\theta}^{\mathbf{v}}_{\text{foot,l}}(t) = SLERP\left(\boldsymbol{\theta}^{k-1}_{\text{foothold,l}}, \boldsymbol{\theta}^k_{\text{foothold,l}}, \alpha(t)\right). \tag{6}$$

Here $\alpha(t)$ is a cubic interpolation parameter that increases from 0 to 1 over the duration of the step with zero initial and final velocity, i.e. $\dot{\alpha}(t_{LLO}) = \dot{\alpha}(t_{LTD}) = 0$. Note that, like the footholds, the nominal swing foot trajectory has zero height and lies in the ground plane. The vertical swing trajectory is determined entirely by the animation parameters as described in Sec. 4.3, Function Curves.

Nominal Pelvis and Upper-Body Trajectories. The nominal pelvis rotation, $\boldsymbol{\theta}^{\mathbf{v}}_{\text{pelvis}}$, is defined such the the pelvis z-axis remains vertical, while the horizontal axes are aligned with the current path frame rotation, producing smooth pelvis yaw during turning. The pelvis height and upper body joint trajectories are specified directly by the stylized reference; thus, we define $z^{\mathbf{v}}_{\text{pelvis}} = 0$ and $\mathbf{q}^{\mathbf{v}}_{\text{upper}} = \mathbf{0}$.

4.3 Animation Parameters

While a biped's walking gait may vary dramatically as a function of speed, emotion, or personal style, the fundamental mechanics and constraints remain the same. If we consider the subspace of all dynamically feasible walking gaits for a specific biped, the achievable styles are determined by the salient differences among those gaits. This motivates the need for design tools that abstract the common complexities of walking gaits and allow artists to focus on the critical features that distinguish one style from another. This is especially important when designing gaits for physical robots, as it is generally impossible to ensure dynamic feasibility of an animated motion without automating some portion of the design process or post-processing the motion, resulting in motions that can significantly deviate from the artistic intent.

We propose an expressive gait parameterization that aids the rapid design of feasible walk cycles for robotic characters. Our goal is to design a set of gait parameters, $\mathbf{\Pi}$, that encode the salient features of a stylized gait. Ideally, a suitable set of gait parameters should strike a balance between several competing design objectives:

- maximizing the expressive range of valid gaits
- minimizing the number of design parameters
- minimizing the expected design time

To be deployed to a physical robot, an animated walk cycle must also satisfy certain kinematic and dynamic constraints. Referring to Fig. 3, the contact events should follow the appropriate gait sequence and the soles of the feet should never penetrate the ground plane or slip during contact. As a necessary condition for dynamic feasibility, the CoP should also lie inside the active support polygon at all times.

The proposed parameterization includes three data types that can be edited during the design phase: time intervals, attributes, and function curves.

Time Intervals. Time intervals are defined by a min and max value relative to the start of the walk cycle (refer to Sec. 4.1, Walk Cycle). Tab. 1 lists the selected intervals which include the full two-step cycle and the left and right swing intervals. Note that each configurable limit maps to a contact event in the walk cycle. The min values of the CY and LS intervals are clamped to 0. We also constrain the limits to satisfy the event ordering illustrated in Fig. 3, i.e. $t_{LLO} < t_{LTD} < t_{RLO} < t_{RTD} < t_{LLO} + T_S$, and enforce a lower bound on the distance between each event time based on the minimum permissible durations of the single and double support phases. This effectively limits the domain of expressible gaits strictly to walking.

Table 1. Time Intervals for a Gait Sample

Name	Description	(Min, Max)
CY	The two-step cycle interval	$(t_{LLO}, t_{LLO} + T_S)$
LS	The left swing interval	(t_{LLO}, t_{LTD})
RS	The right swing interval	(t_{RLO}, t_{RTD})

Function Curves. Function curves are time-varying trajectories that parameterize the stylized reference states for the animated walk cycle, i.e. z^{Π}_{pelvis}, $\boldsymbol{\theta}^{\Pi}_{\text{pelvis}}$, $\mathbf{p}^{\Pi}_{\text{foot,l}}$, $\mathbf{p}^{\Pi}_{\text{foot,r}}$, and $\mathbf{q}^{\Pi}_{\text{upper}}$. Tab. 2 lists the core functions used to animate a stylized walk. Each curve is defined on a specific time interval, i.e. the periodic cycle interval, CY, or discrete swing intervals, LS or RS as illustrated in Fig. 6. The functions are parameterized as cubic splines to enable traditional curve editing, where an artist shapes the function using knot position and optional tangent controls.

Left and right foot translation and rotation curves are defined for each swing interval, allowing an artist to shape the swing trajectory during stepping. These parameters define the time-varying reference pose, $\mathbf{p}^{\Pi}_{\text{foot}}$, with respect to the nominal swing foot frame with coordinates, $\mathbf{p}^{\mathbf{v}}_{\text{foot}}$, for each step side. The desired translation is given by $\mathbf{x}^{\Pi}_{\text{foot}} = \mathbf{x}^{\Pi}_{\text{swing}} + \mathbf{x}^{\Pi}_{\text{heel-toe}}$. Here $\mathbf{x}^{\Pi}_{\text{swing}}$ defines the animated swing trajectory, parameterized by x, y, and z curves that span the time interval from liftoff to touchdown. The initial and final values of the swing interval function curves are clamped to zero to ensure that the swing foot trajectory starts and ends at the nominal liftoff and touchdown pose, assuming a flat foot orientation. The second term, $\mathbf{x}^{\Pi}_{\text{heel-toe}}$, is a translation offset derived from the foot pitch parameters described below and ensures the contact height constraints are satisfied during heel-toe walking.

The stylized foot rotation, $\boldsymbol{\theta}^{\Pi}_{\text{foot}}$, is parameterized by four animation angles, $(\theta, \psi_{\text{heel}}, \psi_{\text{toe}}, \gamma)$, which represent intrinsic rotations applied in the listed order about the z, y, and x axes. Here θ and γ determine the yaw and roll of the foot relative to the nominal swing motion, and are clamped to zero at the swing boundaries. The parameters, ψ_{heel} and ψ_{toe}, represent pitch rotations applied about the heel and toe, respectively. Inspired by the "reverse foot lock" rigs used in character animation, the heel and toe pitch transforms are applied sequentially, as illustrated in Fig. 5. By enforcing $-\frac{\pi}{2} \leq \psi_{\text{heel}} \leq 0$ and $0 \leq \psi_{\text{toe}} \leq \frac{\pi}{2}$, we can ensure that the lowest point on the foot always remains at or above the ground plane. The corresponding heel-toe translation offset, $\mathbf{x}^{\Pi}_{\text{heel-toe}}$, is derived from the sequential pitch rotations given the foot length. At liftoff and touchdown, one of the two pitch parameters is permitted to be non-zero, corresponding to toe or heel contact at the swing boundaries. The transition to and from the nominal flat foothold pose is automated during the support phase, as described in Sec. 4.3, Attributes.

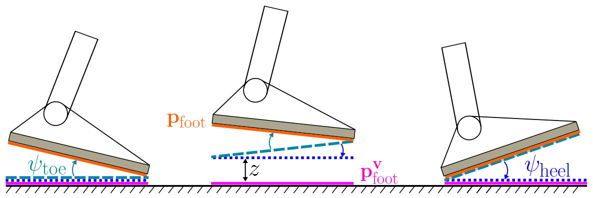
ψ_{toe}
$\mathbf{p}_{\text{foot}}$
z
$\mathbf{p}^{\mathbf{v}}_{\text{foot}}$
ψ_{heel}

Fig. 5. Heel-toe rotation parameters for stylized swing trajectories. Inspired by common character animation rigs, the foot is animated using two pitch rotations, ψ_{heel} and ψ_{toe}, applied first about the nominal heel position, then about the resulting toe position. One of the pitch angles must be constrained to zero at each contact switch event, allowing for either toe or heel contact.

.

The stylized pelvis height, z^{Π}_{pelvis}, is directly parameterized, while the path frame-relative rotation, $\boldsymbol{\theta}^{\Pi}_{\text{pelvis}}$, is prescribed by a set of Euler-ZYX angles that map to yaw, pitch, and roll. The pelvis curves are defined on the full two-step cycle interval along with a set of joint-space function curves that map directly to the upper body joint trajectories, $\mathbf{q}^{\Pi}_{\text{upper}}$. These include curves for each arm, neck, spine, and/or tail, enabling rich upper body motions. We enforce periodic boundary constraints on these functions so that the interpolated trajectories are C1 continuous as the walk cycle loops.

