Terrain-Adaptive Bipedal Locomotion Control

Jia-chi Wu Zoran Popović

University of Washington

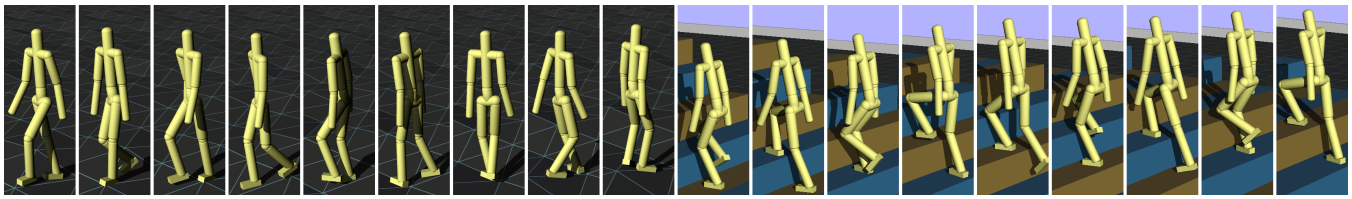

Figure 1: *A biped (left) performs a 180° turn and then walks backwards on uneven terrain and (right) climbs up stairs.*

Abstract

We describe a framework for the automatic synthesis of biped locomotion controllers that adapt to uneven terrain at run-time. The framework consists of two components: a per-footstep end-effector path planner and a per-timestep generalized-force solver. At the start of each footstep, the planner performs short-term planning in the space of end-effector trajectories. These trajectories adapt to the interactive task goals and the features of the surrounding uneven terrain at run-time. We solve for the parameters of the planner for different tasks in offline optimizations. Using the per-footstep plan, the generalized-force solver takes ground contacts into consideration and solves a quadratic program at each simulation timestep to obtain joint torques that drive the biped. We demonstrate the capabilities of the controllers in complex navigation tasks where they perform gradual or sharp turns and transition between moving forwards, backwards, and sideways on uneven terrain (including hurdles and stairs) according to the interactive task goals. We also show that the resulting controllers are capable of handling morphology changes to the character.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1 Introduction

Passive physical simulations in virtual environments such as video games improve the interactivity and realism of simulated objects and simplify the production by removing the need for scripted or data-driven animation contents. Effective physically based locomotion controllers for characters could extend these advantages to active dynamic characters. To be effective, dynamic character controllers should possess a number of characteristics. First, they need to be able to navigate on uneven terrain: terrain in interesting virtual environments is rarely completely flat all the time. Second, the controllers need to be able to perform navigation by changing directions with agility similar to humans (e.g., completing a 180° turn in two steps): quick direction changes are challenging because motion patterns for turning are more complex than forward walk, and increase in agility often compromises stability. Third, the controllers should take optimality principles such as minimum torque into accounts when generating motion: in addition to being physically correct, the motion patterns also need to be similar to those of real animals.

Recent research has made significant progress towards each of these goals separately, either through robust manually constructed and tuned controllers [Yin et al. 2007], or by using robust trajectory-data following controllers to achieve greater realism [Muico et al. 2009]. In this paper we focus on a mechanism that automatically creates controllers from scratch without manual parameter tuning or use of motion data, while at the same time trying to approach the key objectives of natural biped controllers applicable to interactive environments and video games. We focus on a control method that is invariant to the simulation process so that it can be easily combined with existing physics engines.

We use a two-layer control hierarchy where the higher level plans the optimal paths for a small set of key end effectors. These trajectories consider specific properties of the terrain, leading to locomotion that appears efficiently aware of the environment. Other parameters of the trajectory relevant to efficiency and longer horizon stability are automatically determined in the offline optimization process. The lower-level control solves a quadratic program (QP) to determine real-time actuations that follow trajectories.

The key contribution of this work is a framework for automatic synthesis of biped controllers capable of performing various locomotion skills including walking sideways and backwards on uneven terrain; the controllers also exhibit fast, responsive turning abilities that approach those seen in natural systems. Planning for key trajectories and a separate compliant control that follows the trajectories is a particularly flexible control structure that is possibly not far off from the strategies that natural systems use [Abend et al. 1982].

Our examples show that terrain-aware control eliminates the need of using unnaturally high clearance during the swing phase on uneven terrain, a typical strategy that terrain-blind controllers use. We also show that a two-level controller structure is capable of controlling certain morphologically different bipeds, such as one with inverted knees, without any changes to the controller.

2 Related Work

Offline methods can be used to synthesize animation of bipedal locomotion [Rose et al. 1996; Safonova et al. 2004; Liu et al. 2005].

These offline methods use optimization and the optimality principles to generate natural motion. However, they cannot be used directly in a dynamically simulated interactive environments, since they do not synthesize controllers, only trajectories. A procedural kinematic gait generator can be more efficient in generating different gaits [Sun and Metaxas 2001]; however, its kinematic nature still limits the degree of interaction between the character and the environment. Dynamical controllers exhibit more physically realistic behaviors because they interact with the environment and use delicate strategies similar to natural systems to maintain balance. These controllers can be manually constructed [Raibert and Hodgins 1991; Stewart and Cremer 1992; Hodgins et al. 1995] or be further fine-tuned using automatic methods [van de Panne et al. 1992; Laszlo et al. 1996; Yin et al. 2007]. High-fidelity realism of natural locomotion is more difficult to achieve with manual construction. Manually constructing controllers for agile and responsive locomotion skills such as walking sideways with crossing legs or making 180° turns can also be difficult because of the more complex motion patterns necessary for these maneuvers. Motion capture data can be used as initial references for these more complex controllers [Muico et al. 2009].

A common feature of many existing locomotion controllers is the use of reference trajectories for all joint angles. Such controllers follow the references either by using PD controllers at all joints, solving quadratic programs for the tracking torques [da Silva et al. 2008], or by building value functions around the trajectories and deriving control signals from the value functions [Atkeson and Morimoto 2002]. Adapting to uneven terrain is a unique challenge for controllers relying on these references because it requires nontrivial changes to the trajectories for all controllers [Yin et al. 2008]. If motion capture data are used as the source references, it may be necessary to collect motion data on terrain with many different elevation changes for each type of motion.

An end-effector oriented control structure (in contrast to direct joint control) has long been postulated as a key aspect of control in natural systems [Bernstein 1967]. Recently, neuroscientists have shown indications that humans use end-effector control for tasks such as arm reaching [Todorov 2004]. Combining end-effector control and the optimality principles, researchers have computationally reproduced the arm-reaching motion [Bullock and Grossberg 1988]. Control of robot manipulators benefits from describing the dynamics in terms of the end effectors [Khatib 1987], and a related technique has been used to create simple biped robots [Pratt et al. 1997]. In this work, we consider locomotion as a goal-directed task that involves three end effectors: the upper body moves at a roughly constant speed, and the two feet alternate to advance in the movement direction. We show that this simple high-level structure can be used to successfully synthesize agile bipedal locomotion control.

Creating fully automated agents requires a long-horizon planner that plans ahead for many footsteps [Coros et al. 2008]. Static stability of the biped is often used as one of the objectives instead of dynamic stability to make the planning problem feasible [Kuffner et al. 2001; Zhang et al. 2009]. Even though motion primitives can be used to improve the quality of the motion [Hauser et al. 2008], the statically stable motion invariably appears less natural. In this work we focus on a short-horizon (one footstep) planner that acquires dynamic stability via optimization. The long-horizon planning is deferred to the users who interactively command the biped to perform different locomotion skills.

3 Overview

At run-time, our system takes as input the user-specified task goal G at each footstep and in real time outputs the joint torques $\boldsymbol{\tau}_\theta$ that move the biped character in the virtual environment at each simulated timestep. We compute $\boldsymbol{\tau}_\theta$ using a two-layer control hierarchy.

