Optimizing Walking Controllers for Uncertain Inputs and Environments

Jack M. Wang David J. Fleet Aaron Hertzmann

University of Toronto

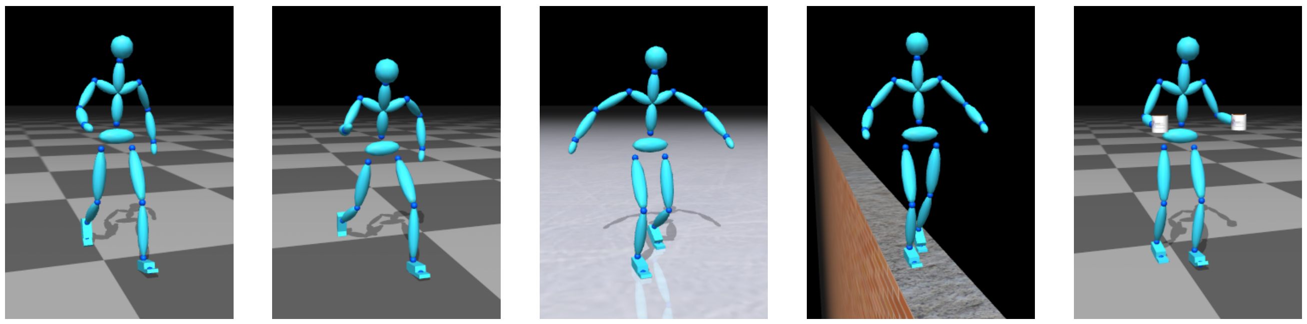


Figure 1: ***Walking controllers optimized for different environments with uncertainty.*** ***(a)*** *Walking can be relaxed in a deterministic environment, without random external perturbations.* ***(b)*** *Under gusty conditions, the gait is more aggressive, with a wider stance.* ***(c)*** *On a slippery surface with internal motor noise, the gait is cautious with arms extended for balance.* ***(d)*** *Walking on a narrow wall on a windy day produces a narrower gait with small steps.* ***(e)*** *With internal motor noise, carrying hot beverages requires a slow gait with steady arms.*

Abstract

We introduce methods for optimizing physics-based walking controllers for robustness to uncertainty. Many unknown factors, such as external forces, control torques, and user control inputs, cannot be known in advance and must be treated as uncertain. These variables are represented with probability distributions, and a return function scores the desirability of a single motion. Controller optimization entails maximizing the expected value of the return, which is computed by Monte Carlo methods. We demonstrate examples with different sources of uncertainty and task constraints. Optimizing control strategies under uncertainty increases robustness and produces natural variations in style.

Keywords: Physics-based animation, controller synthesis, human motion, optimization.

1 Introduction

When designing controllers for character locomotion, one cannot know with certainty all factors that will influence the character's motion at run-time. Many unknown factors, such as external forces due to strong winds, interactive user inputs, or noise in the motor control system, can be significant and must be treated as uncertain. One might try to cope with such uncertainty by making the controller stiff and deliberate. However, it would be more desirable for the control strategy to adapt to different scenarios, much as humans do. For example, when walking on a slippery surface of variable roughness, one might improve stability with a lower center-of-mass (COM), outstretched arms, and smaller steps. Alternatively, when experiencing extreme gusty winds, one might walk more stiffly with a widened stance to avoid being blown over.

This paper introduces a technique for automatically learning robust control strategies under different scenarios and various sources of uncertainty. We consider four sources of uncertainty. First, we incorporate unknown, varying external forces acting on the body, like wind on a gusty day. Second, we incorporate uncertainty due to user inputs. When designing controllers for interactive simulation, *a priori* one cannot know a user's run-time control inputs, e.g., to change a character's heading or speed. A third source of uncertainty arises when transitions occur between controllers, where the start state for one controller depends on the state produced by the previous controller at the point of transition. Any variability in the timing of the transition, or in the motion produced by the first controller, will produce start states for the second controller that cannot be known *a priori*. The fourth source of uncertainty we consider is motor noise. In humans, it is believed that neuronal motor noise influences motor strategies, e.g., in the coordination of eye saccades, finger pointing, and in line drawing [Harris and Wolpert 1998; Todorov 2004; Körding 2007]. Accordingly, the inclusion of motor noise may help to produce human-like control strategies under different environmental conditions. Noise is also a convenient way to capture a wide range of otherwise unmodeled, complex sources of uncertainty that might significantly influence a character's motion.