Table 2. Function Curves of a Gait Sample

Interval: Name	Description
LS: Foot x, y, z	Left foot swing translation
LS: Foot $\theta, \psi_{\text{heel}}, \psi_{\text{toe}}, \gamma$	Left foot swing rotation angles
RS: Foot x, y, z	Right foot swing translation
RS: Foot $\theta, \psi_{\text{heel}}, \psi_{\text{toe}}, \gamma$	Right foot swing rotation angles
CY: Pelvis z	Pelvis z translation
CY: Pelvis θ, ψ, γ	Pelvis rotation angles
CY: Upper body q_i	Upper body joint angles (e.g. arms)

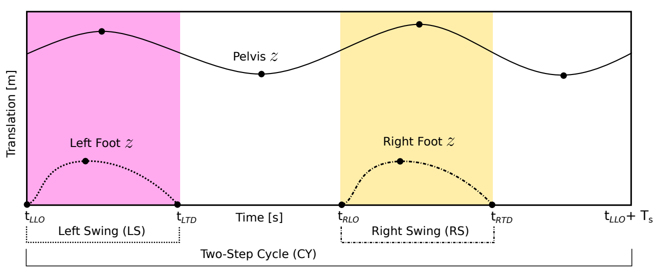

Fig. 6. **Function curves**. Example Pelvis z, Left Foot z, and Right Foot z function curves for a two-step cycle. Pelvis functions are defined on the full two-step cycle, while left and right foot functions are defined on the left and right swing intervals, respectively.

.

Attributes. Attributes are configurable constants that affect some aspect of the walk cycle. The stance splay, width, and x-bias attributes determine the mean angular, lateral, and forward offset between the left and right footholds relative to the walking path. Fig. 7 illustrates the effect of each attribute on the nominal stance pose, $\mathbf{p}_{\text{stance}}$, for each foot when the gait velocity is zero. Recall from Sec. 4.2, Footstep Plan, that the stance poses determine the placement of the planned footholds with respect to the path frame. The proposed parameterization enables a wide range of expression including narrow-stanced, wide-stanced, duck-footed, pigeon-toed, and asymmetric gaits while preventing the animator from having to manually define the foothold poses - an otherwise time-consuming process.

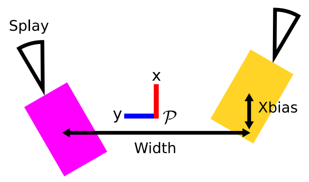

Fig. 7. **Stance attributes**. The left and right stance poses are shown for a stationary path frame, $\mathcal{P}$ (standing or stepping with zero gait velocity).

As mentioned in Sec 4.3, Function Curves, the proposed parameterization includes attributes to automate the foot pitch behavior during the support phases for heel/toe walking. The artist is provided two parameters: $\Delta t_{\psi_{LO}}$ and $\Delta t_{\psi_{TD}}$, which determine the duration of the heel/toe transitions to and from a flat foot configuration prior to liftoff and immediately following touchdown, respectively. These terms control how quickly the foot is lifted and planted during toe/heel-off and heel/toe-strike. Each is capped to a maximum of 50% of the total foothold duration. The planner linearly interpolates the heel or toe pitch angle to zero over the course of the transition.

5 GAIT SAMPLER

As a character's walking velocity varies over time, the walk cycle must adapt to accommodate different step timings and/or stride lengths. Additionally, distinct gait features may emerge with changes in speed, for example, a more pronounced arm swing or toe-off motion while walking at a fast pace. As discussed in Sec. 3, we employ a sample-based design methodology to animate stylized walking gaits that vary over a continuous range of walking velocities. Here we define a gait sample, S, as a set of gait parameters associated with a fixed walking velocity, i.e. $(\mathbf{\Pi}_S, \mathbf{v}_S)$, that defines a periodic walk cycle. In the proposed system, an animator designs a complete walking style by authoring sample walk cycles at set of key walking velocities, representing forward, reverse, strafing, and turning gaits, as described in Sec. 5.3.

Referring to Fig. 2, the Gait Sampler computes the set of gait parameters, $\mathbf{\Pi}$, that are input to the gait planner by interpolating the authored gait samples. In order to generalize the sampled walking style to arbitrary input velocities, it is necessary to define a method to perform velocity-based interpolation of multiple samples. This section describes a phase-based blending strategy that relies on a gait "phase map" to interpolate time-varying parameters that are constrained based on the contact events. By blending the sample parameters as opposed to the corresponding joint trajectories, we can ensure that the resulting whole-body motion satisfies the constraints imposed on the original samples.

5.1 Phase Map

Here we introduce the concept of a *phase map* for a two-step walk cycle. A phase map is an invertible function that maps a generic *gait phase*, ϕ, to a unique *gait time*, t, specific to each walk cycle. The mapping is defined such that the cycle event times, t_{LLO}, t_{LTD}, t_{RLO}, t_{RTD}, and $t_{LLO} + T_S$, correspond to phase values of 0, ϕ_{LTD}, ϕ_{RLO}, ϕ_{RTD}, and 1. The values for ϕ_{LTD}, ϕ_{RLO}, and ϕ_{RTD} can be chosen arbitrarily to satisfy $0 < \phi_{LTD} < \phi_{RLO} < \phi_{RTD} < 1$. We select the intermediate phase values, 0.1, 0.5, and 0.6, respectively, based on typical double and single support durations of a symmetric biped gait. A smooth phase map, $\Omega(\phi)$, is derived by fitting a monotonic cubic spline to the corresponding phase/time waypoints as illustrated in Fig. 8.

Now consider the problem of associating a specific gait time, t_A, from cycle A, with an "equivalent" gait time, t_B, from cycle B. Using the respective phase maps, $\Omega_A(\phi)$ and $\Omega_B(\phi)$, the solution is given by $t_B = \Omega_B(\Omega_A^{-1}(t_A))$. Note that this method guarantees that contact event times, e.g. t_{LTD}, will always map to the equivalent event time in another cycle. This property of the chosen phase map preserves contact event-based timing constraints when mapping gait signals from one cycle to another.

5.2 Blending Two Samples

Given two samples, A and B, an interpolated parameter set, Π_{AB}, can be computed by blending the individual attributes, time intervals, and function curves according to a desired blend ratio, $\alpha \in (0, 1)$. The result is a valid sample associated with an intermediate velocity. Attributes such as the stance width are interpolated using a simple alpha blend, i.e.

$$r = (1 - \alpha)r_A + \alpha r_B \tag{7}$$

where r_A and r_B are the associated sample values. Similarly, the interpolated time intervals are computed by alpha-blending the min and max values independently. Because the blend operation is a convex combination, the ordering constraints imposed on the sample event times are preserved by the blended outputs.

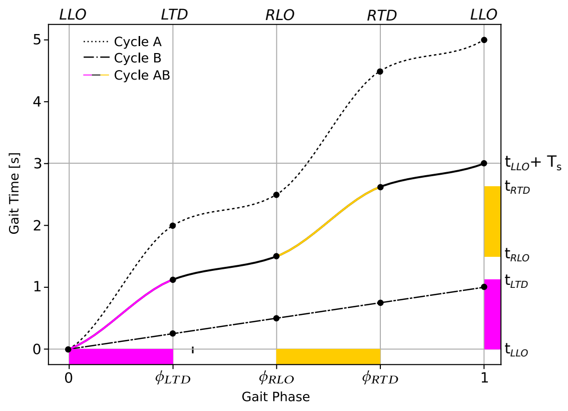

Fig. 8. **Phase maps**. Example phase maps for two gait cycles, A and B, along with the blended cycle, AB, for $\alpha = 0.5$.

In order to blend the function curve parameters, we first need to evaluate the phase map, $\Omega_{AB}(\phi)$, for the interpolated walk cycle based on the blended time intervals. Then, the blended function curve can be evaluated at an abitrary gait time using the following algorithm:

(1) Compute the equivalent gait phase, $\phi = \Omega_{AB}^{-1}(t)$.
(2) Compute the equivalent gait time for sample A, $t_A = \Omega_A(\phi)$.
(3) Compute the equivalent gait time for sample B, $t_B = \Omega_B(\phi)$.
(4) Evaluate function curve A at t_A.
(5) Evaluate function curve B at t_B.
(6) Interpolate the resulting values using Equation 7.

