Feature-Based Locomotion Controllers

Martin de Lasa Igor Mordatch Aaron Hertzmann

University of Toronto

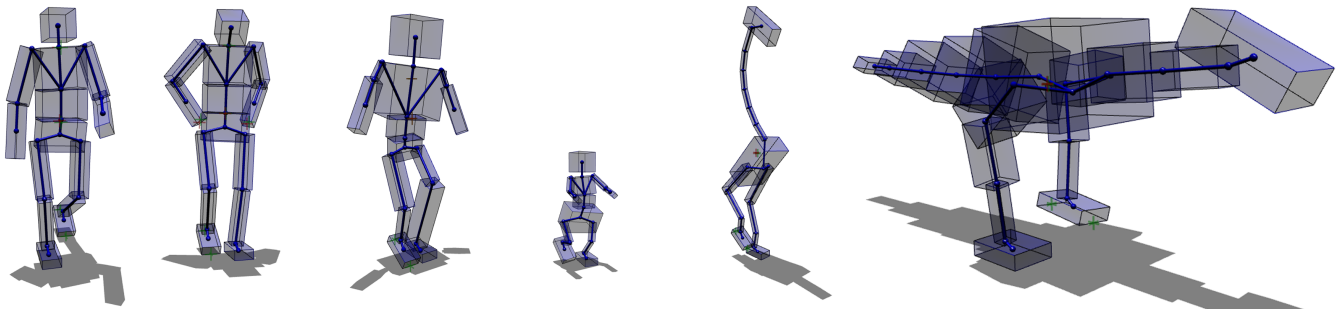

Figure 1: *Controllers based on high-level features can be modified to create new styles, transferred to new characters and are robust to interactive changes in anthropometry. left to right: normal walking, "sad" walking, asymmetric character, "baby," ostrich, dinosaur.*

Abstract

This paper introduces an approach to control of physics-based characters based on high-level features of movement, such as center-of-mass, angular momentum, and end-effectors. Objective terms are used to control each feature, and are combined by a prioritization algorithm. We show how locomotion can be expressed in terms of a small number of features that control balance and end-effectors. This approach is used to build controllers for human balancing, standing jump, and walking. These controllers provide numerous benefits: human-like qualities such as arm-swing, heel-off, and hip-shoulder counter-rotation emerge automatically during walking; controllers are robust to changes in body parameters; control parameters and goals may be modified at run-time; control parameters apply to intuitive properties such as center-of-mass height; and controllers may be mapped onto entirely new bipeds with different topology and mass distribution, without modifications to the controller itself. No motion capture or off-line optimization process is used.

CR Categories: I.3.1 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: Physics-Based Animation, Control, Walking, Jumping, Balancing

1 Introduction

Creating effective, flexible, and realistic locomotion controllers for physics-based characters remains an open problem. Most methods employ joint-space approaches, which parameterize control in terms of individual joint actions. Because individual joint actions combine in a highly non-linear and interdependent way, expressing coordinated or stylized motion in terms of joint-based controllers is extremely difficult. Modifying joint-space controllers for new characters, styles, and tasks often requires extensive retuning or ad-hoc methods of handling redundancy. Due to the these difficulties, some methods turn to motion capture to provide target poses for control. However, these methods are limited to producing motions very similar to the data.

This paper describes an approach to locomotion in which control is expressed in terms of *features*. Each feature describes a high-level, physically-relevant property of character state, such as center-of-mass, angular momentum, or end-effector position. The key questions we address are: what high-level features can be used to create locomotion? How should these features be controlled? We show that locomotion can be expressed as control of a small, intuitive set of features and objectives that can naturally be interpreted in terms of balance and end-effector control. Control of multiple features simultaneously is coordinated by a prioritized optimization scheme that allows feature control to be resolved in strict priority order. A novel formulation of angular momentum control is used for walking, without requiring explicit center-of-pressure specification. We demonstrate balancing, standing jump, and walking controllers.

This control representation provides numerous benefits. Natural properties of human locomotion, such as arm-swing and hip-shoulder counter-rotation, emerge from our model, despite the lack of explicit control for these features. Controllers are robust to mass properties changes. Although some effort is required to produce an initial controller, modifying one is very easy: most parameters have intuitive effects on motion, and walking style can even be modified at run-time. Furthermore, controllers can be transferred to new characters with new topology, often requiring no modification to the controller whatsoever. To our knowledge, none of these capabilities have previously been demonstrated with locomotion controllers, and yet all of our controllers are hand-tuned; no motion capture or costly off-line optimization is required. The generality and flexibility of our feature-space representation suggest that it may be useful for many other problems in animation, including other types of motion, and learning and optimizing control.

http://www.dgp.toronto.edu/~mdelasa/feature

2 Related Work

Most existing locomotion controllers use joint-space representations. These methods use per-joint PD servos, coordinated by a high-level state machine [Faloutsos et al. 2001; Hodgins et al. 1995; Laszlo et al. 1996; Raibert and Hodgins 1991; Wooten 1998; Yin et al. 2007]. This approach has been applied to a wide variety of walking, running, and athletic motions. The combination of high-gain tracking and discrete state machines frequently leads to stiff motions and can be very difficult to tune for natural-looking full-body behaviors. Optimization methods can ease the task of controller tuning [Hodgins and Pollard 1997; Sharon and van de Panne 2005; Wang et al. 2009], but require expensive optimization processes for each new controller or character. Due to these difficulties, many authors have turned to motion capture data to define motion style [Abe et al. 2007; da Silva et al. 2008a; da Silva et al. 2008b; Muico et al. 2009; Sok et al. 2007; Zordan and Hodgins 2002]. Such methods achieve impressive realism, but are limited to motions similar to recorded trajectories. Robust adaptation of mocap tracking controllers, far from reference motion, is an open research problem.

An alternative to joint-space methods is to formulate control in terms of high-level tasks. The original task-space methods, developed for control of redundant robotic manipulators, are based on null-space projection operators, e.g., [Abe and Popović 2006; Hsu et al. 1989; Khatib 1987; Liegeois 1977; Nakamura et al. 1987; Shkolnik and Tedrake 2008]; for a survey, see [Nakanishi et al. 2008]. Most null-space projection methods do not handle unilateral constraints; doing so is extremely complex and restricted to special cases [Mansard and Khatib 2008]. For restricted applications, iterative extensions to projection methods have been proposed (e.g., [Baerlocher and Boulic 2004]) that handle unilateral constraints by solving problems multiple times until all violations are eliminated. No performance guarantees exist for these methods.

An alternative to null-space projection is to combine all tasks into a single quadratic program (QP) [Azevedo et al. 2002; Abe et al. 2007; da Silva et al. 2008a; da Silva et al. 2008b; Fujimoto et al. 1998; Kudoh et al. 2006; Macchietto et al. 2009] or non-convex optimization [Jain et al. 2009]. These methods achieve impressive results in highly-constrained balancing and reaching scenarios. To date, the only QP-based methods for locomotion control are limited to tracking motion capture data. Our work shows how these approaches can be extended to perform locomotion without motion capture, while substantially increasing the flexibility of the controllers. Furthermore, previous methods use weighted combinations of objectives. In practice, different objective terms can "fight" against each other, making it very difficult to adjust weights in complex situations. This becomes especially problematic for locomotion, which requires many objectives. We propose a solver, extending results from [de Lasa and Hertzmann 2009; Kanoun et al. 2009], that handles weighted objectives, strict prioritization, and unilateral constraints in a single framework. Our approach is similar in many respects to task-space approaches, though we prefer the term "feature-based" to avoid confusion with other uses of "task."