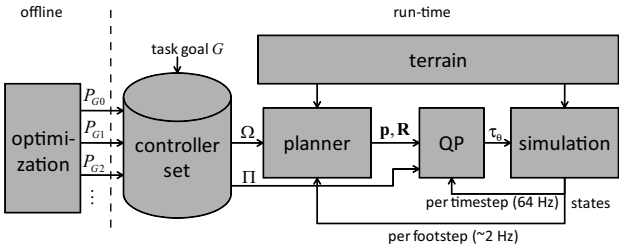

Figure 2: *System overview.*

At the higher level, the end-effector path planner plans the position trajectories $\mathbf{p}(t)$ and orientation trajectories $\mathbf{R}(t)$ for the end effectors for half a locomotion cycle (roughly a footstep). Planning occurs at the start of each half-cycle. We define the end-effector path planner with a compact set Ω of parameters, determined automatically during the preprocessing stage. At run-time, the planner takes terrain information into account and adapts the trajectories for foot-ground collision avoidance and to elevation changes.

The lower-level control follows $\mathbf{p}(t)$ and $\mathbf{R}(t)$ by using the outputs from a set of end-effector PD controllers as a reference. At each timestep, lower-level control solves a QP for the joint torques $\boldsymbol{\tau}_\theta$. The PD controllers are used to ensure compliant behavior. No PD controllers are actually simulated: the joint torques are a product of QP optimization. We include the automatically determined PD controller coefficients in the lower-level control parameter set Π. We devise a simple way of modifying the QP to handle the changed dynamics due to ground contacts without the need for dynamically changing PD controller coefficients.

During the preprocessing stage, the offline optimization tunes controller parameters P (which include both Ω and Π) for each task G by evaluating a cost function using simulations. During interaction, the appropriate values of P are then chosen from the preprocessed results at the start of each half-cycle based on the user-specified task goal G. The system components and their interaction are shown in Figure 2. Note that the run-time portion of the system is also used during offline optimization to evaluate the cost function.

4 End-Effector Path Planning

The path planner plans the paths of the end effectors ahead for a half-cycle whose duration t_h is predetermined. At the beginning of a half-cycle, the path planner takes as input (1) the current positions and orientations of the end-effector links and (2) the task goal $G = (\psi_f, \psi_m, r)$, where ψ_f is the desired facing angle at the end of the half-cycle, ψ_m is the desired movement direction angle, and r is the desired step length. The preset duration t_h and step length r together determine the desired walking speed. The path planner then outputs the desired linear positions $\mathbf{p}(t)$ and orientations $\mathbf{R}(t)$ of the end effectors as functions of the elapsed half-cycle time t.

We consider the upper body and the two feet as the end effectors. Upper limbs can be added with the hands or the forearms as additional end effectors as shown in our results, but we shall omit them from most of our discussion for brevity.

Constructing $\mathbf{p}(t)$ and $\mathbf{R}(t)$ for each end effector follows the process of (1) determining the target (linear or angular) position and (2) constructing a path that connects the starting and target positions. Both steps use parameters in Ω. The optimized values of Ω are chosen from the offline optimization results given the run-time

task goal G. We describe the construction process of $\mathbf{p}(t)$ and $\mathbf{R}(t)$ in §A.

Below, we describe two key aspects which enable the controllers to adapt to and navigate on uneven terrain. End-effector planning makes it easy to implement these two strategies: the planner simply adds offsets to the end effectors' paths or target positions.

Terrain Awareness and Adaptation At run-time, the planner samples the terrain heights along the swing-foot path and adds height offsets to the path to avoid unwanted collisions. The awareness of terrain is a key distinction of our method from the terrain-blind approach in which individual controllers do not adapt to terrain variations. The awareness eliminates the need for (fixed) swing foot paths with high clearances on uneven terrain.

Per-Footstep Balancing At run-time, the planner also modifies the horizontal target positions of the swing foot and upper body based on (piecewise) linear feedback of the upper-body positional deviation at the end of the previous half-cycle. Effectively, if the previous half-cycle control leads the biped away from the expected upper-body state, the planner adjusts the swing-foot and upper-body targets to compensate in the subsequent half-cycle. The feedback and bias coefficients for balancing are part of the planner parameters Ω and are automatically tuned by the optimization.

5 Frames and Frame Tracking

Once we have the desired paths from the planner, we need to compute the joint torques to execute the plan. To facilitate the joint-torque computation, we attach coordinate frames and ideal PD controllers to the end-effector links.

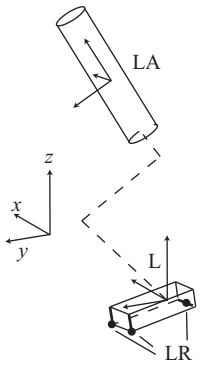

Figure 3: *Frames and their types.*

It is straightforward to compute the desired linear or angular positions and velocities of the frames from the desired paths $\mathbf{p}(t)$ and $\mathbf{R}(t)$ of the links given by the planner. Each frame has a corresponding ideal PD controller that tracks the desired frame position or orientation. We call them *ideal* PD controllers because we do not apply their outputs directly to the underactuated biped. Instead, we treat their outputs as ideal forces and torques we would like to reproduce using joint torques in a physically valid manner.

There are three types of PD controllers and frames: *linear-angular* (LA), *linear* (L), and *linear relative* (LR) (Figure 3).

An LA PD controller tracks both the linear and angular positions and velocities of an LA frame. Given the desired linear and angular positions and velocities ($\mathbf{p}$, $\dot{\mathbf{p}}$, $\mathbf{R}$, and $\boldsymbol{\omega}$) and the current positions and velocities ($\hat{\mathbf{p}}$, $\dot{\hat{\mathbf{p}}}$, etc.) of frame i, we can compute its ideal force $\mathbf{f}_i$ and torque $\boldsymbol{\tau}_i$ as

$$\mathbf{f}_i = k_p \left(\mathbf{p} - \hat{\mathbf{p}}\right) + b_p \left(\dot{\mathbf{p}} - \dot{\hat{\mathbf{p}}}\right) + c_p \mathbf{z} \tag{1}$$

$$\boldsymbol{\tau}_i = \sum_{j=0}^{2} \hat{\mathbf{e}}_j \times \left[k_o \left(\mathbf{e}_j - \hat{\mathbf{e}}_j\right) + b_o \left(\dot{\mathbf{e}}_j - \dot{\hat{\mathbf{e}}}_j\right)\right] \tag{2}$$

where c_p is the bias coefficient added so that the feedback terms do not have to compensate for gravity, and $\mathbf{e}_0$ to $\mathbf{e}_2$ are unit vectors aligned with the (local) x-, y-, and z-axes of the frame.

An L PD controller tracks only the linear position and velocity of the L frame, and we use Equation 1 to compute its ideal force.

An LR PD controller tracks the linear position and velocity relative to the center of the link to which the LR frame is attached. The ideal force is

$$\mathbf{f}_i = k_r \left(\mathbf{r} - \hat{\mathbf{r}}\right) + b_r \left(\dot{\mathbf{r}} - \dot{\hat{\mathbf{r}}}\right) \tag{3}$$

where $\mathbf{r} = \mathbf{p} - \mathbf{p}_0$ is the relative position of the frame with respect to the center $\mathbf{p}_0$ of the link.

We attach an LA frame to the upper body, an L frame to each foot at the ankle, and three LR frames to each foot at the toes and the heel. The L frame at the ankle is primarily for moving the foot. The LR frames at the toes and the heel orient the foot during the swing phase. As we shall see later, the frames on the feet can serve as supports when the geometries they are attached to are in contact with the ground.

The PD controller coefficients (k, b, and c) are included in the lower-level parameter set Π and tuned by the offline optimization.

6 Solving for Joint Torques

In order to compute internal forces and torques, we pair up the frames between which we wish to apply forces. A pair (A_i, A_j) of frames consists of a *reaction* frame A_i and an *action* frame A_j. (We follow the use of terminologies in Pratt and colleagues' work [1997].) For the biped, there is $M = 1$ reaction frame A_0 which is attached to the upper body. All the $N = 8$ frames on the feet are action frames, denoted as $A_1, \ldots, A_8$. We pair up the upper-body reaction frame with each of the 8 action frames on the feet. We use S to denote the set of all $K = 8$ frame pairs.