The proposed phase-space blending method shares commonalities with the time warp registration curves presented in [Kovar and Gleicher 2003], which are evaluated using dynamic time warping (DTW). Like DTW, we map time to a monotonically-increasing parameter, ϕ, allowing time-correlation and blending of motions with different timing characteristics. However, while DTW computes an automatic mapping based on a distance function that compares animation frames, the proposed method relies on a gait-specific correlation based on the contact constraint timings for each walk cycle. As opposed to blending animation frame data (joint/root states), the proposed gait sampler blends the set of procedural gait parameters, which are directly constrained by the contact timings. Overall, registration curve methods are a powerful tool for blending diverse animations, while the proposed strategy is specific to walk cycles and ensures that the contact-related kinematic constraints are satisfied for all blended motions.

5.3 Velocity-Based Sample Interpolation

Using the proposed editor described in Sec. 6, walking styles are authored by animating 9 independent gait samples corresponding to the key walking velocities listed in Tab. 3. Given an arbitrary input velocity, $\mathbf{v} = [\,\dot{x}\ \dot{y}\ \dot{\theta}\,]^T$, the sample parameters are interpolated using a three step process illustrated in Fig. 9. In the first step, we interpolate the forward, reverse, and in place samples by blending the two nearest neighbor samples based only on the value of $\dot{x}$. Let $\dot{x}_A$ and $\dot{x}_B$ represent the velocities of the two nearest forward/reverse samples, A and B. The blend ratio is computed based on the distance from each sample velocity, i.e.

$$\alpha_{\dot{x}} = f_{\text{soft}}\left(\frac{\dot{x}-\dot{x}_A}{\dot{x}_B-\dot{x}_A}\right) \tag{8}$$

where $f_{\text{soft}}(\alpha) = -2\alpha^3 + 3\alpha^2$ is a cubic polynomial that maps the unit domain to the unit range but with a first derivative of zero at both limits. This ensures that there are no discontinuities in the first derivatives of the blended sample parameters when the input velocity crosses into a new region of the sample space.

Table 3. Gait Sample Velocities

	$\dot{x}$	$\dot{y}$	$\dot{\theta}$
Forward Fast	$\dot{x}_{\max}$	0	0
Forward Slow	$\dot{x}_{\text{slow}}$	0	0
Step In Place	0	0	0
Reverse Slow	$-\dot{x}_{\text{slow}}$	0	0
Reverse Fast	$-\dot{x}_{\max}$	0	0
Left Strafe	0	$\dot{y}_{\max}$	0
Right Strafe	0	$-\dot{y}_{\max}$	0
Left Turn	0	0	$\dot{\theta}_{\max}$
Right Turn	0	0	$-\dot{\theta}_{\max}$

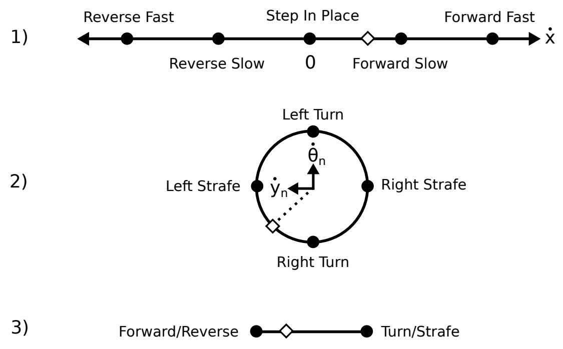

Fig. 9. **Sample interpolation**. Three step process for omnidirectional sample interpolation. In each step the white diamond represents the input velocity while each black circle represents a different sample velocity mapped to the relevant blend space. In 1) and 2) we blend the nearest forward/reverse and turn/strafe samples, respectively. In step 3) we blend the intermediate results to produce the final interpolated parameter set.

In the second step, we interpolate the turn and strafe samples based on the values of $\dot{\theta}$ and $\dot{y}$. First, we normalize the velocities based on the maximum values defined in Tab. 3, i.e. $\dot{\theta}_n = \dot{\theta}/\dot{\theta}_{\max}$ and $\dot{y}_n = \dot{y}/\dot{y}_{\max}$. Then, we compute the equivalent polar coordinates, $\beta = \text{atan2}(\dot{y}_n, \dot{\theta}_n)$ and $\rho_n = \min\left(\sqrt{\dot{y}_n^2 + \dot{\theta}_n^2}, 1\right)$, clipped to the unit circle. Note that the four turn and strafe samples lie on the unit circle in this normalized space and map to $\beta = 0$, $\pi/2$, π, and $3\pi/2$. Similar to the forward/reverse case, we can compute the two nearest neighbors among the turn/strafe samples and blend them using the ratio:

$$\alpha_{\beta} = f_{\text{soft}}\left(\frac{\beta-\beta_A}{\beta_B-\beta_A}\right) \tag{9}$$

where β_A and β_B are the corresponding polar angles of the samples.

Steps 1 and 2 produce a blended forward/reverse sample and a blended turn/strafe sample, respectively. In the third step, we blend these two intermediate samples based on the relative norms of $\dot{x}_n = \dot{x}/\dot{x}_{\max}$ and ρ_n. The final blend ratio is given by

$$\alpha_{\text{turn-strafe}} = f_{\text{soft}}\Big(\rho_n \cdot \big(1 - \eta f_{\text{soft}}\big(|\dot{x}_n|\big)\big)\Big) \tag{10}$$

where $\eta \in (0, 1)$ is an adjustable gain nominally set to 0.5. Note that when $\dot{x}$ is zero, the first term in the product fully determines the ratio at which the turn-strafe sample is blended into the final result. As $\dot{x}$ approaches the maximum forward/reverse velocity, however, the second term in the product effectively scales the ratio by $(1 - \eta)$ to reduce the influence of the turn-strafe samples.

6 INTERACTIVE GAIT EDITOR

In the proposed system, gait samples are authored using a custom graphical user interface that allows artists to design stylized walks for the proposed animation technique. A screenshot of the user interface is show in Fig. 2. The application includes a robot visualizer, style browser, input editor, and sample editor. The visualizer displays a 3D model of the target robot for visual feedback, and the style browser allows the user to load and save various walking styles. The input editor provides an interface to adjust the desired gait velocity, while the sample editor includes an animation curve editor and additional controls to modify the gait parameters including attributes, time intervals, and function curves.

The UI is designed for ease-of-use by animators that are familiar with popular character animation tools. The gait parameters emulate common joint and IK controls used in walking character rigs and were selected in consultation with a professional animator. When designing a gait, an animator can toggle the application interface between one of two modes described below.

6.1 Edit Mode

In edit mode, the user can select a desired sample for the current walking style and edit its parameters. The 3D window displays a live view of the character walking at the corresponding sample velocity with the periodic two-step gait cycle computed by our procedural animation method. The animation is updated in real-time to reflect changes in the sample parameters. This allows the animator to quickly visualize the effect of varying the value of an attribute, time interval, or a knot on a function curve. The visualizer also provides feedback when the character's joints approach their limits by shading the child links red. This allows the artist to catch potential issues early in the design phase and adjust the sample parameters to achieve the desired safety margins.

A screenshot of the sample editor is shown in Fig. 10. The function curves defined in Sec. 4.3 are represented as cubic Hermite splines and modified using an interface inspired by traditional animation curve editors. Boundary constraints are enforced at the limits of the gait cycle and swing intervals to prevent infeasible parameter sets. For example, when the swing intervals are resized, the knot times of the associated function curves are also re-scaled to ensure the swing trajectory remains valid.