Our work is similar to previous approaches, differing in key details. Pratt et al. [2001] introduce Virtual Model Control (VMC), based on describing control in terms of abstract virtual features. Since VMC does not require detailed knowledge of inertial properties, it has modest computational requirements and is straightforward to implement. Unfortunately, because VMC does not explicitly deal with redundancy resolution, and neglects Coriolis and gravitational forces, it can be challenging to apply to complex full-body characters/motions or highly dynamic actions. An alternative, which considers dynamic system properties is Operational Space Control (OSC) [Khatib 1987]. Sentis [2007] extends OSC to whole-body motion for simulated humanoid robots, demonstrating a number of balancing and locomotion behaviors that handle numerous competing goals simultaneously. Because OSC handles tasks in strict priority order, degrees-of-freedom (DOF) for complex behaviors can exceed available system DOF, forcing the control designer to sacrifice control goals. By supporting both prioritization and weighted objective combination our approach mitigates this problem. Our formulation builds on the strengths of OSC in a number of other ways: predictive elements can be included; unilateral constraints are supported; and motion near mechanical singularities is correctly handled.

3 Feature-Based Control

We now describe the design of control in terms of high-level features. These features include task-relevant quantities such as center-of-mass (COM), angular momentum (AM), and end-effector (EE) motion. In the following sections, we show how to build balance, standing jump, and walking controllers using this representation. For example, walking is specified in terms of just a few, high-level features: propel the COM forward and move the feet, while minimizing AM and muscle torques for balance, stability, and style.

At each instant of a simulation, the controller determines the following vector of unknowns:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau}^T & \ddot{\mathbf{q}}^T & \boldsymbol{\lambda}^T \end{bmatrix}^T \tag{1}$$

where $\boldsymbol{\tau}$ denotes joint torques, $\ddot{\mathbf{q}}$ denotes joint accelerations, and $\boldsymbol{\lambda}$ denotes weights in the linearized friction cone bases. The selected value of $\mathbf{x}$ must satisfy dynamics and unilateral constraints:

$$C(\mathbf{x}) = 0, \;\; \mathbf{D}\mathbf{x} + \mathbf{f} \geq 0 \tag{2}$$

described in Sec. 7. Variables may vary with time t, which we omit from these equations for brevity.

3.1 Features and Objectives

To solve for unknowns (1), we formulate an optimization in terms of a N quadratic objective functions $E_i(\mathbf{x})$. Each objective is formulated in terms of some feature $\mathbf{y}$ of the motion, such as COM, foot position, AM, or arm torque. We use 4 different types of objectives: i) Setpoint, ii) Target, iii) Angular Momentum, and iv) Minimum Torque objectives. We now describe these objectives.

Setpoint Objective. This objective applies linear control to some feature of pose. In this case, a feature is any algebraic function:

$$\mathbf{y} = f(\mathbf{q}) \tag{3}$$

of generalized coordinates $\mathbf{q}$. To define a setpoint objective, we first specify a "reference" value $\mathbf{y}_r$ for the feature. A desired acceleration is computed by linear control:

$$\ddot{\mathbf{y}}_d = k_p(\mathbf{y}_r - \mathbf{y}) - k_v\dot{\mathbf{y}} \tag{4}$$

where k_p and k_v are feature-specific gains. In most of our examples, we set $k_v = 2\sqrt{k_p}$. The objective then measures the deviation of the feature acceleration from its desired value:

$$E(\mathbf{x}) = ||\ddot{\mathbf{y}}_d - \ddot{\mathbf{y}}||^2 \tag{5}$$

where the actual feature acceleration $\ddot{\mathbf{y}}$ can be computed by differentiating $f(\mathbf{q})$ twice, to yield $\ddot{\mathbf{y}} = \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}}$, and $\mathbf{J}$, $\dot{\mathbf{J}}$ are the relevant Jacobian and its time derivative. For example, we may define a COM feature $\mathbf{y}^{com}$ and then servo it to a setpoint $\mathbf{y}_r^{com}$. This is normally used to maintain a feature at a fixed value, e.g., maintaining COM over the feet.

Target Objective. When we wish to servo a feature to a distant setpoint, directly applying the setpoint objective would cause large impulsive forces. One alternative is to servo along a smooth time-varying trajectory $\mathbf{y}_r(t)$, e.g., using (4). However, this requires time-consuming trajectory design and servo-gain tuning.

To avoid this manual design step, we propose a method that directly computes $\ddot{\mathbf{y}}_d$ by solving a simple boundary value problem to move the feature from current state $(\mathbf{y}_0, \dot{\mathbf{y}}_0)$ to target state $(\mathbf{y}_T, \dot{\mathbf{y}}_T)$, in duration T. In practice, we use this type of objective to move a feature between two target locations, such as when the COM height is controlled during jumping (Sec. 5) or feet are guided between footholds in walking (Sec. 6). Solving for each element of $\mathbf{y} \in \Re^D$ separately, and assuming a solution of the form:

$$\ddot{y}_d(t) = \left(1 - \frac{t}{T}\right) a + \frac{t}{T} b \tag{6}$$

we seek constants a and b generating the desired motion. Integrating (6) twice, isolating a and b, and writing in matrix form yields:

$$\begin{bmatrix} T^2/3 & T^2/6 \\ T/2 & T/2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_T - y_0 - \dot{y}_0 T \\ \dot{y}_T + \dot{y}_0 \end{bmatrix}. \tag{7}$$

At each instant of the simulation, we recompute a and b, by solving the above linear system in a least-square sense using a singularity-robust pseudoinverse. This avoids instabilities that may occur as $T \to 0$. For large values $T \gg 0$, this approach is equivalent to cubic Hermite interpolation. Then, the objective for each feature is $e(\mathbf{x}) = (\ddot{y}_d(0) - \ddot{y})^2$. Note that this calculation determines the target acceleration independent of the object's mass.

Angular Momentum Objective. As shown by Macchietto et al. [2009], regulating AM improves balance stability and robustness, via strategies such as windmilling. However, their formulation has some limitations. First, they require a reference center-of-pressure (COP) to be specified. Although a fixed COP is sufficient for static motions such as balancing, more complex actions (e.g., walking) require time-varying COP trajectories [Adamczyk et al. 2006]. We propose an alternate AM control formulation that produces good results not only for balance, but also jumping and walking. Second, their method relies on knowledge of contact forces prior to computing control, limiting the technique to simulators with penalty-based ground contact models. To avoid interpenetration and tuning of contact parameters, modern simulators use LCP-based contact models (Sec. 9). These simulators compute contact forces *after* joint-torques are specified. We were unable to make the method of [Macchietto et al. 2009] work in our LCP-based simulator. Our method does not rely on knowledge of contact forces for AM regulation. Lastly, since we do not explicitly introduce coupling between linear and AM regulation, our approach is simpler to implement.

Based on biomechanical observations of human motion [Herr and Popovic 2008; Popovic et al. 2004], we propose an AM objective that seeks to regulate whole-body changes in AM to a desired value using linear control:

$$E_{AM}(\mathbf{x}) = ||\dot{\mathbf{L}}_d - \dot{\mathbf{L}}||^2 \tag{8}$$

$$\dot{\mathbf{L}}_d = k_p(\mathbf{L}_r - \mathbf{L}) \tag{9}$$

where $\mathbf{L}_r$ is the reference angular momentum about the COM. For balancing and walking, we set $\mathbf{L}_r = 0$, which corresponds to damping rotations [Popovic et al. 2004]. For acrobatic jumping manoeuvres, we specify non-zero $\mathbf{L}_r$ to create twisting behavior (Sec. 5).