6.1 Generalized States and Forces

In this subsection we briefly define the generalized state and force vectors we shall use in our discussion and their relationship to the joint torques. Detailed derivations can be found in standard text, e.g., Craig [1989].

The generalized state vector is

$$\mathbf{x} = \left(\mathbf{p}_{i_0 j_0}^\top, \ldots, \mathbf{p}_{i_{K-1} j_{K-1}}^\top\right)^\top$$

Each 3-vector $\mathbf{p}_{ij}$ is the relative position of action frame A_j to the position of reaction frame A_i in the global frame:

$$\mathbf{p}_{ij} = \mathbf{p}_j \left(\boldsymbol{\theta}, \mathbf{p}_{\text{root}}, \boldsymbol{\theta}_{\text{root}}\right) - \mathbf{p}_i \left(\boldsymbol{\theta}, \mathbf{p}_{\text{root}}, \boldsymbol{\theta}_{\text{root}}\right) \quad (A_i, A_j) \in S$$

where $\boldsymbol{\theta}$ is the joint angles, and $\mathbf{p}_{\text{root}}$ and $\boldsymbol{\theta}_{\text{root}}$ are the global position and orientation of the root link. (We use the upper body as the root.)

The generalized force vector is

$$\mathbf{f} = \left(\mathbf{f}_{i_0 j_0}^\top, \ldots, \mathbf{f}_{i_{K-1} j_{K-1}}^\top\right)^\top$$

Each 3-vector $\mathbf{f}_{ij}$ is the force applied at the origin of action frame A_j from reaction frame A_i; for each $\mathbf{f}_{ij}$, an opposite force $-\mathbf{f}_{ij}$ is applied to the reaction frame A_i at the same location.

The Jacobian $\mathbf{J} = \partial \mathbf{x} / \partial \boldsymbol{\theta}$ is useful for computing the equivalent joint torques $\boldsymbol{\tau}_\theta$ given the generalized forces: $\boldsymbol{\tau}_\theta = \mathbf{J}^\top \mathbf{f}$. We do not compute the Jacobian with respect to $\mathbf{p}_{\text{root}}$ or $\boldsymbol{\theta}_{\text{root}}$ because they are unactuated.

6.2 Quadratic Program Formulation

Given the ideal PD controller forces $\mathbf{f}_i$ and $\mathbf{f}_j$, and torques $\boldsymbol{\tau}_i$, we would like the generalized forces to exactly reproduce them at each frame:

$$\sum_{j,(A_i,A_j)\in S} -\mathbf{f}_{ij} = \mathbf{f}_i \quad i = 0,\ldots,M-1$$

$$\sum_{i,(A_i,A_j)\in S} \mathbf{f}_{ij} = \mathbf{f}_j \quad j = M,\ldots,M+N-1$$

$$\sum_{j,(A_i,A_j)\in S} (\mathbf{p}_j - \mathbf{p}_i) \times -\mathbf{f}_{ij} = \boldsymbol{\tau}_i \quad i = 0,\ldots,M-1$$

or in matrix form

$$\mathbf{A}\mathbf{f} = \mathbf{b} \tag{4}$$

where $\mathbf{b}^\top = \left(\mathbf{f}_0^\top,\ldots,\mathbf{f}_{M+N-1}^\top,\boldsymbol{\tau}_0^\top,\ldots\boldsymbol{\tau}_{M-1}^\top\right)$. The equality (Equation 4) is in general infeasible for an underactuated system. We solve a quadratic problem instead for the physically valid generalized forces that minimize the ideal force and torque errors:

$$\min_{\mathbf{f}} \quad \|\mathbf{A}\mathbf{f}-\mathbf{b}\|_{\mathbf{W}}^2 \tag{5a}$$

$$\text{subject to} \quad -\mathbf{f}_{max} \le \mathbf{f} \le \mathbf{f}_{max} \tag{5b}$$

$$-\boldsymbol{\tau}_{max} \le \mathbf{J}^\top\mathbf{f} \le \boldsymbol{\tau}_{max} \tag{5c}$$

$$\mathbf{Z}\mathbf{f} = 0 \tag{5d}$$

$\mathbf{Z}$ is a matrix whose columns are the basis for the null space of $\mathbf{J}^\top$, and $\mathbf{f}_{max}$ ($= 2000$) and $\boldsymbol{\tau}_{max}$ are bounds of the generalized forces and joint torques. The null space constraint $\mathbf{Z}\mathbf{f} = 0$ makes sure that $\mathbf{f}$ lies in the row space of $\mathbf{J}^\top$ and that the joint torques can exert the generalized forces as desired. We include $\mathbf{W}$ in Π so that the relative weights are automatically balanced by the offline optimization. Diagonal entries in $\mathbf{W}$ corresponding to weights for the upper-body ideal forces and torques are set to a common value chosen by the optimization; the weights for the feet are set to 1.

It is worth noting that our QP formulation uses the *end-effector force* and *torque* errors as the objective, while previous work has used *joint acceleration* errors as part of the objectives [Abe et al. 2007].

6.3 Ground Contacts

With the changing dynamics caused by ground contacts, the PD controllers with linear feedback are unlikely to produce suitable tracking forces for all different cases. The deficiency of the PD controllers suggests that we need to solve for the generalized forces differently when there are ground contacts.

We devise a naive scheme that modifies the QP to handle the changed dynamics. This scheme works well when combined with the offline optimization. When the geometry to which a *stationary* (as specified by the desired path) action frame is attached is in contact with the ground, we treat the frame as if it were pinned to the ground. Under this assumption, the ideal force $\mathbf{f}_j$ would have no effect on a "pinned" action frame A_j, and therefore we remove errors of $\mathbf{f}_j$ from the objective (Equation 5a) by removing the corresponding rows and elements in $\mathbf{A}$ and $\mathbf{b}$. Because the frames are never actually pinned to the ground, combinations of desired end-effector paths and PD controller coefficients that cause excessive movement of the "pinned" frames will automatically be penalized by the cost function the offline optimization uses. This scheme avoids the need to explicitly tie our controllers to the collision handling algorithm and the need to add complicated constraints to the QP.

The current and desired states of a stationary frame A_j must satisfy

$$\begin{aligned} \|\mathbf{p}_j - \hat{\mathbf{p}}_j\|_2 &< \varepsilon_p, \text{ and} \\ \|\dot{\mathbf{p}}_j\|_2 &< \varepsilon_{\dot{p}} \end{aligned}$$

where $\varepsilon_p = 0.2$ m and $\varepsilon_{\dot{p}} = 0.05$ m/s. On each foot we use two box geometries for contact, one for the toes and one for the heel, both of which are attached to the rigid foot link. The advantage of using multiple frames on the foot over using a single one is that when three or more frames on the feet are "pinned", they form an area of support for the upper body. The area may degenerate into a line or a point when there are fewer "pinned" frames.

The resulting problem is a simple convex QP that can be solved using various QP solvers; we use the MOSEK optimization software [Mosek ApS 2009]. Once we have the generalized forces $\mathbf{f}$, we can compute and apply the equivalent joint torques at the joints, plus the null-space control torques if necessary (§B).

7 Locomotion Controllers

We define a *cyclical* locomotion controller using a tuple P_c of unknown parameters; P_c contains two sets Ω_0 and Ω_1 of planner parameters (one for each half-cycle) and two sets Π_0 and Π_1 of lower-level control parameters:

$$P_c = (\Omega_0, \Omega_1, \Pi_0, \Pi_1)$$

A *transitioning* controller connects two cyclical controllers using one or more half-cycles depending on the length of the transition. To allow more natural transitions, we further include the task goals during the transition as unknowns. The parameters of a T-half-cycle transitioning controller are

$$P_t = (\Omega_0,\ldots,\Omega_{T-1},\Pi_0,\ldots,\Pi_{T-1},G_0,\ldots,G_{T-1})$$

During interaction, if a task goal G_{new} requires transition from the current cyclical controller, the proper transitioning controller will be used during the transition. After the transition, the new goal G_{new} will be used as the input to the new cyclical controller.