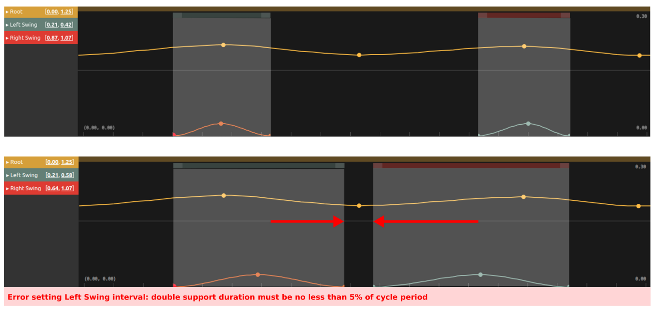
▸ Root [0.00, 1.25]
▸ Left Swing [0.21, 0.42]
▸ Right Swing [0.87, 1.07]
0.30
(0.00, 0.00)
0.00
▸ Root [0.00, 1.25]
▸ Left Swing [0.21, 0.58]
▸ Right Swing [0.64, 1.07]
0.30
(0.00, 0.00)
0.00
Error setting Left Swing interval: double support duration must be no less than 5% of cycle period

Fig. 10. **Time interval and function curve editing**. Before (top) and after (bottom) editing the left and right swing intervals. Here the user is alerted to an error when the animation engine detects that double support duration is below a minimum threshold after adjusting the intervals.

.

6.2 Test Mode

In test mode, the user can interactively control the character's walking velocity using joystick controls. This allows the animator to inspect the authored style as the gait cycle varies over a range of continuous input velocities. The animator can also quickly evaluate and compare walking styles and/or test the controller performance while expressing a given style to determine if further refinement of the sample parameters is required.

7 MODEL-BASED CONTROLLER AND ESTIMATOR

To realize the stylized walking gaits on hardware, we employ a Model-Based Controller (compare with Fig. 2) to track the time-varying reference motion, $\mathbf{r}$, computed by the Gait Planner. The purpose of this feedback controller is to account for disturbances and modeling imperfections in the real world. When deploying the proposed system to a new robot, the model-based controller is tuned to accommodate a variety of motions. The control parameters are then duplicated in the animation authoring/simulation tool, which runs an identical control stack, and remain fixed during the design and deployment phase.

Note that the presented control stack is abstracted from the gait planner and sampler introduced in the above sections. In practice, the proposed control strategy could be replaced by alternative model- or learning-based controllers designed to track the planned task-space reference trajectories for bipedal walking.

7.1 Model-Based Controller

The tracking controller utilized for the experiments in Sec. 8 adopts well-known methods from the humanoid robotics field. A model predictive controller (MPC) similar to [Wieber 2006] plans the Center of Mass (CoM) reference of the robot over a short preview window including the expected footholds. The CoM and gait reference states are tracked using an optimization-based whole-body controller similar to [Koolen et al. 2016]. In the following, we provide additional detail regarding the control stack.

Center of Mass Planner. As discussed in Sec. 4, the gait planner does not specify a horizontal pelvis translation; instead, we rely on the MPC formulation detailed in the Appendix (MPC) to plan a dynamically-feasible, horizontal CoM trajectory given the anticipated footholds and step timings computed by the gait planner. The MPC is formulated as a sparse quadratic program (QP) that jointly optimizes the horizontal CoM and Center of Pressure (CoP) trajectories. Fig. 11 includes an example footstep plan with optimized CoM and CoP trajectories. The MPC is evaluated on each gait planner update (at a rate of 125Hz), and uses a preview window of $T = 2s$ (or approx. 4 footholds at a gait cycle time of 1s).

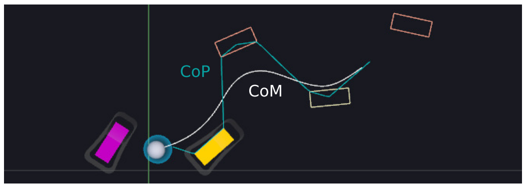

Fig. 11. **Optimized CoM and CoP.** The desired reference trajectories are computed by the model predictive controller given desired footstep sequence and gait timings.

Whole-Body Controller. The kinematic and dynamic references output by the gait and CoM planners are tracked by an optimization-based whole-body controller that closes the state feedback loop at a rate of 500 Hz. We rely on joint-space and Cartesian feedback control to compute desired acceleration and momentum rate of change objectives that are resolved by a task-space inverse dynamics solver described in the Appendix (ID). The solver, formulated as a whole-body QP, resolves the set of task-space objectives by optimizing the commanded joint accelerations and ground reaction forces subject to kinematic and dynamic constraints including Coulomb friction and acceleration limits.

From the inverse dynamics solution, we derive the vector of desired joint torque setpoints, $\boldsymbol{\tau} \in \mathbb{R}^N$. We also compute joint velocity and position setpoints by integrating the optimized accelerations and clipping the integral based on a maximum error from the measured quantities. The same joint-level setpoints are computed for the simulated and physical robot. On the physical character, a hardware abstraction layer transforms the joint setpoints into actuator commands, while in simulation, the joint positions, velocities, and torques are tracked directly using PD control with feedforward torques.

Gait Feedback. If the controller is unable to sufficiently track the CoM reference, it may be necessary to modify the robot's gait to maintain stability. CoM tracking errors may occur in response to unexpected disturbances, large accelerations in the commanded velocity, or periodically during the course of walking for certain gait expressions. Let $\hat{\mathbf{x}}_{\text{com}}$ and $\mathbf{r}_{\text{com}} \in \mathbb{R}^2$ represent the estimated and reference CoM positions projected on the ground plane. We employ three heuristic feedback mechanisms to regulate the CoM error, $\hat{\mathbf{x}}_{\text{com}} - \mathbf{r}_{\text{com}}$, in the form of velocity adjustment, step adjustment, and time scaling.

Velocity and Step Adjustment. The target walking velocity is given by the sum of the velocity command and a linear velocity offset, i.e. $\mathbf{v}^* = \mathbf{v} + [\Delta_{\text{vel}}^\top \; 0]^\top$. The offset, $\Delta_{\text{vel}} \in \mathbb{R}^2$, is computed as a linear combination of the CoM position and velocity error vectors passed through a deadband function and low-pass filter with a 0.15 Hz cutoff. Intuitively, this strategy accelerates the robot in the direction of a persistent error, buying additional time to recover at the expense of drift. Likewise, the upcoming foothold position is computed as the sum of the nominal foothold position and a horizontal step adjustment, i.e. $\mathbf{x}_{\text{foothold,l}}^{k^*} = \mathbf{x}_{\text{foothold,l}}^{k} + [\Delta_{\text{step}}^\top \; 0]^\top$, in the case of a left step. By adjusting the foothold location in the direction of the CoM error, this strategy aims to place the future CoP in a more favorable contact region to reduce the error in the next support phase.

Time Scaling. Because the centroidal dynamics resemble an unstable inverted pendulum during stepping, it is common for the CoM to lead or lag the planned reference trajectory. As such, we find that speeding up or slowing down the CoM reference trajectory is an effective strategy to reduce the centroidal error for dynamic gaits. Intuitively, this approach, which we refer to as *time scaling*, allows the estimated CoM to "catch up" to the reference, or vice versa. The proposed controller computes a time scaling rate, $\kappa_{\text{time}} \in (0, \infty)$, such that $\kappa_{\text{time}} > 1$ speeds up the reference, while $\kappa_{\text{time}} < 1$ slows it. We consider only the horizontal component of the CoM position and velocity error projected onto the axis pointing from the current to the next foothold, ϵ and $\dot{\epsilon}$. The time scaling rate is computed as $\kappa_{\text{time}} = e^{(k_p \epsilon + k_d \dot{\epsilon})}$, where k_p and k_d are experimentally-tuned, positive feedback gains. To ensure that the kinematic and dynamic motion tasks remain consistent for the given gait, we time-scale all reference trajectories in response to the CoM error, effectively slowing or speeding up the gait to aid stability, while maintaining the intended kinematic motion.

7.2 Estimator

As defined in Sec. 4, the robot state vector, $\mathbf{s}$, is estimated by a whole-body state estimator that fuses the robot's sensor measurements at 1kHz (compare with Fig. 2, State Estimator). The sensor measurements include the actuator encoder positions and inertial data from an IMU located on the pelvis (root) link. An Extended Kalman Filter based on [Solà 2015] is used to estimate the robot's root orientation. Relying on leg kinematics, the pelvis translation is estimated using zero-velocity observations when the feet are in contact with the ground plane. From the estimated state vector and rigid body model, we compute all Cartesian and centroidal estimates required for control.