To calculate $\dot{\mathbf{L}}$, we express the AM about the COM [Orin and Goswami 2008] as:

$$\mathbf{L} = \mathbf{P}\mathbf{J}\dot{\mathbf{q}} \tag{10}$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1^T & \dots & \mathbf{J}_n^T \end{bmatrix}^T \tag{11}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_1 & \dots & \mathbf{I}_n \end{bmatrix} \tag{12}$$

where $\mathbf{I}_i$ is the inertia about the COM of the i^{th} link, $\mathbf{J}_i$ maps joint velocities to angular velocities about each link's COM, and $\dot{\mathbf{q}}$ is the vector of joint velocities. Differentiating (10), we obtain:

$$\dot{\mathbf{L}} = \mathbf{P}\mathbf{J}\ddot{\mathbf{q}} + (\dot{\mathbf{P}}\mathbf{J} + \mathbf{P}\dot{\mathbf{J}})\dot{\mathbf{q}} \tag{13}$$

$$\dot{\mathbf{P}} = \begin{bmatrix} \omega_1 \times \mathbf{I}_1 & \dots & \omega_n \times \mathbf{I}_n \end{bmatrix} \tag{14}$$

where ω_i is the angular velocity of the i^{th} link.

Minimum Torque Objective. For stylistic reasons, we use an additional objective to minimize joint torques:

$$E_\tau(\mathbf{x}) = ||\boldsymbol{\tau}||^2 = || \begin{bmatrix} \mathbf{S} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{x} ||^2 \tag{15}$$

where the selection matrix $\mathbf{S}$ is a square matrix of zeros, with ones on the diagonal for joints we want to make passive. We use this objective in walking to produce natural-looking arm sway, by minimizing shoulder and elbow torques (cf. Sec. 6).

3.2 Prioritized Optimization

Previous methods combine objectives by weighted combination with weights α_i:

$$E(\mathbf{x}) = \sum_i \alpha_i E_i(\mathbf{x}) \tag{16}$$

subject to dynamics constraints. This formulation is used by several previous methods for balancing and motion capture tracking [Abe et al. 2007; da Silva et al. 2008b; da Silva et al. 2008a; Macchietto et al. 2009]. However, this approach can require substantial effort to properly tune parameters (e.g., [Jain et al. 2009]), and can lead to problems with objective terms that "fight." Instead, we apply *prioritized optimization* [de Lasa and Hertzmann 2009; Kanoun et al. 2009], also known as lexicographic optimization [Marler and Arora 2004].

Given an ordered list of N objectives, $E_1(\mathbf{x}), ..., E_N(\mathbf{x})$, prioritized optimization finds the solution minimizing each objective without conflicting with higher-priority objectives. This is defined recursively as:

$$\begin{aligned} h_i = & \min_{\mathbf{x}} E_i(\mathbf{x}) \\ \text{subject to } & E_k(\mathbf{x}) = h_k, \ \forall k < i \\ & C(\mathbf{x}) = 0 \\ & \mathbf{D}\mathbf{x} + \mathbf{f} \geq 0. \end{aligned} \tag{17}$$

In other words, the first optimization problem is $h_1 = \min_{\mathbf{x}} E_1(\mathbf{x})$, subject to the dynamics constraints. Subsequent steps are required to achieve this minimum: the second problem is $h_2 = \min_{\mathbf{x}} E_2(\mathbf{x})$, subject to $h_1 = E_1(\mathbf{x})$ and the dynamics constraints. This optimization is repeated recursively, returning the point $\mathbf{x}_N$ that solves the N-th problem. In Sec. 8, we present an algorithm for solving prioritized optimization in an efficient and stable manner.

Table 1: ***Objectives for Balance and Jumping.*** *Each objective corresponds to the control of one feature.*

Priority	Objective	Role
1	$E_{contact}$	Keep feet planted
2	$E_{com\parallel}$	Keep COM over base-of-support
	$E_{com\perp}$	Control COM height
3	E_{AM}	Minimize AM changes (Sec. 3.1)
	E_{pose}	Servo joints to a rest pose
	E_{ankle}	Servo ankles rel. to COM **(jump)**

4 Balance

We begin with a basic balancing control task; this controller will later form the basis for our jumping and walking controllers. Our balancing strategies are based on previous results [Kudoh et al. 2006; Macchietto et al. 2009], with a number of improvements.

Balancing involves several actions, summarized in Table 1. We divide actions into 3 priority levels, combining objectives at the same level with equal weight, i.e., $E_1(\mathbf{x}) + E_2(\mathbf{x})$. With the exception of E_{AM}, we use setpoint objectives for remaining features (Sec. 3.1).

The first priority level ensures the feet remain planted on the ground. This is enforced by an objective, $E_{contact}$, which keeps each foot's sole on the contact surface. This condition is enforced using a setpoint objective (3.1), applied to each point on the bottom of the foot. The target value $\mathbf{y}_r^{contact}$ is obtained by projecting each of these points onto the contact surface.

The second priority level is responsible for regulating linear momentum. To accomplish this, $E_{com\parallel}$ regulates the projection of the COM to the centroid of the base-of-support (BOS). Specifically, $\mathbf{y}_r^{com\parallel}$ is set to the average location of the contact points. Control of the COM component perpendicular to the contact surface uses a similar objective, $E_{com\perp}$; $y_r^{com\perp}$ is chosen based on the character's stature ($\approx 80\%$ max COM height). Varying this height produces deep-knee squats.

At the lowest priority level, two tasks are used: an AM task E_{AM} compensates for unwanted rotations, and a rest pose objective E_{pose} (Sec. 3.1) resolves remaining whole-body redundancies. The rest pose objective is a special case of the setpoint objective, which simplifies to $E_{pose} = ||k_p(\mathbf{q}_r - \mathbf{q}) - k_v\dot{\mathbf{q}} - \ddot{\mathbf{q}}||^2$. See da Silva et al. [2008b] for a discussion of how to compute position error (i.e., $\mathbf{q}_r^i - \mathbf{q}^i$) for spherical joints. We select a target for $\mathbf{y}_r^{pose} = \mathbf{q}_r$ that commands most joints to a neutral pose; knees are bent slightly to encourage rotation away from joint-limits. A set of low-stiffness PD servo gains, k_p^{pose}/k_v^{pose}, are used for all reported behaviors. Contact constraints (Sec. 7.2) are also enforced for all contact points close to the ground.

5 Standing Jumps

We now describe a variety of jumping behaviors (Figure 5), created by simple modifications to the balance controller described above. A state machine determines reference values for individual balance controller features. Using the state machine, COM height is first lowered and then quickly raised, producing the jump. An additional objective, E_{ankle}, controls the ankle position in mid-air. Varying reference values yields different jumps, including twisting, in-air kicks/splits, and forward jumps.

The state machine for jumping includes the following stages: standing, compression, thrust, ascent, descent, and impact (Figure 2). During compress, we use the basic balance controller, but lower the COM height, so that the character crouches. Then, thrust begins: $y_r^{com\perp}$ is quickly raised, while continuing to balance. When the COM passes a threshold height, we enter the ascent state. In this stage, COM tracking and contact objectives are disabled (i.e., $\alpha_{com} = \alpha_{contact} = 0$). Simultaneously, ankle control, E_{ankle} is enabled (i.e., $\alpha_{ankle} = 10$). This keeps the ankles at a fixed position relative to the COM, throughout flight, improving stability upon landing. During descent, and once the feet are about to touchdown, ankle rest pose is updated to match the ground slope. Upon impact, we disable E_{ankle} and re-enable E_{COM} and $E_{contact}$ (i.e., $\alpha_{ankle} = 0$, $\alpha_{com} = \alpha_{contact} = 1$). The COM motion in this phase is determined by the balance controller, which helps decelerates the body. Once the COM starts moving upwards, a target objective $E_{com\perp}$ is used to bring the character back to a standing posture. All changes to COM height are done using a target objective. This avoids the need for careful trajectory design or gain tuning. For the standing jump, AM damping (i.e., $\mathbf{L}_r = \mathbf{0}$) is applied throughout the motion.