8 Offline Optimization

The offline optimization tunes P_c or P_t for each locomotion controller. We use the CMA evolution strategy as a black-box optimizer [Hansen 2006]. CMA has previously been used to successfully solve locomotion and controller problems [Wampler and Popović 2009; Wang et al. 2009]. CMA starts with a Gaussian prior distribution of the unknowns in the search space. At each iteration (generation), it generates a number λ of samples using this distribution and evaluates the performance of each sample according to the cost function. It then picks the top-performing samples (the elites) and uses their positions in the search space to update the distribution so that the new distribution is more likely to contain good samples. It repeats the process until convergence, i.e., when the distribution mean no longer changes significantly.

8.1 Cost Function

The cost function we use consists of (1) the frame-tracking errors, (2) the per-footstep center-of-mass (COM) deviation error, and (3) the energetic cost.

We compute the frame-tracking error $e_{track,i}$ of frame i using Equations 1–3 (depending on the frame type) by replacing the PD controller feedback coefficients with the square roots of the

weights and setting the bias coefficient, if any, to zero. Empirically, we have found that *not* penalizing deviations that are smaller than a tolerance ε_{track} makes the motion more natural, possibly because the tolerance encourages the controller to use passive dynamics near the planned paths. The tracking error is the sum of squares of the "forces" (and "torques", if it is an LA frame) we thus compute minus the tolerance. For example, let $\mathbf{f}_{track,w_{kp},w_{bp}}$ and $\boldsymbol{\tau}_{track,w_{ko},w_{bo}}$ denote the terms for an LA frame we compute by using the weights w_{kp}, w_{bp}, w_{ko}, and w_{bo} as replacement coefficients; the tracking error is

$$e_{track} = \left\{ \left\| \mathbf{f}_{track,w_{kp},w_{bp}} \right\|_2^2 - \varepsilon_{track} \right\}^+ + \left\{ \left\| \boldsymbol{\tau}_{track,w_{ko},w_{bo}} \right\|_2^2 - \varepsilon_{track} \right\}^+$$

where $\{x\}^+$ denotes $\max(x, 0)$. The tolerance ε_{track} is set to 2.5e-3.

The COM deviation error is the squared COM deviation, $\|\Delta \mathbf{c}_k\|_{\mathbf{W}_c}^2$, where $\Delta \mathbf{c}_k = \hat{\mathbf{c}}_k - \mathbf{c}_k$ denotes the deviation of COM horizontal position $\hat{\mathbf{c}}_k$ from the desired horizontal position $\mathbf{c}_k$ at the end of the k-th half-cycle. We compute the desired horizontal position as $\mathbf{c}_k = \hat{\mathbf{c}}_{k-1} + \mathbf{R}_z(\psi_m)(r\mathbf{y})$, where $\mathbf{R}_z(\psi)$ is the rotation matrix representing the rotation about the z-axis by ψ. We do not penalize COM deviations in the vertical direction.

The energetic cost is simply $\|\boldsymbol{\tau}_\theta\|_{\mathbf{W}_\tau}^2$. Let i denote the frame index, j the timestep index, L the number of timesteps, and k the half-cycle index. The total cost is

$$E(P) = \sum_{j=0}^{L-1} \sum_i e_{track,i} + \sum_k \|\Delta \mathbf{c}_k\|_{\mathbf{W}_c}^2 + \sum_j \|\boldsymbol{\tau}_\theta\|_{\mathbf{W}_\tau}^2 \quad (6)$$

where we have omitted the timestep indexes from the frame-tracking errors and the joint torques, and P can be either P_c or P_t.

8.2 Helper Force

To better the chance of finding good solutions, we use the *helper force* to guide the optimization. Similar helper forces have been used for locomotion control where the force is applied to the upper body [van de Panne and Lamouret 1995]. Here we define the helper force $\mathbf{h}$ for all frames to be the difference between the ideal PD controller outputs and the actual outputs the generalized forces produce: $\mathbf{h} = \mathbf{b} - \mathbf{A}\mathbf{f}$.

We introduce the upper bound h_{max} as an unknown parameter in the offline optimization. The actual helper force applied to the frames in the simulation during the offline optimization is $\text{clip}\left(-\{h_{max}\}^+, \{h_{max}\}^+, \mathbf{h}\right)$, where $\text{clip}(lb, ub, x) = \max(lb, \min(ub, x))$.

Relating the use of the helper force to the general *constrained optimization* problems, we convert the constraint ($h_{max} \leq 0$) into an exact nonsmooth penalty function $\{h_{max}\}^+$ and add the scaled penalty to the cost function. The controller optimization is therefore

$$\min_{P, h_{max}} E(P) + w_h L \{h_{max}\}^+ \quad (7)$$

The advantage of this exact nonsmooth penalty function over a smooth one such as the quadratic penalty $(\{h_{max}\}^+)^2$ is that the performance of the optimization depends less on the value of the penalty parameter w_h and how we update it [Nocedal and Wright 2006]. With the nonsmooth penalty function, the optimization may still find an infeasible solution (i.e., $h_{max} > 0$) if w_h is too small. We may use the standard *continuation* procedure and restart the optimization with a larger value for w_h after the optimization approximately converges to an infeasible solution. However, CMA is able to find feasible solutions for all our controllers with the initial weights, and therefore we do not have to resort to continuation.

8.3 Optimization Setup

For each cyclical controller, we accumulate the cost for 4 locomotion cycles. The weights of the tracking errors are $w_{kp} = w_{ko} = w_{kr} = 1$ and $w_{bp} = w_{bo} = w_{br} = 0$ for all frames. $\mathbf{W}_c = 100\mathbf{I}$ ($\mathbf{I}$ is the identity matrix), and $\mathbf{W}_\tau$ = (8e-2)$\mathbf{W}_{pref}$, where $\mathbf{W}_{pref}$ is a diagonal matrix whose diagonal elements are set to the squared inverses of the joint torque upper bounds to encourage use of stronger muscles. The penalty parameter w_h = 5e-3. We set t_h to 0.55 for all walk controllers; $r = 0.65$ for forward and backward walk controllers, $r = 0.5$ for side-stepping controllers, and $r = 0$ for the standing controller.

A different random height field is used in each CMA iteration, while samples in the same iteration use the same height field. The grid spacing of the height fields is 1 m × 1 m, and the grid point heights are uniformly distributed between 0 and 0.2 m. Terrain with vertical drops created with boxes is also used for the optimization of the forward walk controller. The box heights are uniformly distributed between 0 and 0.2 m.

For each transitioning controller, we accumulate the cost for six locomotion cycles, with the first three half-cycles using the first cyclical controller, followed by the transitioning controller, and then the second cyclical controller. All our transitioning controllers consist of two half-cycles. The cost-function weights are the same as those for cyclical controllers, but the COM deviation error term is disabled during the transition. We add additional per-half-cycle tracking errors starting from a half-cycle after the transition to encourage quick entry into cyclical motion. The average poses of the second cyclical controller at the start of each half-cycle are used as desired poses, and the weights for the additional tracking errors are $w_{kp} = w_{ko} = w_{kr} = 5$ and $w_{bp} = w_{bo} = w_{br}$ = 5e-2. The transitioning-controller optimization uses the same type of height fields that the cyclical-controller optimization uses.

We expect that in game environments the upper limbs may need to be controlled independently from the locomotion. We include a few different types of predefined upper-limb motion, such as natural swing and spreading out, in the optimization as "noise," similar to the way we include random terrain. The upper limbs are dynamically controlled with the forearms as end effectors, and the additional ideal PD controller coefficients are also solved for in the offline optimization.

The number of unknowns ranges from 46 to 70. The CMA population size λ is set to 32 for all controller optimizations. We list the bounds of the unknowns in Table 1. The value of h_{max} is bounded between -200 and 1000: the lower bound is set to a negative value so that CMA can more easily generate samples that do not use any helper force with the rejection sampling scheme. When passed to CMA, the bounds are normalized to $[0, 1]$. The normalized initial values are uniformly set to 0.5 and standard deviations uniformly 0.3. We terminate the CMA optimization when the change of the distribution mean drops below 1e-3, or when it reaches the 4000th iteration.