8 RESULTS

We demonstrate the utility of our modeling on three robots: a small-scale dinosaur-inspired biped, an adult-sized humanoid, and a child-sized robot themed as Groot [Panzarino 2021]. Referring to Tab. 4, these robots were selected to demonstrate the generality of the proposed approach for animating gaits for bipeds with diverse proportions, mass distributions, and kinematics. While we provide in-simulation results for the adult-sized and dinosaur-inspired bipeds, we also demonstrate the viability of the proposed approach on hardware using a physical copy of the Groot robot along with a lower-body-only version of the same hardware platform.

Table 4. Key statistics for the different robots.

	#Act	height (mm)	weight (kg)
Groot	25	1155	18.5
Groot Lower Body	10	650	19.3
Humanoid	25	1640	74.5
Dino	20	615	16.9

#Act: number of actuators.

It should be noted that the gaits for each robot presented in this section were achieved by animating the gait parameters from Sec. 4 only. The underlying control stack presented in Sec. 7 is fixed and regarded a black box during gait design. All of the simulations in this section were implemented using MuJoCo [Todorov et al. 2012]. Each robot was modeled using physically plausible actuator sizing, and the simulation and hardware experiments were collected in real-time using an Intel i7 computer. Demonstrations of the authored gaits and experiments are included in the accompanying video.

8.1 Small-Scale Dinosaur Simulation

We used the proposed approach to design two distinct dynamic gaits for the small-scale dinosaur model shown in Fig. 1: a stylized "dino gait" and an asymmetric "injured gait". This character features 6-DoF legs with reverse knee joints, a 4-DoF neck with bird-like kinematics, and a 4-DoF segmented tail with alternating yaw and pitch joints.

Dino Gait. As illustrated in Fig. 12 (top), the cycle duration of the dino gait varies significantly across the *In Place*, *Forward Slow*, and *Forward Fast* samples, with the stepping frequency more than doubling as the character accelerates from 0 to 0.4 m/s. In addition to a more natural appearance, this timing variation serves to reduce the required stride length at high speeds, while maintaining a casual stepping frequency at slow speeds. Note that all of the samples also feature symmetric swing intervals. As the neck and tail are prominent features of this character, most of the authoring time was spent animating the respective periodic joint trajectories for each sample. The neck was animated to imitate a typical bird walk, while the tail animation was inspired by video of a simulated theropod dinosaur gait [Bishop et al. 2021].

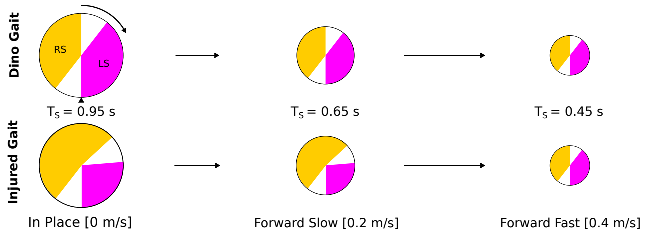

Fig. 12. **Dinosaur gait timing**. The colored regions indicate the time spent in the left and right swing phase. The radius of each circle represents the cycle duration. Top: the Dino Gait features symmetric swing intervals with a decreasing cycle time as the gait velocity increases. Bottom: the Injured Gait is asymmetric at slow speeds and symmetric at the *Forward Fast* speed.

Injured Gait. Referring again to Fig. 12 (bottom), the injured gait features asymmetric swing intervals for the *In Place* and *Forward Slow* samples. When coupled with a positive stance x-bias, the shorter left swing phase is intended to read as an injury to the right leg. For the *Forward Fast* sample, we use identical parameters to the dino gait, allowing the character to blend back to the symmetric gait at higher speeds.

Push Recovery Experiment. Referring to the accompanying video, we demonstrate the stability of model-based planning and control stack by applying external forces to the robot's pelvis link during walking. Fig. 13 illustrates the character's recovery following a 25 N, 1 s forward push as the robot walks at a slow forward velocity. We can make several conclusions observing the timing and magnitude of the three gait adjustment strategies discussed in Sec. 7.1, Gait Feedback.

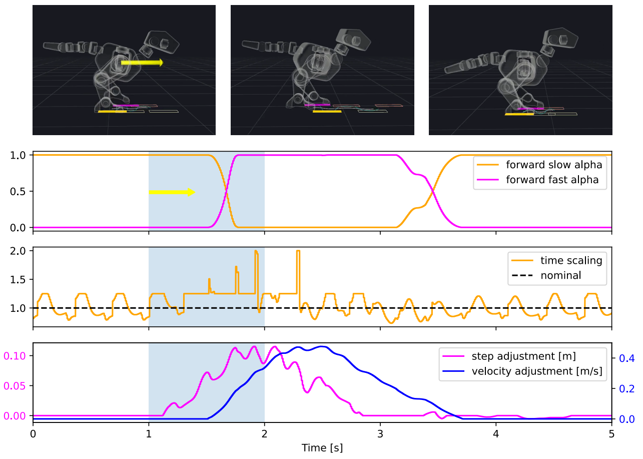

Fig. 13. **Dino push experiment**. The shaded area indicates the push interval. The top plot shows the sample activation as the robot adjusts it's velocity. The bottom plots show the response of the gait adjustment mechanisms used for recovery.

First, we note that time scaling is active during the nominal gait cycle, even before the push is applied. This is not unexpected for a dynamic gait, especially given the narrow foot width and fast CoM dynamics of the character. While the original step timings are modified, the total cycle timing is generally preserved, and the designer immediately sees this effect during authoring.

Shortly after the push begins, the time scaling rate saturates to the maximum single and double support limits, attempting to increase the step frequency. Simultaneously, the step adjustment begins to increase, peaking at around 0.1 m. Near the end of the push, the velocity adjustment begins to increase, peaking at approx. 0.4 m/s. Intuitively, the time scaling mechanism responds quickly to small errors, the step adjustment to larger errors that can be corrected by foot placement, and the velocity adjustment to persistent errors when the previous strategies fail.

Finally, note that as the velocity adjustment rises and falls, the gait sampler activates the *Forward Fast* sample. In addition to ensuring the behavior remains consistent with the authored style, the sample blending aids the recovery in this scenario by increasing the step frequency.

8.2 Adult-Sized Humanoid Simulation

To investigate our technique's performance on a human-inspired character, we designed several walking gaits for the simulated adult-sized humanoid robot shown in Fig. 1, based on a design from [Rodriguez et al. 2019]. This character features 6 DoF legs, a 2 DoF spine, 4 DoF arms, and a 3 DoF neck. The robot's height entails significantly slower CoM dynamics than the dinosaur model. In the accompanying video, we show the simulated robot walking in a default style that demonstrates the nominal reference computed by the gait planner, followed by a stylized walk that emulates a relaxed human gait. The slower cycle duration ($T_S = 1.5s$ *In Place*) and open stance help establish the casual style, and the pelvis and spine yaw animation emulate natural counter rotation of the hips and shoulders during the swing phase. We also demonstrate the robot executing three additional gaits that feature diverse upper-body motion and heel/toe contact, including a sneak walk, a double-bounce walk, and a neutral walk with heel-strike and toe-off.

8.3 Groot Hardware Platform

To investigate how authored styles transfer to physical robots, we designed and evaluated several stylized gaits for the Groot-themed hardware platform shown in Fig. 1. This child-sized humanoid robot features 5 DoF legs with line-contact feet, 6 DoF arms with a shoulder shrug function, and a 3 DoF neck. The robot's joints are controlled using electric actuators that track position, velocity, and feed-forward torque setpoints.

To compare the simulation and hardware results, we conducted a forward walking experiment, deploying the same authored style to three configurations of the character: the simulated robot, the physical robot, and the physical robot in full costume. Fig. 14 includes the statistical tracking error of several gait functions for each configuration. In our experience, the sim-to-real gap is fairly small. In general, the mean tracking error is similar between the simulated and physical robots; however, the standard deviation is smaller in simulation, indicating a more repeatable performance. Despite the higher variation in tracking error, the style transfers well to the hardware platform and the controller reliably tracks the desired gait across the desired range of input velocities. The walking gait is also robust to the unmodeled costume and handles disturbances from layered upper-body motions, such as waving while walking.

The accompanying video demonstrates additional walking styles, including a slouched walk, a playful walk with extended arms, and a double-bounce walk with significant arm swing. These styles highlight some of the variations that are achievable with the proposed gait parameterization, including diverse step timings, posture, and upper body motion.