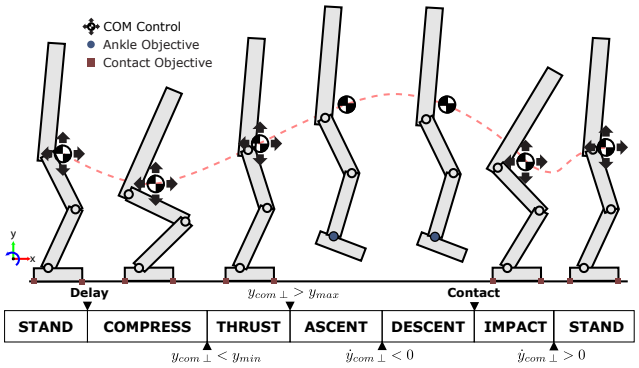

Figure 2: ***Planar illustration of jump controller actions/state machine.*** *State transition are based on COM state and contact events.*

Jump with twist. To create twisting jumps, non-zero vertical AM is commanded about the COM during thrust. The greater the AM at take-off, the more the character will rotate in the air. Sample $90^\circ$and 180° twisting jumps are shown in the accompanying video.

Jumping kicks and splits. To create scissor kick or air-split motions (Figure 5), standing-jump controller values for $\mathbf{y}_r^{ankle}$ are varied throughout flight. To do this, we rotate the ankle offset relative to the COM about a user-specified axis, reversing the direction of motion once descent begins. Accompanying videos show natural counter-rotation of the torso during ballistic motion.

Forward Jumps. Jumps that produce large COM displacements can be created by further extending the acrobatic controllers. During thrust, a small displacement in COM position is commanded in the direction of desired travel. During flight, ankle position trajectories are modified to prepare for landing. This technique can be used to create jumps with forwards displacements. See the accompanying video for examples of this type of motion.

6 Walking

We now describe a locomotion controller capable of walking at different speeds, with different stepping heights, and in a variety of styles. The features used by this controller can be divided into two

categories. First, foot and COM features are controlled to drive the character forward. A state machine governs targets for these features. The remaining objectives are used to maintain balance and control additional nuances of style. These objectives are independent of the state machine; many of them are derived from the balancing controller in Sec. 4. The complete list of features/objectives is summarized in Table 2.

6.1 State-dependent objectives

Control of the feet and the COM is governed by a state machine. Each foot can be in one of four states: swing, plant, support, and heeloff. State transitions are determined as shown (Fig. 3). At specific state machine transitions, targets for the feet and COM are recomputed, based on user-specified properties of gait such as step length and duration.

Foot control. Two objectives are used to control the feet, $E_{contact}$ and E_{swing}. When a foot is in one of the contact states (plant, support, heeloff), the $E_{contact}$ objective enforces contact conditions, and is implemented as in the balancing controller. Depending on the current state, either the front, back, or all four corners of the foot are kept in contact with the ground (cf. Fig. 3).

When a foot is in the swing phase, the E_{swing} target objective controls its trajectory. This objective first raises the foot to a user-specified step-height h and then lowers it, all the while moving the foot forward. One target objective is used for the toe, and one for the heel. The desired swing phase duration T is set by the user (cf. Table 2).

More specifically, the target location $\mathbf{y}_r^{swing}$ for the heel is split into a component parallel to the ground plane $\mathbf{y}_r^{swing\,\parallel}$ and a target height $y_r^{swing\perp}$. The parallel component is determined when the foot enters the swing phase, as an offset from $\mathbf{y}^{stance}$, the heel's position at the start of the swing phase, as:

$$\mathbf{y}_r^{swing\,\parallel} = \mathbf{y}^{stance} - 2\,\alpha\,l_{hip}\,\mathbf{d}_\perp + l_{step}\,\mathbf{d}_\parallel, \qquad (18)$$

where $\mathbf{d}_\parallel$ is the direction of travel, $\mathbf{d}_\perp$ is the right-handed direction perpendicular to $\mathbf{d}_\parallel$, and l_{hs} is the stride width. The indicator α is $+1/-1$ for the right and left swing swing foot, respectively. For the first half of the swing phase duration, $y_r^{swing\perp}$ is set to a user-specified step-height h. For the remaining duration, it is set to zero. The target objective for the toe is identical to that of the heel, but delayed by a small amount.

Note that the swing foot objectives do not precisely specify foot motion. Foot motion can deviate for many reasons, including conflicts with higher-priority tasks, unmodelled disturbances, and inconsistencies between commands and the character's foot size. Instead, objectives are meant to "guide" the feet in a sufficient way to avoid toe-stubbing, while moving the foot to the next foot plant.

COM control. The COM is controlled along a sinusoidal path that propels the character forward, placing the COM over the stance foot for balance. Whenever a swing foot enters the plant phase, this target trajectory is recomputed. To synchronize the COM and the feet, the desired COM trajectory duration is set to the actual duration of the previous swing phase, $T_{com}^i = T_{swing}^{i-1}$. This simple feedback mechanism encourages the character to enter a stable limit-cycle after only a few steps. The duration of the COM trajectory is parameterized with a phase variable $\phi \in [0...1]$.

COM motion parallel to the ground plane is commanded along a trajectory:

$$\mathbf{y}_r^{com\parallel}(\phi) = \phi\mathbf{p}_1 + (1-\phi)\mathbf{p}_0 + \alpha l_{hip}\sin(\pi\phi)\mathbf{d}_\perp \qquad (19)$$

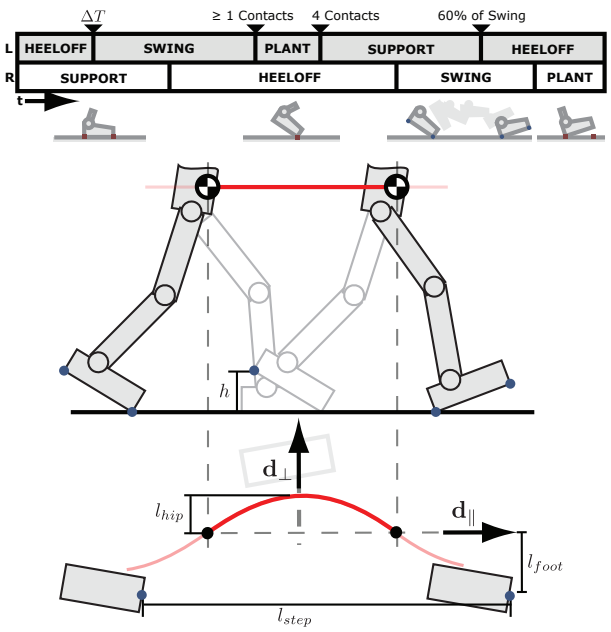

Figure 3: ***Walking Actions and Parameters. Top:*** *A state machine is used to coordinate contact ($E_{contact}$), COM ($E_{com\parallel}$), and stepping (E_{swing}) objectives.* ***Middle/Bottom:*** *Illustration of parameters (cf. Table 2) used to specify swing targets $\mathbf{y}_r^{swing}$ and COM trajectory $\mathbf{y}_r^{com\parallel}$.*

relative to the initial COM position $\mathbf{p}_0$, with:

$$\mathbf{p}_1 = \mathbf{y}^{stance} - \alpha\,l_{foot}\,\mathbf{d}_\perp + \frac{1}{2}\,l_{step}\,\mathbf{d}_\parallel \qquad (20)$$

where l_{hip} and l_{step} specify hip shift and step length respectively (Table 2). In our walking controller, we leave the COM motion perpendicular to the ground plane unconstrained. This allows the walking motion to pivot on the stance leg, rather than walking with some predefined height or attempting to hand-craft trajectories that may conflict with the system dynamics. To adjust walking height, we vary the knee angle in E_{pose}.