9 Results

We have found CMA to be robust to local minima, at least in terms of finding solutions that are feasible and have consistent resulting

motion: multiple trials of the forward walk controller optimization give visually identical results. The sufficiently large initial standard deviation allows CMA to properly sample the search space at the beginning, and therefore there is no need to manually tweak the initial values of the unknowns for each optimization.

We create five cyclical controllers for moving in four different directions and standing still, and we selectively create a number of transitioning controllers to connect the cyclical controllers (Table 2). With a cyclical controller, we can also use the task goal G for small ($\pm 15^\circ$) facing and movement direction changes. We use both cyclical and transitioning controllers in navigation tasks on uneven terrain (with the same specification used in the optimizations) and on slopes of ± 0.2 ($\pm 11.3^\circ$). The user interactively specifies G, and the system chooses appropriate cyclical or transitioning controllers to achieve the goals. Since each cyclical or transitioning controller is created automatically without manually tuning any controller parameters, and each controller adapts to terrain at run-time, the total set of controllers can be easily expanded. For example, we have created a single expert controller that climbs up stairs with rise heights between 0 and 0.4 m, and another that walks down. We have also created a controller that walks diagonally across random boxes.

A number of emergent patterns in motion arise from optimization. The side-stepping controller exhibits natural leg-crossing motion, and benign inter-leg collisions that alter non-end effectors' states are allowed to happen, just like in natural systems, because the controller is mainly concerned about the task-related end-effector positions and not about trying to control the exact angle of every joint. Appropriate foot placements for nontrivial turns, such as a 180° turn, or a 90° turn from forward walk to side-stepping, are all determined automatically by the optimizations. The placements are nontrivial because the planner needs to avoid motion that causes tangling of the legs during turning. All these behaviors happen at the same time each individual controller adapts itself to the terrain encountered at run-time.

To demonstrate the importance of making controllers terrain-adaptive, we compare the adaptive forward controller with a *blind* forward controller. The blind forward controller is optimized on uneven terrain and with the terrain adaptation turned off. To avoid unwanted collisions with the ground, the blind controller uses higher clearance, and the motion looks visibly less natural. From this we stipulate that terrain adaptation is an important aspect of the simulated locomotion strategy for the modeling of natural systems.

Because the planner plans in the end-effector space instead of the joint space, our controllers are capable of handling certain morphology changes of the character (Figure 4). We freeze one of the knees of the biped by setting the joint movement range to zero. The unaltered forward walk controller is still able to walk on uneven terrain in limping motion. The lower-level control automatically self-adjusts to use more pronounced motion at the hip to lift the swing foot. The unaltered forward walk controller also works on a biped with inverted knees. In a sense, path planning of key trajectories abstracts away many details of the biped structure and thus enables the controllers to work with these morphology changes.

Our tests with controllers for the running gait or on stairs with high rises show that it requires additional mechanisms to obtain good controllers. For example, the hip link tends to move erratically during running. An active spring that keeps the waist joint close to its neutral position is able to stabilize the movement. We also add active springs at the ankles and allow the offline optimization to tune their spring and damper coefficients. These active springs correspond to the active change of muscle stiffness in human [Farley and Gonzlez 1996], and we include the torques they exert in the energetic cost.

Figure 4: *Applying the unaltered forward controller to two different bipeds: one with its left knee frozen (top), and one with inverted knees (bottom).*

Because generating motion similar to that of real humans is one of our goals, we avoid including disturbances that natural systems do not routinely encountered. For example, we experimented with optimizing a controller by including external pushes of 700 N with 0.1 s durations. The controller lowers the center of mass and walks with knees always bent. This is a more robust strategy but not a "natural" one, most likely because natural systems do not routinely expect large disturbances. In addition, walking with knees always bent is more exhausting [Carey and Crompton 2005]. The optimized controller is not able to reliably recover from these pushes, and different controllers may be necessary for recovery from pushes coming from different directions. However, even though the normal forward walk controller is optimized without push disturbances, it can sustain sideways or forward pushes of 350 N with 0.1 s durations. The controller is more sensitive to backward pushes, and it either falls or recovers using less natural motion.

On a rougher height field with grid point heights distributed between 0 and 0.4 m, although the terrain adaptation appears to work well, occasionally the controller fails when stepping on a steep slope. Switching to controllers with slower motion, or a higher-level planner that plans ahead for more footsteps and places footfalls at flatter regions may be required to keep the biped from falling on rougher terrain.

10 Conclusion

In this paper, we present a method to automatically generate biped controllers capable of adapting to and navigating on uneven terrain. We show that planning of end-effector paths is an effective abstraction that makes balancing and adaptation to terrain more straightforward and can produce controllers capable of handling certain morphology changes. Furthermore, our experiments show that terrain adaptation results in controllers more natural than those that are blind to terrain. The synthesis is fully automatic, requiring no captured data or manual controller parameter tuning.

We note that the controller model does not use ZMP or other static stability rules to maintain balance. The controllers maintain balance by learning through optimization how to modify the end-effector paths at run-time in order to compensate for drift from the planned trajectory. This strategy leads to increased agility and allows our controllers to perform 90° and 180° turns in two footsteps by using motion that is not constrained by static stability rules.

Similar to most other methods that use the optimality principles in the optimization, our framework does not provide direct control over the final motion style to the users. Changes to the energetic

Parameter	Bounds	Parameter	Bounds
upper body k_p	[0, 4000]	t_e	[0, 0.6] × t_h
upper body b_p	[0, 800]	$\mathbf{c}_{sw}$	[-0.3, 0.3]
upper body c_p	[0, 140] × 9.8	k_x, k_y (in $\mathbf{K}_+$ and $\mathbf{K}_-$)	[0, 3]
upper body k_o	[0, 4000]	γ_1, γ_2	[0, 1]
upper body b_o	[0, 80]	θ_{sw1}, θ_{sw2}	[-180, 80] (degrees)
ankle k_p	[0, 4000]	$w_{sw1,u}$	[0, 0.8]
ankle b_p	[0, 400]	$w_{sw2,u}$	[0.2, 1.0]
ankle c_p	[0, 10] × 9.8	$w_{sw1,v}$, $w_{sw2,v}$	[0, 0.8]
toe & heel k_r	[0, 8000]	$w_{ub1,u}$	[0, 0.66]
toe & heel b_r	[0, 100]	$w_{ub2,u}$	[0.33, 1]
		$c_{ub,x}$, $c_{ub,y}$	[-0.3, 0.3]
		z_{ub}	[-0.3, 0.3] + z_{ub0}, z_{ub0}
		$\theta_{ub,eh}$, $\phi_{ub,eh}$	[-15, 15] (degrees)
		$\psi_{ub,eh}$	[-30, 30] (degrees)

Table 1: Bounds of unknowns in offline optimization. *Each half-cycle in a controller has its own path planning parameters, except for symmetric controllers. The offset z_{ub0} is the natural height of the upper body.*

from \ to	f	l	r	b	s
f	tl90, tr90, tl180, tr180	nt, tr90	nt, tl90	tl180, tr180	nt
l	nt, tl90	tl180	tr180	tr90	nt
r	nt, tr90	tl180	tl180	tl90	nt
b	tl180, tr180	tl90	tr90		nt
s	nt	nt	nt	nt	

Table 2: Transitions between cyclical controllers. *Cyclical controllers: f (forward), l (sideways left), r (sideways right), b (backward), s (stand). Transition types: nt (no turning), t{l,r}{90,180} (turn {left,right} {90,180} degrees). The facing directions are used as references for turning angles.*

cost's weight, or additional objective terms, such as one that penalizes large heel-strike impulses, need to be used as an indirect way to shape the optimization into producing motion closer to the desired styles. More dramatic morphology changes, such as one that results in a large horizontal COM shift, or large changes in the links' mass, will likely invalidate the unaltered controller. These larger morphology changes can be handled with re-optimization of the controller.

Robustness of the controllers can be extended by creating additional controllers using our automatic process specific to the disturbance variation, such as higher stairs, steeper slopes, and push disturbance. Coupled with a higher-level planner, the union of these automatically generated controllers would be significantly more robust [Coros et al. 2009; Chestnutt et al. 2003]. Because individual terrain-aware controllers can be created automatically, we expect that these controllers will be ideal building blocks for higher-level control that covers a broad range of natural locomotion behaviors.