We also demonstrate an alternative dynamic gait on a lower-body-only robot of the same design. The line-contact foot and lack of ankle roll present certain animation challenges, including the inability to track long swing intervals or animate the full 6-DoF pose of the foot during swing (we choose here not to track foot roll). Still, the proposed approach enables relatively easy authoring of dynamic gaits on the constrained hardware.

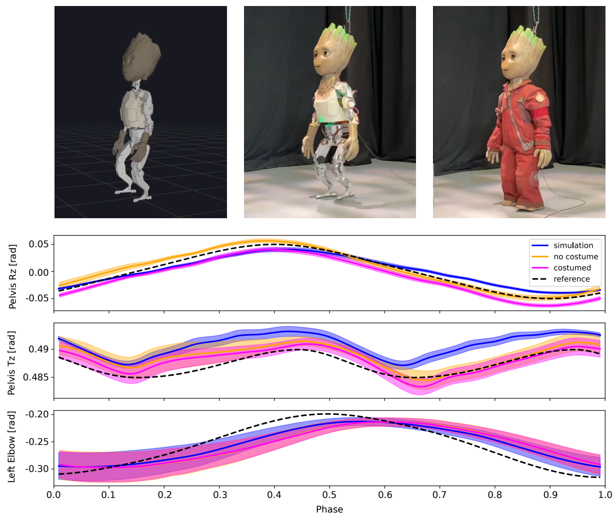

Fig. 14. **Animation tracking for the Groot character.** *Early Concept Exploration, Only.* We compare the pelvis yaw, pelvis height, and elbow bend tracking for three configurations: the simulated robot, the physical robot, and the physical character with full costume. The highlighted regions show the standard deviation of the errors over several walk cycles as a function of gait phase.

8.4 Gait Design

The amount of time required to animate a gait using the proposed system varies depending on the complexity of the style and target character. With only 10 actuated degrees of freedom, we were able to design an expressive bouncy gate for the lower-body Groot robot in a matter of minutes, editing the parameters directly on the hardware platform. For the Groot robot, some styles were authored directly on hardware while others were authored in simulation before being deployed to hardware. The adult-sized humanoid gaits were adapted from existing animation curves that were first authored using a traditional animation tool. Finally, the dinosaur gaits, which required characteristic tail and neck motions, were authored from scratch in approximately 1-2 hours.

8.5 User Study

To evaluate the ease-of-use of the proposed editor, we conducted a small user study with 4 participants unfamiliar with the tool. Two participants had previous animation experience in Maya, while two had no animation experience but had worked with humanoid robots. After a brief introduction to the tool, each participant was asked to design a gait sample for the adult-sized humanoid based on their initial concept. The participants took between 25 and 45 minutes to edit their respective gait, after which they were asked to answer the following questions: (1) *How does the resulting gait compare to* *your initial concept?* (2) *How did the interactive editing experience influence your design?* (3) *How does the tool compare to a traditional animation tool like Maya?* (4) *What is difficult and what is easy about the process/tool?*

Overall, responses were positive. For (1) users reported they were generally satisfied with their ability to design a gait based on their initial concept but made concessions as they discovered limitations related to step timing and range of motion. For (2) users felt the interactive nature of the editor helped them iterate quickly and encouraged experimentation. One user unfamiliar with traditional animation stated the process helped them visualize how each parameter affected the personality of the gait. For (3) one user with Maya experience reported missing the ability to define keyframes on the timeline. Another user reported that, while missing some features, the proposed tool was easier to get started on and led to faster results. For (4) all users had suggestions for UI features (e.g. color coded splines; slow-motion playback) that could improve the user experience. Several users reported that the animation curve editor was easy to use.

9 CONCLUSION

We have devised a system to author stylized walking gaits for bipedal robotic characters. As we demonstrated with three examples, our modeling applies to bipeds of varying size, proportion, actuated degrees of freedom, and mass distribution. Animators who have little or no prior robotics experience can design stylized walking gaits quickly and intuitively, with a user interface that exposes a rich set of animation parameters while hiding the complexity of the physical system. This is made possible by an interactive editing workflow that can be employed directly on hardware and is enabled by the proposed gait parameterization, phase-space sample blending, and gait adjustment strategies that preserve the authored style. The latter ensures that characters that perform a stylized gait respond naturally to disturbances as we demonstrate with our Dino character.

Limitations and Future Work. Our authoring approach has several limitations. While our tool can be used to author content directly on hardware, with several safeguards in place to prevent the robot from falling, we cannot guarantee that the animator's edits won't exceed the capabilities of the controller. Online authoring tools that can provide safety guarantees are an existing avenue for future work.

While our interactive editor provides a self-contained user interface for designing walking gaits, we also acknowledge that manual curve editing is not always the appropriate method of gait design across all use cases. In the future, we plan to investigate automated porting of walk cycles from keyframed animation content and motion capture data.

We have restricted our current authoring to walk cycles. An interesting future direction is an extension to more challenging gaits such as running and skipping and/or cyclic acrobatic motions that are well-suited for modeling in a procedural authoring tool.

Finally, model-based control stacks are difficult to build and require tuning. It is challenging to identify sim-to-real gaps, and even more challenging to address them. Learning-based control, or more specifically, imitation learning, is a promising alternative, with robustness to disturbances and tolerance to modeling imperfections acquired during training. We plan to explore the use of the proposed technique to build dense, artifact-free gait libraries for training RL policies. We are also interested in incorporating learning-based reference tracking controllers to supplement the proposed gait planner and editor, maintaining an interactive animation workflow.

ACKNOWLEDGMENTS

We would like to thank the Groot project team and participants of the user study.

REFERENCES

Evan Ackerman. 2021. How Boston Dynamics Taught Its Robots to Dance. *IEEE Spectrum* (Jan. 7 2021). https://spectrum.ieee.org/how-boston-dynamics-taught-its-robots-to-dance

Aaron D Ames. 2014. Human-inspired control of bipedal walking robots. *IEEE Trans. Automat. Control* 59, 5 (2014), 1115–1130.

Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan Hurst. 2018. Fast Online Trajectory Optimization for the Bipedal Robot Cassie. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation.

Peter J. Bishop, Antoine Falisse, Friedl De Groote, and John R. Hutchinson. 2021. Predictive simulations of running gait reveal a critical dynamic role for the tail in bipedal dinosaur locomotion. *Science Advances* 7, 39 (2021), eabi7348. https://doi.org/10.1126/sciadv.abi7348 arXiv:https://www.science.org/doi/pdf/10.1126/sciadv.abi7348

Steven Bohez, Saran Tunyasuvunakool, Philemon Brakel, Fereshteh Sadeghi, Leonard Hasenclever, Yuval Tassa, Emilio Parisotto, Jan Humplik, Tuomas Haarnoja, Roland Hafner, et al. 2022. Imitate and Repurpose: Learning Reusable Robot Movement Skills From Human and Animal Behaviors. *arXiv preprint arXiv:2203.17138* (2022).

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 130, 9 pages. https://doi.org/10.1145/1833349.1781156

Behzad Dariush, Michael Gienger, Arjun Arumbakkam, Christian Goerick, Youding Zhu, and Kikuo Fujimura. 2008. Online and markerless motion retargeting with kinematic constraints. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Robert J Griffin, Georg Wiedebach, Sylvain Bertrand, Alexander Leonessa, and Jerry Pratt. 2017. Walking stabilization using step timing and location adjustment on the humanoid robot, atlas. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 667–673.

Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. 2004. Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*. 522–531.

A. Hertzmann and V. Zordan. 2011. Physics-Based Characters. *IEEE Computer Graphics and Applications* 31, 04 (jul 2011), 20–21. https://doi.org/10.1109/MCG.2011.61

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 71–78. https://doi.org/10.1145/218380.218414

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Trans. Graph.* 36, 4, Article 42 (jul 2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Ahmad Abdul Karim, Thibaut Gaudin, Alexandre Meyer, Axel Buendia, and Saida Bouakaz. 2013. Procedural locomotion of multilegged characters in dynamic environments. *Computer Animation and Virtual Worlds* 24, 1 (2013), 3–15.

Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas De Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. 2016. Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics* 13, 01 (2016), 1650007.

Lucas Kovar and Michael Gleicher. 2003. Flexible automatic motion blending with registration curves. *ACM Symposium on Computer Animation*, 214–224.