6.2 State-independent objectives

The remaining objectives are defined independently of the state machine. As with in-place motions, we regulate AM to improve balance (Sec. 3.1). Biomechanical studies have indicated that AM is tightly regulated during human walking [Herr and Popovic 2008]. This quantity has also proven useful in control optimization [Wang et al. 2009]. To our knowledge, ours is the first method which actively regulates AM for human locomotion. For style, an additional objective E_{arms} is also included to encourage passive arm motion. E_{arms} uses the minimum torque objective (Sec. 3.1), selecting shoulder and elbow joints via the matrix $\mathbf{S}$. The combination of AM damping and minimum upper-body torque control produces natural-looking in-phase arm swing and hip-torso counter oscillation. Two additional objectives are used. To control the character's posture, we use a setpoint objective $E_{trunk\parallel}$ specifying the location of the base of the neck relative to the COM:

$$\mathbf{y}_r^{trunk\,\parallel} = \mathbf{y}_r^{com\parallel} + \mathbf{d}_\parallel\,l_{trunk}. \qquad (21)$$

Lastly, a setpoint objective E_{head} is used to stabilize the orientation of the head [Pozzo et al. 1990]. A quaternion setpoint $\mathbf{q}_{head}$

Table 2: ***Objectives and Parameters for Walking. Top:*** *Walking Objectives* ***Bottom:*** *Walking controller parameters. See Fig. 3 for an illustration of these quantities. Quantities in parameter table are commanded values. These will differ from actual values.*

Priority	Objective	Task
1	$E_{contact}$	Keep stance foot planted
2	$E_{com\parallel}$	Move COM parallel to ground plane
	$E_{trunk\parallel}$	Control neck position wrt COM
3	E_{swing}	Control swing foot
	E_{AM}	Dampen angular momentum
	E_{pose}	Servo joints to rest pose
	E_{arms}	Minimize arm torques
	E_{head}	Servo global orientation of head

Parameter	Symbol	Value
Swing Duration	T	0.3 - 0.7 s
Step Length	l_{step}	0 - 0.7 m
Swing Height	h	0 - 0.1 m
Foot Spread	l_{foot}	0 - 0.03 m
Swing Delay	ΔT	0 - 0.3 s
Hip Shift	l_{hip}	0 - 0.3 m
COM offset	l_{trunk}	± 0.3 m
Head Orientation	$\mathbf{q}_{head}$	$\pm 180^\circ$

specifies the desired orientation of the head about $\mathbf{d}_\perp$, while also pointing the head in the direction of travel.

7 Dynamics Constraints

We now describe the dynamics constraints used by our optimization (Sec. 3.2).

7.1 Equality Constraints

The equations of motion:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} + \mathbf{J}_c^T \mathbf{f}_c \tag{22}$$

relate joint accelerations $\ddot{\mathbf{q}}$, joint torques $\boldsymbol{\tau}$, and contact forces $\mathbf{f}_c$. The quantities $\mathbf{M}$ and $\mathbf{h}$ are the joint space inertia matrix, Coriolis/centrifugal and gravitational forces respectively. The contact force Jacobian $\mathbf{J}_c$ maps generalized velocities $\dot{\mathbf{q}}$ to world space Cartesian velocities at contact points. For brevity, we omit dependence on $\mathbf{q}$ and $\dot{\mathbf{q}}$ in the remainder of the paper. Contact forces:

$$\mathbf{f}_c = \mathbf{V}\boldsymbol{\lambda} \tag{23}$$

are expressed in the basis of the linearised friction cone $\mathbf{V}$ [Abe et al. 2007]. Substituting (23) and (1) into (22), and rearranging gives the equality constraints:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} & \mathbf{J}_c^T\mathbf{V} \end{bmatrix} \mathbf{x} - \mathbf{h} = 0. \tag{24}$$

7.2 Inequality Constraints

To accurately solve for control torques, it is necessary to account for contact forces arising from joint-limits and ground interaction. Together, these define the inequality constraints $\mathbf{D}\mathbf{x} + \mathbf{f} \geq 0$ used by our simulator.

Ground Contact. Ground contact forces must be strictly repulsive and respect friction [Abe et al. 2007; da Silva et al. 2008b; Fang and Pollard 2003]. In our experience, strictly enforcing zero acceleration at contact points [Abe et al. 2007] leads to numerically sensitive/infeasible problems. We have found that using non-penetration constraints [Baraff 1994], expressed in the linearised friction cone basis, produces more stable results. These conditions can be summarized by the inequality constraints:

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{25}$$

$$\mathbf{a}_c \equiv \mathbf{V}^T\mathbf{J}_c\ddot{\mathbf{q}} + \mathbf{V}^T\dot{\mathbf{J}}_c\dot{\mathbf{q}} + \dot{\mathbf{V}}^T\mathbf{J}_c\dot{\mathbf{q}} \geq \mathbf{0} \tag{26}$$

Although complementarity constraints are not enforced (i.e., $\boldsymbol{\lambda}^T\mathbf{a}_c = \mathbf{0}$),we have not found this simplification to adversely impact controller performance or stability.

Joint Limits. To prevent the character from assuming unnatural postures, we include additional constraints in the contact Jacobian:

$$\mathbf{J}_c' = \begin{bmatrix} \mathbf{J}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{lim} \end{bmatrix} \tag{27}$$

for all joints at limits. For a joint, whose generalized coordinate configuration q_i, has range $q_l \leq q_i \leq q_u$, we set the rows of $\mathbf{J}_{lim}$ to the standard basis $\mathbf{e}_i$ when $q_i = q_l$ and to $-\mathbf{e}_i$ when $q_i = q_u$. This increases the number of $\boldsymbol{\lambda}$s in $\mathbf{x}$.

Torque Limits. Bounds on torques help prevent superhuman feats of strength and are expressed as: $-\boldsymbol{\tau}_{max} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max}$.

8 Prioritized Optimization

We now introduce a new algorithm for prioritized optimization. Our goal is to solve for the optimizer $\mathbf{x}_N$ of the problem defined in (17). At first glance, the constraints $E_k(\mathbf{x}) = h_k$ are quadratic constraints, which in general require non-convex optimization. The key to solving prioritized optimization efficiently is to observe that optimal solutions to a quadratic must lie on a linear subspace (Figure 4). Previous authors have taken two different approaches to exploit this observation.

8.1 Linear Constraint Method

The approach of Kanoun et al. [2009] can be summarized as follows. Each objective (5) is quadratic and positive semi-definite, and thus can be written as:

$$E_k(\mathbf{x}) = ||\mathbf{A}_k\mathbf{x} - \mathbf{b}_k||^2 \tag{28}$$

for some $\mathbf{A}_k$ and $\mathbf{b}_k$. Let $\mathbf{x}_k^*$ be an optimizer of the k-th recursive optimization problem, and $h_k = E_k(\mathbf{x}_k^*)$. Then the subsequent constraint $E_k(\mathbf{x}) = h_k$ is equivalent to:

$$\mathbf{A}_k\mathbf{x} - \mathbf{b}_k = \mathbf{A}\mathbf{x}_k^* - \mathbf{b}_k \tag{29}$$

or, simply, $\mathbf{A}_k(\mathbf{x} - \mathbf{x}_k^*) = 0$. Substituting (29) into (17) yields:

$$\begin{aligned} h_i = &\min_{\mathbf{x}} E_i(\mathbf{x}) \\ \text{subject to } &\mathbf{A}_k(\mathbf{x} - \mathbf{x}_k^*) = 0 \; \forall k < i \\ &C(\mathbf{x}) = 0 \\ &\mathbf{D}\mathbf{x} + \mathbf{f} \geq 0 \end{aligned} \tag{30}$$

which can then be solved recursively using N QPs. This approach applies an increasing number of linear constraints at each step, which often lead to numerical instability. Kanoun et al. also

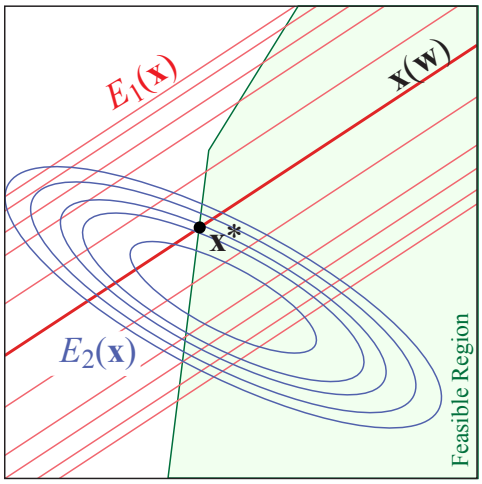

Figure 4: ***Illustration of prioritized optimization with two objectives.*** *The minima of $E_1(\mathbf{x})$ (in red) lie in a linear subspace $\mathbf{x}(\mathbf{w})$, shown as a dark red line. The solution must lie on this subspace. The dynamics constraints require the solution to lie the green polytope. The solution that minimizes $E_2(\mathbf{x})$, subject to these two constraints, is given by $\mathbf{x}^*$.*

describe a method for handling prioritized inequality constraints, which we do not discuss here. In practice, we have noticed this method has difficulties with rapidly changing constraints, such as when the number of contact points changes; noisy torques are generated often resulting in simulation failure. This formulation is also sensitive to the choice of QP solver, failing on active-set methods.