Acknowledgments

Our thanks go to Emanuel Todorov and Jovan Popović for their feedback and comments on earlier versions of this paper. This work was supported by the UW Animation Research Labs, UW Center for Game Science, Microsoft, Intel, Adobe, and Pixar.

A Planner Details

Here we describe a specific planner whose path representation is very compact in terms of the number of optimization parameters. The paths computation process can be easily replaced with another, since lower-level control uses only the output paths from the planner.

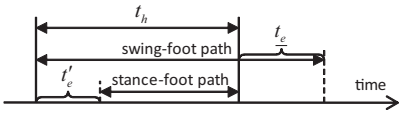

Figure 5: *Durations of foot paths. The swing-foot path duration extends into the next half-cycle. The stance-foot path starts at t'_e time into the current half-cycle, where t'_e is from the previous half-cycle.*

The angles ψ_f and ψ_m in the task goal are defined as the angles from the y-axis to the desired directions. We have designed the path planner to be orientation invariant: a planner that takes a specific task goal (ψ_f, ψ_m, r) can also be used when the goal is $(\psi_f + \Delta\psi, \psi_m + \Delta\psi, r)$ for any $\Delta\psi$, as long as the biped has been oriented and positioned properly at the start of the half-cycle.

In the following discussion, we shall mark each unknown parameter in Ω with an *underline*.

A.1 Timing

We define the (nonstandard) swing phase to also include the intervals when the foot is in contact with the ground but not stationary. We allow the swing phase to be longer than the half-cycle duration t_h by an additional amount of time $\underline{t_e}$ (Figure 5). We refer to the swing phase duration $t_h + \underline{t_e}$ as the footstep duration. The variable $\underline{t_e}$ is automatically determined by the optimization. The stance-foot path is stationary.

Given the elapsed half-cycle time t, we compute the normalized *half-cycle* time $s_h(t) = t/t_h$ and normalized (foot-)*step* time $s_s(t) = t/(t_h + \underline{t_e})$. We use the subscript eh to denote the end of a half-cycle and es the end of a footstep.

A.2 Targets

Swing-Foot horizontal target is computed in a two-step process. First, the horizontal target is placed (approximately) along the goal movement direction from the stance-foot horizontal position $\mathbf{q}_{st}$ at location $\mathbf{q}_{st} + \mathbf{q}_{sw,0}$. (We shall use $\mathbf{q}$ to refer to a point or vector on the horizontal x y-plane from here on.) Second, this location is adjusted by an offset $\mathbf{q}_{sw,bal}$ using the feedback from the upper body deviation $\Delta\mathbf{q}'_{ub,eh}$ from the previous path's target. The target position is then

$$\mathbf{q}_{sw,es} = \mathbf{q}_{st} + \mathbf{q}_{sw,0} + \mathbf{q}_{sw,bal} \tag{8}$$

The goal-based placement $\mathbf{q}_{sw,0}$ is determined by adding the goal movement $r\mathbf{y}$ (oriented using ψ_m) to the offsets that preserve the natural distance between feet due to the hip width (see Figure 6). Formally,

$$\mathbf{q}_{sw,0} = \mathbf{R}_z\left(\psi'_f\right)\mathbf{d}_0 + \mathbf{R}_z\left(\psi_m\right) r\mathbf{y} + \mathbf{R}_z\left(\psi_f\right)\mathbf{d}_0$$

where ψ'_f is the previous facing angle.

The adjustment $\mathbf{q}_{sw,bal}$ is represented as a piecewise linear function of the upper-body deviation in a local coordinate system. The local y-axis of the coordinate system is aligned with the movement direction defined by ψ_m. Formally,

$$\mathbf{q}_{sw,bal} = \mathbf{R}_z(\psi_m)(\underline{\mathbf{K}_+}\{\Delta\mathbf{q}'_{ub,eh}\}^+ + \underline{\mathbf{K}_-}\{\Delta\mathbf{q}'_{ub,eh}\}^- + \underline{\mathbf{c}_{sw}})$$

where $\underline{\mathbf{c}_{sw}}$ is a horizontal bias. $\underline{\mathbf{K}_+}$ and $\underline{\mathbf{K}_-}$ are diagonal gain matrices. Note that the upper-body deviation $\Delta\mathbf{q}'_{ub,eh}$ is computed in

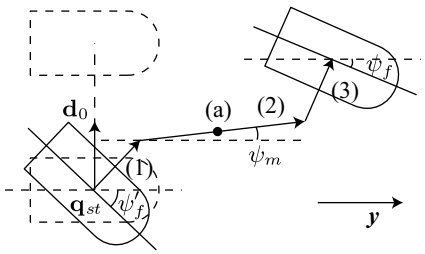

Figure 6: *Three components of* $\mathbf{q}_{sw,0}$*: (1)* $\mathbf{R}_z\left(\psi'_f\right)\mathbf{d}_0$*, (2)* $\mathbf{R}_z\left(\psi_m\right)r\mathbf{y}$*, and (3)* $\mathbf{R}_z\left(\psi_f\right)\mathbf{d}_0$*. Vector* $\mathbf{d}_0$ *is the offset from the stance foot to midway between the two feet when the biped is in the neutral position and facing the* $+y$ *direction. Neutral positions of the feet are shown with dotted lines. Point (a) is the upper-body target position before balance adjustment.*

the local coordinate system. We use a piecewise linear function because excessive adjustment may lead to leg collisions, but with the piecewise function the optimization can choose more conservative coefficients in the collision-prone half-space while still adjusting aggressively in the safer half-space.

The target z orientation $\psi_{sw,es}$ is set to point in the goal direction ψ_f. The terrain slope specifies the target x orientation $\theta_{sw,es}$ (the angle between the foot proximodistal axis and the x y-plane).

Upper-Body target position $\mathbf{p}_{ub,eh}$ is computed in a two-step process similar to the aforementioned swing-foot computation process. Its horizontal component is

$$\mathbf{q}_{ub,eh} = \mathbf{q}_{st} + \mathbf{q}_{ub,0} + \mathbf{q}_{ub,bal} \tag{9}$$

The goal-based offset would place the upper body at midway (i.e., the point (a)) of the second vector in Figure 6:

$$\mathbf{q}_{ub,0} = \mathbf{R}_z\left(\psi'_f\right)\mathbf{d}_0 + \frac{1}{2}\mathbf{R}_z\left(\psi_m\right)r\mathbf{y}$$

The linear (instead of piecewise linear) balance adjustment is

$$\mathbf{q}_{ub,bal} = \mathbf{R}_z(\psi_m)(\underline{\mathbf{K}_{ub}}\Delta\mathbf{q}'_{ub,eh} + \underline{\mathbf{c}_{ub}})$$

Because the upper body's vertical position is constrained by the leg length and the lower of the two foot positions, we define the vertical component of $\mathbf{p}_{ub,eh}$ to be the lower of the two foot positions at the end of the footstep plus an adjustable offset $\underline{z_{ub}}$.

The target upper-body orientation $\mathbf{R}_{ub,eh}$ is free varying. All three offset angles $\underline{\theta_{ub,eh}}$, $\underline{\phi_{ub,eh}}$, and $\underline{\psi_{ub,eh}}$ are determined by the optimization:

$$\mathbf{R}_{ub,eh} = \mathbf{R}_z(\psi_f)\mathbf{R}_y(\underline{\phi_{ub,eh}})\mathbf{R}_x(\underline{\theta_{ub,eh}})\mathbf{R}_z(\underline{\psi_{ub,eh}}) \tag{10}$$

Edge Avoidance is performed on terrain with vertical drops. The planner is extended to perform a simple search near $\mathbf{q}_{sw,es}$ for a location that would result in the least amount of foot rotation to avoid stepping on edges. The search is performed at $\mathbf{q}_{sw,es}$ and the two locations $\pm d_s$ distance way from $\mathbf{q}_{sw,es}$ in the direction defined by ψ_m. The distance d_s is set to 0.75 of the foot length. If the search results in $\Delta\mathbf{q}_a$ change in swing-foot target, we also offset the upper-body horizontal target by $\underline{\alpha_a}\Delta\mathbf{q}_a$.