Przemyslaw Kryczka, Petar Kormushev, Nikos G. Tsagarakis, and Darwin G. Caldwell. 2015. Online regeneration of bipedal walking gait pattern optimizing footstep placement and timing. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3352–3357. https://doi.org/10.1109/IROS.2015.7353844

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 (2021).

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-Driven Biped Control. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 129, 8 pages. https://doi.org/10.1145/1833349.1781155

Gabriel B Margolis and Pulkit Agrawal. 2022. Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior. *Conference on Robot Learning* (2022).

Franck Multon, Laure France, Marie-Paule Cani-Gascuel, and Giles Debunne. 1999. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation* 10, 1 (1999), 39–54.

Shin'ichiro Nakaoka, Shuuji Kajita, and Kazuhito Yokoi. 2010. Intuitive and flexible user interface for creating whole body motions of biped humanoid robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1675–1682. https://doi.org/10.1109/IROS.2010.5649038

Shin'ichiro Nakaoka, Atsushi Nakazawa, Fumio Kanehiro, Kenji Kaneko, Mitsuharu Morisawa, Hirohisa Hirukawa, and Katsushi Ikeuchi. 2007. Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances. *The International Journal of Robotics Research* 26, 8 (2007).

Michael Neunert, Cédric de Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. 2016. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

Koichi Nishiwaki, Satoshi Kagami, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. 2002. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *IEEE/RSJ international conference on intelligent robots and systems*, Vol. 3. IEEE, 2684–2689.

David E. Orin and Ambarish Goswami. 2008. Centroidal Momentum Matrix of a humanoid robot: Structure and properties. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 653–659. https://doi.org/10.1109/IROS.2008.4650772

Matthew Panzarino. 2021. Disney Imagineering's Project Kiwi is a free-walking robot that will make you believe in Groot. https://tcrn.ch/3sHRaZ3 Accessed on May 16, 2023.

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, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. 2020. Learning agile robotic locomotion skills by imitating animals. *Robotics: Science and Systems* (2020).

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 Trans. on Graph. (TOG)* 41, 4 (2022).

Nancy S. Pollard and Paul S. A. Reitsma. 2001. Animation of Humanlike Characters: Dynamic Motion Filtering with a Physically Plausible Contact Model.

Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. 2006. Capture Point: A Step toward Humanoid Push Recovery. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*. 200–207. https://doi.org/10.1109/ICHR.2006.321385

Alba M Rios Rodriguez, Steven Poulakos, Maurizio Nitti, Mattia Ryffel, and Robert Sumner. 2019. Parameterized animated activities. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. 1–9.

Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan Hurst. 2021. Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (Xi'an, China). IEEE Press, 7309–7315. https://doi.org/10.1109/ICRA48506.2021.9561814

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph.* 26, 3 (2007).

Joan Solà. 2015. Quaternion kinematics for the error-state Kalman filter. *ArXiv* abs/1711.02508 (2015).

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

Pierre-Brice Wieber. 2006. Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 137–142.

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

Weitao Xi and C. David Remy. 2014. Optimal gaits and motions for legged robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

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

Qingfeng Yao, Jilong Wang, Shuyu Yang, Cong Wang, Hongyin Zhang, Qifeng Zhang, and Donglin Wang. 2022b. Imitation and Adaptation Based on Consistency: A Quadruped Robot Imitates Animals from Videos Using Deep Reinforcement Learning. *arXiv preprint arXiv:2203.05973* (2022).

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: simple biped locomotion control. *ACM Trans. Graph.* 26, 3 (jul 2007), 105–es. https://doi.org/10.1145/1276377.1276509

Peng Zhao and Michiel van de Panne. 2005. User interfaces for interactive control of physics-based 3d characters. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. 87–94.

Ziyi Zhou, Bruce Wingo, Nathan Boyd, Seth Hutchinson, and Ye Zhao. 2022. Momentum-Aware Trajectory Optimization and Control for Agile Quadrupedal Locomotion. *arXiv preprint arXiv:2203.01548* (2022).

APPENDIX

Model Predictive Controller (MPC) Formulation

The horizontal components of the CoM position $\mathbf{x}_{\text{com}} = \begin{bmatrix} x_{\text{com}} & y_{\text{com}} \end{bmatrix}^\top$ evolve according to

$$\begin{aligned} \ddot{x}_{\text{com}} &= \frac{1}{z_{\text{com}}} \left((\ddot{z}_{\text{com}} + g)(x_{\text{com}} - x_{\text{cop}}) - \frac{\tau_y}{m} \right) \\ \ddot{y}_{\text{com}} &= \frac{1}{z_{\text{com}}} \left((\ddot{z}_{\text{com}} + g)(y_{\text{com}} - y_{\text{cop}}) + \frac{\tau_x}{m} \right) \end{aligned} \tag{11}$$

where $\mathbf{x}_{\text{cop}} = \begin{bmatrix} x_{\text{cop}} & y_{\text{cop}} \end{bmatrix}^\top$ is the location of the CoP on the ground plane, z_{com} is the CoM height, g is the gravitational constant, and τ_x and τ_y are the torques on the CoM caused by changes in angular momentum of the robot with mass m.

Given predictions of the CoM height and the angular momentum of the robot, these equations can be integrated over a preview window of length T. We make no assumptions about the evolution of z_{com} or τ_x and τ_y and instead numerically integrate the horizontal CoM trajectory over the preview window using an integration step of 0.02s. This formulation allows for arbitrarily varying height trajectories and flight phases in possible future extensions to the gait design. However, for the results presented in this work, we make the simplifying assumption that the CoM height and the angular momentum are constant. Note that this still allows the animation to track a varying pelvis height. The assumptions are for the purpose of planning the horizontal components of the CoM only.

The preview footholds are converted to a series of K convex support polygons and their respective time intervals in the preview window. To ensure a dynamically feasible plan, the CoP must always lie inside the active support area. This is achieved by optimizing for two CoP waypoints per support interval: one at its midpoint, constrained to the active support polygon, and one at its end, constrained to the intersection of the active support and the next. Interpolating the resulting $N = 2K$ waypoints linearly, the resulting CoP is guaranteed to remain within the convex support regions. We jointly optimize CoP waypoints

$$\mathbf{z} = \begin{bmatrix} x_{\text{cop},0} & y_{\text{cop},0} & x_{\text{cop},1} & y_{\text{cop},1} & \cdots & x_{\text{cop},N} & y_{\text{cop},N} \end{bmatrix}^\top \tag{12}$$

and the final horizontal CoM position and velocity at the end of the preview window

$$\xi_T = \begin{bmatrix} \mathbf{x}_{\text{com},T}^\top & \dot{\mathbf{x}}_{\text{com},T}^\top \end{bmatrix}^\top \tag{13}$$

in a quadratic program:

$$\begin{aligned} \min_{\mathbf{z}, \xi_T} \quad & \left(\mathbf{A}_T \xi_T - \mathbf{c}_T\right)^\top \mathbf{W}_T \left(\mathbf{A}_T \xi_T - \mathbf{c}_T\right) \\ & + (\mathbf{z} - \mathbf{c})^\top \mathbf{W}_\mathbf{c} (\mathbf{z} - \mathbf{c}) + (\mathbf{F}\mathbf{z})^\top \mathbf{W}_v (\mathbf{F}\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{x}_{\text{cop},0} = \mathbf{x}_{\text{cop}}^i \\ & \mathbf{x}_{\text{com},T} + \mathbf{A}_p \mathbf{z} + \mathbf{b}_p = \mathbf{x}_{\text{com}}^i \\ & \dot{\mathbf{x}}_{\text{com},T} + \mathbf{A}_v \mathbf{z} + \mathbf{b}_v = \dot{\mathbf{x}}_{\text{com}}^i \\ & \mathbf{x}_{\text{cop},n} \in C_n \quad \forall n \in [0, N-1] \end{aligned} \tag{14}$$

$\mathbf{A}_T$ is chosen such that $\mathbf{A}_T \xi_T$ is the final Capture Point (CP), a linear combination of CoM position and velocity [Pratt et al. 2006], at the end of the preview window. The first objective brings the final CP close to the centroid of the last support polygon $\mathbf{c}_T$. The second term of the objective encourages each CoP waypoint to lie close to the centroid of its respective constraint polygon ($\mathbf{c}$ is the vector of the centroid locations). The third term penalizes large CoP velocities with $\mathbf{F}$ being a finite difference matrix. $\mathbf{W}_T$, $\mathbf{W}_\mathbf{c}$, and $\mathbf{W}_v$ are the respective objective weight matrices.