8.2 Reparameterization Method

A second approach to prioritized optimization was proposed by de Lasa and Hertzmann [2009], based on reparameterizing the problem in terms of each optimal linear subspace. We now generalize this method to handle inequality constraints, which were not handled by the original formulation. The idea is as follows. Suppose we have two prioritized objectives, $E_1(\mathbf{x})$ and $E_2(\mathbf{x})$. The space of optimal solutions to $E_1(\mathbf{x}) = ||\mathbf{A}_1\mathbf{x} - \mathbf{b}_1||^2$, subject to the dynamics constraints, can be parameterized by a vector $\mathbf{w}$ as:

$$\mathbf{x}(\mathbf{w}) = \mathbf{C}_1\mathbf{w} + \mathbf{d}_1 \tag{31}$$

where $\mathbf{C}_1 = \text{null}(\mathbf{A}_1)$ is a nullspace basis for $\mathbf{A}_1$, and $\mathbf{d}_1$ is any minimizer of $E_1(\mathbf{x})$ that satisfies the dynamics constraints. The solution $\mathbf{d}_1$ can be obtained by solving a QP, and $\mathbf{C}_1$ can be computed using Singular Value Decomposition (SVD). Then, any choice of $\mathbf{w}$ that satisfies the inequality constraints produces a valid minimizer of $E_1(\mathbf{x})$, although not all values of $\mathbf{w}$ satisfy the dynamics constraints. Substituting $\mathbf{x}(\mathbf{w})$ into $E_2(\mathbf{x})$ gives:

$$E_2(\mathbf{w}) = ||\mathbf{A}_2\mathbf{x}(\mathbf{w}) - \mathbf{b}_2||^2 \tag{32}$$

$$= ||\mathbf{A}_2\mathbf{C}_1\mathbf{w} + \mathbf{A}_2\mathbf{d}_1 - \mathbf{b}_2||^2 \tag{33}$$

The second objective is now reparameterized so as to restrict it to the space of solutions to the first objective. Likewise, substituting $\mathbf{x}(\mathbf{w})$ into the inequality constraints gives reduced constraints

$$\mathbf{D}\mathbf{C}_1\mathbf{w} + \mathbf{D}\mathbf{d}_1 + \mathbf{f} \geq 0 \tag{34}$$

The optimal solution is then given by solving a second QP:

$$\mathbf{d}_2 = \arg\min_{\mathbf{w}} E_2(\mathbf{w})$$
$$\text{subject to } \mathbf{D}\mathbf{C}_1\mathbf{w} + \mathbf{D}\mathbf{d}_1 + \mathbf{f} \geq 0\,. \tag{35}$$

Algorithm 1: Constrained Quadratic Prioritized Solver

```
1  C̄ ← I , d̄ ← 0
2  for i = 1 to N do
3  |   Ā_i ← A_i C̄
4  |   b̄_i ← b_i − A_i d̄
5  |   d_i ← arg min_w ||Ā_i w − b̄_i||^2
6  |         subject to D_0 C̄ w + D_0 d̄ + f_0 ≥ 0
7  |   if problem is infeasible then
8  |   |   return d̄
9  |   end
10 |   d̄ ← d̄ + C̄ d_i
11 |   if Ā_i is full rank then
12 |   |   return d̄
13 |   end
14 |   C̄ ← C̄ null(Ā_i)
15 end
16 return d̄
```

Substituting $\mathbf{d}_2$ into (31) gives the solution in the original space:

$$\mathbf{x}^* = \mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1 \tag{36}$$

Given N objectives, the same process may be repeated recursively. For example, when there are three objectives, the solutions to (33) are parameterized by another linear subspace, and $E_3(\mathbf{w}_2)$ minimized within this space. An efficient algorithm for computing the solution for N objectives is given in Algorithm 1. This is a straightforward generalization of the procedure of de Lasa and Hertzmann [2009]; the only difference is that inequality constraints are now enforced when solving for $\mathbf{d}_i$ in lines 5-8.

We find that directly including the equality constraints $C(\mathbf{x}) = 0$ (Eq. 24) as constraints in the QP optimization can lead to infeasible solutions due to numerical issues, particular at contact transitions where rows of $\mathbf{J}_c$ become linearly dependent (to machine precision). Instead, we obtain stable results by adding a top-priority objective $E(\mathbf{x}) = ||C(\mathbf{x})||^2$. Since this objective has higher priority than all other objectives, it will always be minimized to zero, ensuring that the equality constraints are always satisfied.

If the prioritized optimization problem is well-posed, then line 16 will never be reached. For example, if $\mathbf{A}_N$ is full-rank, then $\bar{\mathbf{A}}_N$ must also be full-rank. In our controllers, this is ensured by the rest-pose task E_{pose}.

We have tested our algorithm with several QP solvers, including MOSEK's (www.mosek.com) interior-point method and the active-set solver in QPC (sigpromu.org/quadprog). We find that MOSEK works for all problems, but is somewhat slower that QPC. QPC is faster, but may not satisfy all physical constraints, leading to slightly sticky ground contact.

8.3 Weighted Objectives

We combine objectives at the same priority level according to (16). These objectives $(\mathbf{A}_j, \mathbf{b}_j)$ can be combined into a single quadratic as:

$$\mathbf{A} = \begin{bmatrix} \alpha_1\mathbf{A}_1 \\ \vdots \\ \alpha_J\mathbf{A}_J \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \alpha_1\mathbf{b}_1 \\ \vdots \\ \alpha_J\mathbf{b}_J \end{bmatrix}. \tag{37}$$

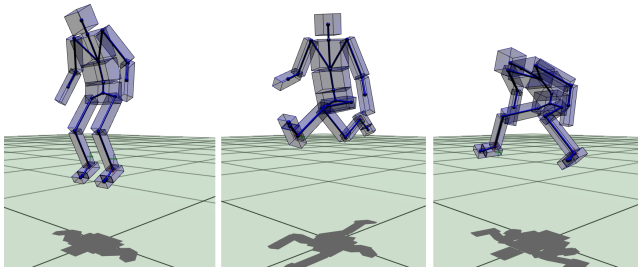

Figure 5: ***Sample jumping styles (left to right):*** *Normal, Scissor-Kicks, Air-Splits*

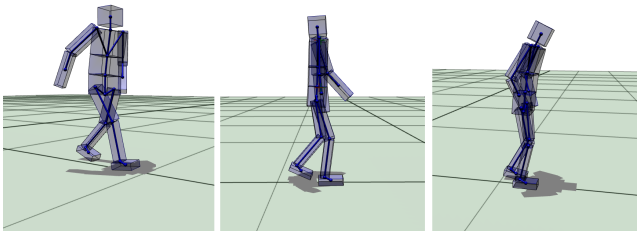

Figure 6: ***Sample walking styles (left to right):*** *Fast, slow, "sad"*

9 Results

We now describe results of applying feature-based controller design to balancing, standing jump, and walking motions, seen in Figures 5 and 6 and the accompanying video.