A.3 Paths

We use three types of curves to represent the paths. In increasing order of flexibility, they are (I) a simple slow-in/slow-out curve, (II) a Bézier curve, and (III) a Bézier curve whose timing is controlled by another Bézier curve.

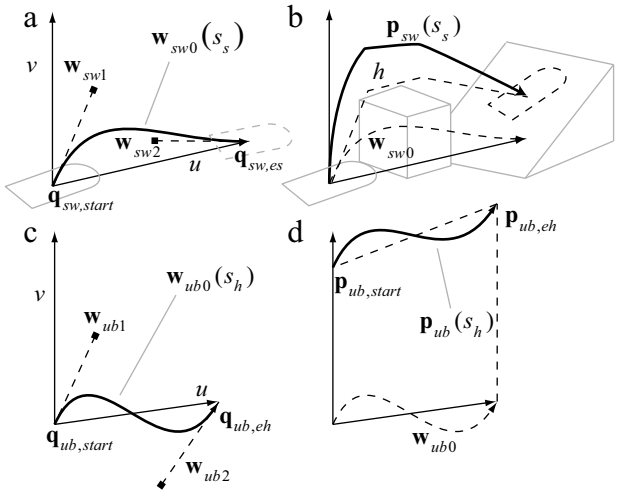

Figure 7: *(a) Initial swing-foot path* $\mathbf{w}_{sw}(s_s)$ *in local* u v*-coordinate has two tunable control points* $\mathbf{w}_{sw1}, \mathbf{w}_{sw2}$*. (b) Swing-foot path* $\mathbf{p}_{sw}(s_s)$ *after height adjustment. (c) Initial upper-body path* $\mathbf{w}_{ub}(s_h)$ *in its local* u v*-coordinate has two tunable control points* $\mathbf{w}_{ub1}, \mathbf{w}_{ub2}$*. (d) Upper-body path* $\mathbf{p}_{ub}(s_h)$ *after slope height adjustment.*

I. $C(s, \alpha_0, \alpha_1) = \frac{\alpha_1 - \alpha_0}{2}(1 - \cos(s\pi)) + \alpha_0$

II. $B(s, \alpha_0, \alpha_1, \alpha_2, \alpha_3) = \sum_{i=0}^{3} \mathsf{b}_i(s)\alpha_i$

III. $V(s, \alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$
$= B(B(s, 0, \alpha_4, \alpha_5, 1), \alpha_0, \alpha_1, \alpha_2, \alpha_3)$

where s is the normalized time (can be either s_h or s_s), and b_i are the Bernstein polynomials. The α's are standins for the actual parameters used in the planner, or the starting or end point.

Swing-Foot position path lies in the vertical plane defined by the points $\mathbf{q}_{sw,start}$ and $\mathbf{q}_{sw,es}$ and the global z-axis. At run-time, this curve is height-adjusted to make it collision free with terrain.

More formally, let $\mathbf{q}_{sw,start}$ denote the horizontal position of the swing foot at the start of the half-cycle, and $\mathbf{q}_{sw,es}$ the target horizontal position at the end of the step. The initial swing-foot position path is a Type III curve in the u v-coordinate system defined using $\mathbf{q}_{sw,start}$ and $\mathbf{q}_{sw,es}$ (Figure 7a). The origin of the u v-coordinate system is located at $\mathbf{q}_{sw,start}$; the u-axis is aligned with the vector $(\mathbf{q}_{sw,es} - \mathbf{q}_{sw,start})$ with $(u, v) = (1, 0)$ at $\mathbf{q}_{sw,es}$. The v-axis parallels, and has the same scale as, the global z-axis. The swing-foot path has two remaining internal 2D control points $\underline{\mathbf{w}_{sw1}}$ and $\underline{\mathbf{w}_{sw2}}$. Its timing is controlled by two parameters $\underline{\gamma_1}$ and $\underline{\gamma_2}$:

$$\mathbf{w}_{sw0}\left(s_s\right) = V\left(s_s, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \underline{\mathbf{w}_{sw1}}, \underline{\mathbf{w}_{sw2}}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \underline{\gamma_1}, \underline{\gamma_2}\right)$$

It is also possible to define $\underline{\mathbf{w}_{sw1}}$ and $\underline{\mathbf{w}_{sw2}}$ as 3D control points to make the path more flexible during turning as in our implementation. The extension is straightforward, and we describe only the simpler 2D case here.

The swing-foot orientation paths vary only two angles (twist about the foot proximodistal axis is fixed to 0). The z orientation path is a Type I curve with starting and end points $\psi_{sw,start}$ and $\psi_{sw,es}$;

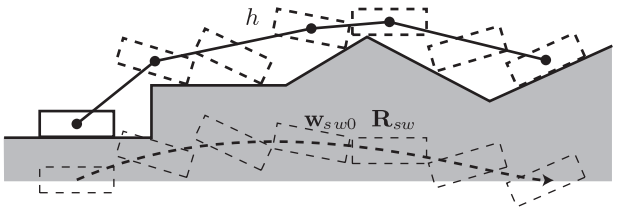

Figure 8: *Collision-free convex hull of foot heights.*

the x orientation path is a Type III curve with endpoints $\theta_{sw,start}$ and $\theta_{sw,es}$, and free varying internal control points $\underline{\theta_{sw1}}$ and $\underline{\theta_{sw2}}$.

Specifically, the z and x orientation angles of the swing-foot during the footstep are, respectively,

$$\psi_{sw}(s_s) = C(s_s, \psi_{sw,start}, \psi_{sw,es})$$
$$\theta_{sw}(s_s) = V\left(s_s, \theta_{sw,start}, \underline{\theta_{sw1}}, \underline{\theta_{sw2}}, \theta_{sw,es}, \underline{\gamma_1}, \underline{\gamma_2}\right)$$

Note that θ_{sw} shares the same timing control parameters with $\mathbf{w}_{sw0}$. From these two we can compute the orientation path

$$\mathbf{R}_{sw}(s_s) = \mathbf{R}_z(\psi_{sw}(s_s))\,\mathbf{R}_x(\theta_{sw}(s_s)) \tag{11}$$

The path $\mathbf{w}_{sw0}(s_s)$ is adjusted to ensure that it is collision free: $\mathbf{w}_{sw0}(s_s)$ assumes that the ground is the x y-plane and may result in collisions of the foot with the actual terrain. We uniformly sample along the local u-axis at 64 locations using horizontal positions and orientations from $\mathbf{w}_{sw0}(s_s)$ and $\mathbf{R}_{sw}(s_s)$. At each location, the lowest collision-free vertical position of the foot is determined using the oriented bounding box of the foot. We then construct the convex hull $h(u)$ from these collision-free positions (Figure 8). The convex hull h is added to $\mathbf{w}_{sw0}(s_s)$ as a vertical offset to produce the swing-foot position path $\mathbf{p}_{sw}(s_s)$ (Figure 7b).

Upper-Body position path is defined in a plane of two predetermined endpoints $\mathbf{q}_{ub,start}$ and $\mathbf{q}_{ub,eh}$ and has two freely varying control points $\underline{\mathbf{w}_{ub1}}$ and $\underline{\mathbf{w}_{ub2}}$ (Figure 7c). The construction process is similar to that of the swing-foot path. However, unlike the swing-foot path, it is a Type II curve instead of Type III. The height adjustment is also simpler here: instead of using the convex hull, we add the linear slope defined by the heights of the upper body's starting and target positions $\mathbf{p}_{ub,start}$ and $\mathbf{p}_{ub,eh}$ to get the final upper-body path $\mathbf{p}_{ub}(s_h)$ (Figure 7d). The upper-body orientation path is a Type I curve interpolating between the starting and end orientations.