For continuity of the control output, the equality constraints consist of the initial CoP location $\mathbf{x}^i_{\text{cop}}$ as well as initial horizontal CoM position $\mathbf{x}^i_{\text{com}}$ and velocity $\dot{\mathbf{x}}^i_{\text{com}}$. The matrices $\mathbf{A}_p$, $\mathbf{A}_v$, $\mathbf{b}_p$, $\mathbf{b}_v$ are the discrete reverse-time integration matrices for the CoM according to Eq. 11, allowing us to constrain the initial CoM while optimizing for the final CoM $\mathbf{x}_{\text{com},T}$. The final constraint enforces that each CoP waypoint lies within its respective convex constraint polygon $\boldsymbol{C}$.

Task-Space Inverse Dynamics (ID) Solver

Dynamic Constraints. We assume the robot is modeled as an articulated rigid body system with dynamics,

$$\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\tau}\end{bmatrix} = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - \sum_c \mathbf{J}_c^\top \mathbf{f}_c. \tag{15}$$

Here $\boldsymbol{\tau} \in \mathbb{R}^N$ represents the vector of joint torques, $\mathbf{H}(\mathbf{q})$ the joint-space inertia matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ the vector of centrifugal, Coriolis and gravity torques, $\mathbf{f}_c$ the external contact forces, and $\mathbf{J}_c$ the corresponding contact Jacobians.

The centroidal dynamics are described by the Newton-Euler constraints, which equate the rate of change of whole body momentum, $\dot{\mathbf{h}}$, to the sum of external wrenches acting on the system, i.e.

$$\dot{\mathbf{h}} = \sum_c \begin{bmatrix}\mathbf{I}\\ \mathbf{r}_{\times_c}\end{bmatrix}\mathbf{f}_c + \begin{bmatrix}-m\mathbf{g}\\ \mathbf{0}\end{bmatrix}. \tag{16}$$

Here $\dot{\mathbf{h}} = [\dot{\mathbf{l}}^\top \ \dot{\mathbf{k}}^\top]^\top \in \mathbb{R}^6$ is the vector of linear and angular momentum rates, $\mathbf{r}_{\times_c}$ is a cross product matrix computed from the vector pointing from CoM to each contact point, and $-m\mathbf{g}$ is the force of gravity

Contact Model. As in [Pollard and Reitsma 2001], we rely on a polygonal friction cone approximation to optimize the forces at each contact point subject to Coulomb friction constraints. Fig. 15 illustrates the adopted contact model, which assumes contact points are defined at the convex hull of the active support polygon. Each friction cone is approximated by four basis vectors, $\boldsymbol{\beta}_{c,i} \in \mathbb{R}^3$, that span a convex subset of the cone. Valid contact forces are encoded as a vector of negative weights, $\boldsymbol{\rho}_c \in \mathbb{R}^4$, using the mapping,

$$\mathbf{f}_c = \begin{bmatrix}\boldsymbol{\beta}_{c,1} & \boldsymbol{\beta}_{c,2} & \boldsymbol{\beta}_{c,3} & \boldsymbol{\beta}_{c,4}\end{bmatrix}\boldsymbol{\rho}_c. \tag{17}$$

Task-Space Objectives. We assume that a whole-body controller computes a set of task-space acceleration and momentum rate of change objectives to track a desired reference motion. Given the current robot state, $\mathbf{s} = [\mathbf{q}^\top \ \dot{\mathbf{q}}^\top]^\top$, joint and spatial acceleration objectives, $\mathbf{u}_t$, can be expressed as a linear function of the joint

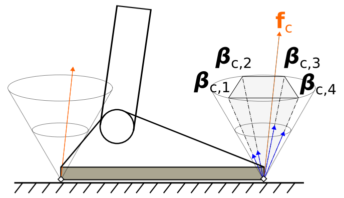
$\mathbf{f}_c$
$\boldsymbol{\beta}_{c,2}$
$\boldsymbol{\beta}_{c,3}$
$\boldsymbol{\beta}_{c,1}$
$\boldsymbol{\beta}_{c,4}$

Fig. 15. **Contact model.** Contact forces acting at points on the convex hull of the foot are expressed as a non-negative, weighted sum of basis vectors, $\boldsymbol{\beta}_{c,i}$, that lie within the friction cone.

.

acceleration vector, i.e.

$$\mathbf{u}_t = \dot{\mathbf{J}}_t\dot{\mathbf{q}} + \mathbf{J}_t\ddot{\mathbf{q}}, \tag{18}$$

where $\mathbf{J}_t$ is a configuration-dependant task Jacobian. Likewise, the centroidal momentum rate can be expressed as,

$$\dot{\mathbf{h}} = \dot{\mathbf{A}}_h\dot{\mathbf{q}} + \mathbf{A}_h\ddot{\mathbf{q}}, \tag{19}$$

where $\mathbf{A}_h$ represents the configuration-dependent centroidal momentum matrix (CMM) [Orin and Goswami 2008].

Task-Space Inverse Dynamics Optimization. Now let $\mathbf{u}$ represent the stacked vector of desired task-space accelerations and momentum rate objectives and let $\mathbf{J}$ represent the corresponding matrix of Jacobians. The proposed inverse dynamics solver resolves the set of task-space objectives by optimizing the vector of joint accelerations and contact forces using the following quadratic program formulation:

$$\min_{\ddot{\mathbf{q}},\boldsymbol{\rho}} \quad (\mathbf{u} - \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\ddot{\mathbf{q}})^\top \mathbf{W}_t \,(\mathbf{u} - \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\ddot{\mathbf{q}}) \tag{20}$$

$$+ \lambda_{\ddot{\mathbf{q}}}\ddot{\mathbf{q}}^\top\ddot{\mathbf{q}} + \lambda_{\boldsymbol{\rho}}\boldsymbol{\rho}^\top\boldsymbol{\rho} + \lambda_{\bar{\boldsymbol{\tau}}}\bar{\boldsymbol{\tau}}^\top\bar{\boldsymbol{\tau}}$$

$$\text{s.t.} \quad \dot{\mathbf{A}}_h\dot{\mathbf{q}} + \mathbf{A}_h\ddot{\mathbf{q}} = \sum_c \begin{bmatrix}\mathbf{I}\\ \mathbf{r}_{\times_c}\end{bmatrix}\mathbf{f}_c + \begin{bmatrix}-m\mathbf{g}\\ \mathbf{0}\end{bmatrix} \tag{21}$$

$$k_l(\mathbf{q}_{\min} - \mathbf{q}) - b_l\dot{\mathbf{q}} \le \ \ddot{\mathbf{q}} \tag{22}$$

$$k_l(\mathbf{q}_{\max} - \mathbf{q}) - b_l\dot{\mathbf{q}} \ge \ \ddot{\mathbf{q}} \tag{23}$$

$$\mathbf{0} \le \ \boldsymbol{\rho} \tag{24}$$

$$\forall_c \ \mathbf{f}_c \le \ \mathbf{f}_{c,\max}, \tag{25}$$

where

- $\boldsymbol{\rho}$ represents the vector of $\boldsymbol{\rho}_c$ weights for each contact point.
- $\mathbf{W}_t$ is the objective weighting matrix.
- $\lambda_{\boldsymbol{\rho}}$, $\lambda_{\ddot{\mathbf{q}}}$, and $\lambda_{\bar{\boldsymbol{\tau}}}$ are regularization weights.
- $\bar{\boldsymbol{\tau}} := -\sum_c \mathbf{J}_c^T\mathbf{f}_c$ represents the component of torque that is induced by the contact forces in Eq. 15. Regularizing $\bar{\boldsymbol{\tau}}$ reduces unnecessary internal forces that otherwise arise as a result of the contact force regularization.
- (21) enforces the centroidal dynamics constraint.
- (22) and (23) enforce soft position limits with second-order dynamics defined by stiffness and damping gains, k_l and b_l.
- (24) enforces the non-negativy constraints of the adopted contact force model.
- (25) limits the max contact force to $\mathbf{0}$ for inactive contacts.

From the optimized $\ddot{\mathbf{q}}$ and $\boldsymbol{\rho}$ values, we derive the final contact force and joint torque solution using Eqs. 17 and 15.