Simulation Details. Our simulator uses Featherstone's algorithm, and provides automatic computation of Jacobians and equations of motion (EOM) [Featherstone 2008]. Equations are integrated using the semi-implicit scheme of Guendelman et al. [2003]. Ground contact is modelled, using an inelastic impulse-based model, that includes friction ($\mu = 1$). Our double-precision single-threaded implementation runs at $50 - 100\%$ real-time, on a Dual Core 3GHz Intel Xeon CPU with 4GB RAM, running OSX 10.5, for all presented examples. Speed varies depending on character complexity, number of active contacts, number of prioritization levels, and the choice of QP solver. Our default character model has 35 degrees of freedom, with height and weight corresponding to a 50th percentile North American male. Skeletal dimensions and link mass are taken from Winter [2004]. Link inertias are calculated using uniform density shapes scaled to match skeletal dimensions.

Once optimal torques are computed, the joint accelerations $\ddot{\mathbf{q}}$ can be computed in two ways. First, one may directly apply forward dynamics using the torques. Alternatively, because the optimizer outputs $\ddot{\mathbf{q}}$ values as well as torques (both are included in $\mathbf{x}$), one may instead directly use the output of the optimizer. Because complementarity is not enforced during optimization, using these $\ddot{\mathbf{q}}$'s may lead to slight numerical inaccuracies. Furthermore, one must use a QP solver that guarantees that all constraints will be satisfied: interior-point methods do satisfy all constraints, whereas active set methods may not, leading to small sticky (bilateral) forces. However, using the optimizer output allows dynamics to be advanced at significantly larger timesteps. Specifically, using forward dynamics integration and control must be done at 1kHz to maintain stability; the simulation can operate at 100Hz using $\ddot{\mathbf{q}}$ from (1). In our tests, we developed the controllers using $\ddot{\mathbf{q}}$ values from the optimizer, and then perform the final simulations using forward dynamics.

Balance & Jumping. Starting from the balance controller, it was straightforward to produce jumps of different heights, for characters with varying skeletal properties, requiring only minor changes to parameters controlling hopping height. Despite large changes in motion, the controller produced consistent landings. This robustness is atypical of joint-space parameterization. The examples we demonstrate are qualitatively similar to those of Wooten [1998], who explored a vast array of acrobatic manoeuvres for simulated humans.

Starting from the basic jump controller, it took about 30 minutes to make the character perform both twisting jumps and in-air acrobatics. The feature-based parameterization also provides a useful dimensionality reduction. Only 3 parameters (i.e., crouch depth, thrust phase duration, and reference angular momentum) vary between standing and twisting jumps. Designing similar motions using conventional methods would have been very difficult, requiring careful trajectory design to avoid unwanted rotations prior to flight. Injecting AM in a coordinated manner would also be very fragile and time consuming to design.

Walking. Using the feature-based representation, we generate walking in a variety of styles. This can be done by reasoning about the desired look and feel of the motion. For example, to generate the sad walk example shown in our video, we begin with the slow walk, activate a setpoint objective to place the character's hands in his "pockets," tilt the head forward, add a little bit of forward lean to the body, and add a small amount of knee bend to the rest pose. This can be done without concern for coupling between different features. For example, since we are explicitly controlling the COM in our walking controller, trunk lean angle does not directly influence forward speed. This allows lean angle to be varied over a large range (Table 2), without worrying about its effect on stability. This is not the case with many joint-space walkers, which tightly couple lean and forward velocity (e.g., [Yin et al. 2007]). Furthermore, our method allows interactive changes to gait parameters, by directly adjusting control parameters such as those in Table 2.

Changes to Body Shape. As with previous model-based approaches [Abe et al. 2007; Macchietto et al. 2009], our controllers are robust to large changes to skeletal and inertial properties. We demonstrate this for both balancing and walking by changing morphology for our biped during simulation (Figure 1). No control parameters are changed for these experiments. Limb dimensions are increased by a factor of 2, with mass and inertia scaled appropriately.

Furthermore, we can apply the same controllers to characters with vastly different topologies and mass distributions (Figure 1). Aside from rest pose, all of our features are independent of character topology. We parameterize all of our characters in such a way that 0 can be used for the rest pose for all joints. Our implementation automatically computes all necessary features, Jacobians, and objective terms without requiring any additional effort. We find controllers can be successfully applied onto new characters *without any manual modification to the existing controller whatsoever*. This illustrates the power of the feature-based representation as an abstraction for reasoning about and creating motion.

Benefits of Prioritization. In our experiments, we have found prioritized optimization to have several advantages over weighted multiobjective combination. Although weighted optimization is sufficient for control with a small set of objectives [Abe et al. 2007; Jain et al. 2009; Mordatch et al. 2010], it can behave poorly when

many objectives are enabled/disabled at runtime. For example, despite only minor differences between the balance and jumping controllers, we were not able to generate jumping motions using weighted optimization. Results were numerically sensitive, producing infeasible problems, despite activating only one additional objective (e.g., E_{ankle}). As controllers become more complex and use a larger set of objectives, tuning objective weights becomes more difficult and time-consuming. From a design perspective, using prioritization allows control to be layered, improving workflow. Control of fundamental motion features can be "locked-in", without worrying about impact of lower-priority objectives on stability or motion execution.

In our experiments with in-place standing motions, we found prioritized optimization to be more robust to external disturbances. Because weighted optimization does not explicitly decouple objectives, disturbances affect all objectives. As drift increases in one objective, weighted optimization will begin to favor other objectives (e.g., error in E_{com} will increase, while E_{pose} decreases), eventually leading to task failure. Prioritized optimization does not suffer from this failure mode. Instead, tracking of low-priority features is sacrificed to minimize errors in higher priority goals. This yields human-like disturbance rejection, consistent with hypotheses from motor neuroscience [Todorov and Jordan 2002]. See the project web page for a comparison of weighted/prioritized optimization.

We also converted our walking controller to a weighted optimization. However, minimizing interference between the different objectives was extremely difficult and sensitive to parameter tuning. Stiff-looking walking can be obtained by disabling all objectives except for $E_{contact}$, $E_{com\|}$, E_{swing} and E_{pose}. Stylistic terms such as E_{AM} and E_{arms} proved especially sensitive and required careful tuning. Small changes in these objectives produced motion with "flailing" arms or excessive torso-counter rotation that "fought" rest-pose objectives. Coupling between user control parameters also increased. Because of these issues, we do not believe we would not have been able to develop a good set of objectives in the first place without using prioritized optimization.

Prioritized optimization has a few disadvantages as compared to weighted optimization. Prioritized optimization is slower and more work to implement, since each priority level requires solving both a QP and a SVD. For locomotion control, prioritized optimization requires the use of double-precision arithmetic for numerical stability, whereas single-precision arithmetic is sufficient for weighted optimization, as used in [Mordatch et al. 2010]. In our attempts to use prioritized optimization with single-precision arithmetic we found that Jacobians for high priority objectives (e.g., $E_{contact}$), can become rank deficient (to machine precision). This makes prioritized optimization sensitive to preconditioning techniques used by the QP solver. Weighted multiobjective combination does not have these problems, since adding many low-rank matrices together can produce full-rank ones. When the problem is formulated without unilateral constraints these issues can be avoided [de Lasa and Hertzmann 2009].

10 Discussion

We have presented a set of techniques for designing controllers for complex physics-based characters based on high-level motion features. By exploiting prioritization, our method reduces objective weight tuning, increases robustness, and allows controllers to be designed incrementally, while decreasing coupling between control objectives. To illustrate our method, we design controllers for biped locomotion behaviors. Although our controllers only use a small number of features, many natural human behavior properties arise, including human-like arm movement and hip-swaying in walking. These motion features cannot easily be generated using previous joint-space approaches.