B Null-Space Control

The biped is in a singular configuration when the knee is fully extended: the rank of the Jacobian matrix decreases at this configuration, and a force from ideal PD controllers that attempts to pull the ankle towards the hip, for example, results in no torque at the knee. Near the singular configuration, we may compute and apply an additional torque $\boldsymbol{\tau}_\emptyset$ to bring the system away from the configuration when necessary: when the current distance $\hat{d}$ between the ankle and the upper body is larger than the planned distance d (the distance is computed along the direction from the ankle to the hip joint), we set the additional bending torque at the knee to

$$\tau_{\emptyset,knee} = \rho\cos(\theta_{knee})\{\hat{d} - d\}^+$$

where $\theta_{knee} = 0$ when the knee is fully extended, and the cos function smoothly reduces the additional torque as the system moves away from the singularity. We set the other elements in $\boldsymbol{\tau}_\emptyset$ to 0.

link	mass (kg) and (I_x, I_y, I_z) (kg·m^2)	length (m)	joint type	τ_{max} (N·m)
upper body	40, (1, 1, 0.1)	0.8		
			spherical	1600
hip	8.8, (0.1, 0.1, 0.1)	0.15 (radius)		
			spherical	1200
thigh	8.552, (0.1, 0.1, 0.05)	0.5		
			revolute	700
shank	4.352, (0.04, 0.04, 0.01)	0.55		
			spherical	400
foot	1, (0.03, 0.001, 0.03)	0.25		

Table 3: *Link and joint properties.*

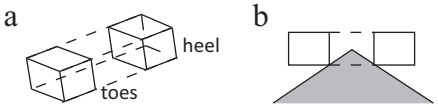

Figure 9: *(a) Foot collision geometries. (b) The gap in the middle of the foot.*

We include the coefficient ρ in the lower-level control parameter set Π and use the offline optimization to tune its value. We also need to compute a canceling torque $\mathbf{J}^\top\mathbf{f}_\emptyset$ to keep the end-effector dynamics unaffected [Khatib 1987]. We find the canceling torque by solving a QP for $\mathbf{f}_\emptyset$ that minimizes $\left\|\boldsymbol{\tau}_\emptyset + \mathbf{J}^\top\mathbf{f}_\emptyset\right\|_2^2$. The final joint torque becomes

$$\boldsymbol{\tau}_\theta = \mathbf{J}^\top\mathbf{f} + \left(\boldsymbol{\tau}_\emptyset + \mathbf{J}^\top\mathbf{f}_\emptyset\right) \tag{12}$$

C Implementation Notes

We use the NVIDIA PhysX SDK in software mode (no hardware acceleration) for the physics simulation [NVIDIA Corporation 2008]. The timestep size is 1/64 seconds, and therefore the controllers operate at 64 Hz. PhysX further divides each timestep into 32 sub-timesteps internally. The friction coefficient between the biped and terrain is set to 1. We list the important link and joint properties in Table 3. A gap between the toes and heel geometries helps the foot "bite" into the terrain to improve stability (Figure 9). Because our path planner lacks the intricacy in resolving foot collisions with the other body links, we disable such collisions in the simulation. The leg links can still collide with each other to prevent the optimization from exploiting gaits with interpenetrating legs.

Each offline optimization performs parallel computation on 4 machines with dual quad-core 2.66 GHz processors. For a cyclical-controller optimization, the computation time for the maximum 4000 iterations is 2.5 hours. Including rendering time, the interactive demo takes up 45% (or less than 5% without upper limbs) CPU time on a machine with a 2.10 GHz Intel Core 2 Duo Processor running at 64 frames per second.

References

ABE, Y., DA SILVA, M., AND POPOVIĆ, J. 2007. Multiobjective control with frictional contacts. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 249–258.

ABEND, W., BIZZI, E., AND MORASSO, P. 1982. Human Arm Trajectory Formation. *Brain 105*, 2, 331–348.

ATKESON, C., AND MORIMOTO, J. 2002. Nonparametric representation of policies and value functions: A trajectory-based approach. In *Neural Information Processing Systems 2002*.

BERNSTEIN, N. 1967. *The Co-ordination and Regulation of Movements*. Oxford: Pergamon Press.

BULLOCK, D., AND GROSSBERG, S. 1988. Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review 95*, 1, 49–90.

CAREY, T. S., AND CROMPTON, R. H. 2005. The metabolic costs of 'bent-hip, bent-knee' walking in humans. *Journal of Human Evolution 48*, 1, 25 – 44.

CHESTNUTT, J., KUFFNER, J., NISHIWAKI, K., AND KAGAMI, S. 2003. Planning biped navigation strategies in complex environments. In *Proceedings of the 2003 International Conference on Humanoid Robots*.

COROS, S., BEAUDOIN, P., YIN, K. K., AND VAN DE PANN, M. 2008. Synthesis of constrained walking skills. *ACM Transactions on Graphics 27*, 5 (Dec.), 113:1–113:9.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2009. Robust task-based control policies for physics-based characters. *ACM Transactions on Graphics 28*, 5 (Dec.), 170:1–170:9.

CRAIG, J. J. 1989. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum 27*, 2, 371–380.

FARLEY, C. T., AND GONZLEZ, O. 1996. Leg stiffness and stride frequency in human running. *Journal of Biomechanics 29*, 2, 181 – 186.

HANSEN, N. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, Eds. Springer, 75–102.

HAUSER, K., BRETL, T., LATOMBE, J.-C., HARADA, K., AND WILCOX, B. 2008. Motion Planning for Legged Robots on Varied Terrain. *The International Journal of Robotics Research 27*, 11-12, 1325–1349.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of SIGGRAPH 95*, ACM, Computer Graphics Proceedings, Annual Conference Series, 71–78.

KHATIB, O. 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of 3*, 1 (Feb.), 43–53.

KUFFNER, J.J., J., NISHIWAKI, K., KAGAMI, S., INABA, M., AND INOUE, H. 2001. Footstep planning among obstacles for biped robots. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, 500–505.

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96*, ACM, Computer Graphics Proceedings, Annual Conference Series, 155–162.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (Jul.), 1071–1081.

MOSEK APS, 2009. The mosek optimization software version 6.0. http://www.mosek.com/.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28*, 3 (Aug.), 81:1–81:9.

NOCEDAL, J., AND WRIGHT, S. J. 2006. *Numerical Optimization*, 2nd ed. Springer.

NVIDIA CORPORATION, 2008. NVIDIA PhysX SDK version 2.8.1. http://developer.nvidia.com/object/physx.html.

PRATT, J., DILWORTH, P., AND PRATT, G. 1997. Virtual model control of a bipedal walking robot. In *IEEE Conference on Robotics and Automation*, 193–198.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings SIGGRAPH 91)*, ACM, 349–358.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 96*, ACM, Computer Graphics Proceedings, Annual Conference Series, 147–154.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (Aug.), 514–521.

STEWART, A. J., AND CREMER, J. F. 1992. Animation of 3d human locomotion: climbing stairs and descending stairs. In *Eurographics Workshop on Animation and Simulation*, 152–168.

SUN, H. C., AND METAXAS, D. N. 2001. Automating gait generation. In *Proceedings of SIGGRAPH 2001*, ACM, Computer Graphics Proceedings, Annual Conference Series, 261–270.

TODOROV, E. 2004. Optimality principles in sensorimotor control. *Nature Neuroscience 7*, 9 (Sep.), 907–915.

VAN DE PANNE, M., AND LAMOURET, A. 1995. Guided optimization for balanced locomotion. In *6th Eurographics Workshop on Animation and Simulation, Computer Animation and Simulation, September, 1995*, Springer, Maastricht, Pays-Bas, D. Terzopoulos and D. Thalmann, Eds., Eurographics, 165–177.

VAN DE PANNE, M., FIUME, E., AND VRANESIC, Z. 1992. A controller for the dynamic walk of a biped across variable terrain. In *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, 2668 –2673 vol.3.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics 28*, 3 (Aug.), 60:1–60:8.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing walking controllers. *ACM Transactions on Graphics 28*, 5 (Dec.), 168:1–168:8.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (Jul.), 105:1–105:10.

YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2008. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics 27*, 3 (Aug.), 81:1–81:7.

ZHANG, L., PAN, J., AND MANOCHA, D. 2009. Motion planning of human-like robots using constrained coordination. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, 188 –195.