We have found it challenging to generate certain types of motion with the described set of objectives/priorities. For example, we have not yet been able to design motions requiring large inertial changes in the sagittal plane, such as cartwheels and flips. The simple foot placement strategy we propose for walking has some limitations. Runtime updates to hip shift, swing duration, and step length must be coordinated to ensure the COM moves sufficiently close to the stance foot, to allow the stance leg to support the body. As swing duration decreases motion becomes more stable since the walker spends less time in single-support. Furthermore, because these curves are defined in world coordinates, the controller is not robust to perturbations. In follow-up work, we propose a planning strategy that addresses these issues [Mordatch et al. 2010].

We believe that the feature-based representation provides a powerful way to reason about the fundamentals of locomotion and to build new controllers. Our approach provides a vocabulary for expressing motion in terms of features, objectives, and priorities, while abstracting out the specifics of individual joints or masses. This provides an abstraction layer for control design that could be used for many types of motion, not just locomotion. For example, imagine designing a controller for throwing a ball. Target objectives might accelerate the hand forward and servo the wrist, while balance objectives regulate posture. With a more forceful throw, the body might automatically pivot and balance as a result of the combination of arm and AM objectives. A baseball pitch, could employ a more sophisticated sequence of target objectives for wind-up and release.

Acknowledgements

The authors thank David J. Fleet and the reviewers for valuable feedback. This research is supported in part by NSERC, CIFAR, CFI, and Ontario MRI. Part of this work was done while AH was on a sabbatical visit at Pixar Animation Studios.

References

ABE, Y., AND POPOVIĆ, J. 2006. Interactive Animation of Dynamic Manipulation. In *Proc. SCA*, 195–204.

ABE, Y., DA SILVA, M., AND POPOVIĆ, J. 2007. Multiobjective Control with Frictional Contacts. In *Proc. SCA*, 249–258.

ADAMCZYK, P. G., COLLINS, S. H., AND KUO, A. D. 2006. The advantages of a rolling foot in human walking. *J. Experimental Biology 209*, 20, 3953–3963.

AZEVEDO, C., POIGNET, P., AND ESPIAU, B. 2002. Moving horizon control for biped robots without reference trajectory. In *Int. Conf. Robotics and Automation*, 2762–2767.

BAERLOCHER, P., AND BOULIC, R. 2004. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer 20*, 6, 402–417.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. SIGGRAPH*, 23–34.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive Simulation of Stylized Human Locomotion. *ACM Trans. Graphics 27*, 3, 82.

DA SILVA, M., YEUHI, A., AND POPOVIĆ, J. 2008. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. *Computer Graphics Forum 27*, 2, 371–380.

DE LASA, M., AND HERTZMANN, A. 2009. Prioritized Optimization for Task-Space Control. In *Proc. IROS*.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable Controllers for Physics-Based Character Animation. In *Proc. SIGGRAPH*, 251–260.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. Graphics*, 417–426.

FEATHERSTONE, R. 2008. *Rigid Body Dynamics Algorithms*. Springer-Verlag.

FUJIMOTO, Y., OBATA, S., AND KAWAMURA, A. 1998. Robust biped walking with active interaction control between foot and ground. In *Int. Conf. Robotics and Automation*, 2030–2035.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex Rigid Bodies with Stacking. *ACM Trans. Graphics 22*, 3, 871–878.

HERR, H., AND POPOVIC, M. 2008. Angular momentum in human walking. *J. Experimental Biology 211*, 467–481.

HODGINS, J. K., AND POLLARD, N. S. 1997. Adapting Simulated Behaviors for New Characters. In *Proc. SIGGRAPH*, 153–162.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proc. SIGGRAPH*, 71–78.

HSU, P., MAUSER, J., AND SASTRY, S. 1989. Dynamic control of redundant manipulators. *J. Robotic Systems 6*, 2, 133–148.

JAIN, S., YE, Y., AND LIU, C. K. 2009. Optimization-Based Interactive Motion Synthesis. *ACM Trans. Graphics 28*, 1, 1–10.

KANOUN, O., LAMIRAUX, F., WIEBER, P.-B., KANEHIRO, F., YOSHIDA, E., AND LAUMOND, J.-P. 2009. Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *Int. Conf. Robotics and Automation*, 724–729.

KHATIB, O. 1987. A Unified Approach to Motion and Force Control of Robot Manipulators: The Operational Space Formulation. *J. Robotics and Automation 3*, 1, 43–53.

KUDOH, S., KOMURA, T., AND IKEUCHI, K. 2006. Stepping Motion for a Human-like Character to Maintain Balance against Large Perturbations. In *Int. Conf. Robotics and Automation*, 2661–2666.

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proc. SIGGRAPH 1996*, 155–162.

LIEGEOIS, A. 1977. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *Trans. on Systems, Man and Cybernetics 7*, 12, 868–871.

MACCHIETTO, A., ZORDAN, V., AND SHELTON, C. 2009. Momentum Control for Balance. *ACM Trans. Graphics 28*, 3, 80.

MANSARD, N., AND KHATIB, O. 2008. Continuous control law from unilateral constraints. In *Int. Conf. Robotics and Automation*, 3359–3364.

MARLER, R. T., AND ARORA, J. S. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization 26*, 6, 369–395.

MORDATCH, I., DE LASA, M., AND HERTZMANN, A. 2010. Robust Physics-Based Locomotion Using Low-Dimensional Planning. *ACM Trans. Graphics 29*, 3.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware Nonlinear Control of Dynamic Characters. *ACM Trans. Graphics 28*, 3, 81.

NAKAMURA, Y., HANAFUSA, H., AND YOSHIKAWA, T. 1987. Task-Priority Based Redundancy Control of Robot Manipulators. *Int. J. Robotics Research 6*, 2.

NAKANISHI, J., CORY, R., MISTRY, M., PETERS, J., AND SCHAAL, S. 2008. Operational Space Control: A Theoretical and Empirical Comparison. *Int. J. Robotics Research 27*, 6.

ORIN, D., AND GOSWAMI, A. 2008. Centroidal Momentum Matrix of a Humanoid Robot: Structure and Propreties. In *Int. Conf. on Robotics and Intelligent Systems*.

POPOVIC, M., HOFMANN, A., AND HERR, H. 2004. Angular Momentum Regulation during Human Walking: Biomechanics and Control. In *Int. Conf. Robotics and Automation*.

POZZO, T., BERTHOZ, A., AND LEFORT, L. 1990. Head stabilization during various locomotor tasks in humans. *Exp. Brain Res. 82*, 97–106.

PRATT, J., CHEW, C.-M., TORRES, A., DILWORTH, P., AND PRATT, G. 2001. Virtual Model Control: An intuitive approach for bipedal locomotion. *Int. J. Robotics Research*.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. *SIGGRAPH Comput. Graph. 25*, 4, 349–358.

SENTIS, L. 2007. *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Stanford.

SHARON, D., AND VAN DE PANNE, M. 2005. Synthesis of Controllers for Stylized Planar Bipedal Walking. In *Int. Conf. Robotics and Automation*, 2387–2392.

SHKOLNIK, A., AND TEDRAKE, R. 2008. High-Dimensional Underactuated Motion Planning via Task Space Control. *Int. Conf. on Robotics and Intelligent Systems*, 3762–3768.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating Biped Behaviors from Human Motion Data. *ACM Trans. Graphics 26*, 3, 107.

TODOROV, E., AND JORDAN, M. I. 2002. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience 5*, 11, 1226–1235.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing Walking Controllers. *ACM Trans. Graphics 28*, 5, 168.

WINTER, D. A. 2004. *Biomechanics and Motor Control of Human Movement*, 3rd ed. Wiley.

WOOTEN, W. 1998. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Trans. Graphics 26*, 3, 81.

ZORDAN, V., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *Proc. SCA*, 89–96.