Learning different control strategies is formulated in terms of optimizing 3D locomotion controllers in the presence of unknown environmental variables and controller inputs. We use a probabilistic formulation in which all prior beliefs over unknown quantities are modeled by probability distributions. Together with a controller and dynamics, they define a probability distribution over motions. A *return function* scores the quality of a given motion. Our goal then is to optimize a controller to maximize the *expected return*, a quantity not computable in closed-form. We use Monte Carlo methods to approximate the expected return. This approximation is op-

http://www.dgp.toronto.edu/~jmwang/optuie/

timized by Covariance Matrix Adaptation (CMA) [Hansen 2006]. As a result, the character's control strategy and style of movement are determined automatically as a function of stochastic variables and the return function.

Our controllers exhibit increased robustness compared to *baseline controllers* that are optimized for scenarios where all significant factors are known *a priori*. For example, the amount of unexpected external force that a given walking controller can withstand can be greatly increased. Furthermore, the type and degree of robustness can be controlled through the specification of probability distributions over the unknowns. Different tasks and types of uncertainty together lead to different styles of movement. For example, a character walking on a slippery surface with increased motor noise tends to extend his arms and bend his knees for balance and stability. In contrast, a controller walking on a narrow beam with uncertain external forces will be conservative, taking relatively small steps with a narrow gait width to avoid stepping too close to the edge.

Optimization under uncertainty is also useful when composing controllers. In general, switching between controllers is unreliable, unless the controllers are highly robust, or they operate in similar regions of state space. We find that controllers optimized under uncertainty are generally more robust and therefore more tolerant to state variability at transitions. We also show that controllers optimized with uncertain start states can be used to create transition and recovery controllers to facilitate composition.

The approach we advocate here is conceptually intuitive and broadly applicable, supporting an extremely general class of uncertainty. Given a basic controller and CMA, this method is also very simple to implement. Optimization under uncertainty is expensive, often requiring overnight computations. Nevertheless, once optimized, the resulting controllers run at real-time rates.

2 Related Work

Previous work in optimal control for physics-based animation assumed deterministic dynamical systems. Grzeszczuk et al. [1995; 1998] demonstrate control optimization for high-dimensional marine animals. In land-based locomotion, early methods focused on low-dimensional characters [van de Panne and Fiume 1993; Sims 1994; van de Panne and Lamouret 1995]. More recently, controllers based on tracking motion capture data have been optimized for both 2D [Sharon and van de Panne 2005; Sok et al. 2007] and 3D [da Silva et al. 2008; Muico et al. 2009] humanoid locomotion. Because the resulting motion style is defined by the motion capture data, such techniques cannot be used to investigate the effect of uncertainty on style. In our previous work [Wang et al. 2009], we optimized full-body locomotion controllers in deterministic settings. These controllers are robust only to limited external disturbances, and do not exhibit stylistic variation as a function of uncertainty. We extend this approach to stochastic environments and user inputs, and show that different sources of uncertainty lead to stylistically different control strategies.

Some techniques have been proposed for switching between low-level physics-based controllers. Faloutsos et al. [2001] describe a method for learning at what instants one can transition from one controller to another. Coros et al. [2009] learn optimal control policies to transition between low-level controllers to accomplish different tasks. These methods assume that the low-level controllers, and hence the motion style, are specified in advance and fixed. Our approach complements these, as we focus on learning low-level controllers, including those capable of making transitions.

Reinforcement learning has been used for control in kinematic motion graphs [Lee and Lee 2006; Treuille et al. 2007; Lo and Zwicker 2008; Lee et al. 2009]. These techniques assume completely deterministic systems. McCann and Pollard [2007] proposed the use of a probability distribution over user inputs, and maximize expected return with respect to this distribution. Motion graph control is significantly different from low-level physical control, however, because actions in motion graphs are discrete and low-dimensional. Motion-graph methods do not explicitly handle the possibility of control failure, such as tripping and falling.

As robotic controllers must account for uncertain conditions, optimization has been used for several low-dimensional robotic controllers. For example, Tedrake et al. [2004] optimize the control of a low-dimensional passive-based humanoid walker. This walker is designed to be very stable even in the absence of control, thereby reducing greatly the role of uncertainty. Abbeel et al. [2007] demonstrate controllers for impressive helicopter stunts, which is a system with low-dimensional dynamics but no ground-contact discontinuities. Byl and Tedrake [2009] optimize a two-dimensional walker in the presence of stochastic ground roughness.

3 Optimal Control under Uncertainty

Our approach to controller design combines probabilistic modeling of uncertain (i.e., unknown) quantities, and optimization under uncertainty. We begin with the formulation of the deterministic case.

3.1 Deterministic simulation and optimization

An articulated rigid-body character at time t is represented by a state vector $\mathbf{s}_t$, comprising joint angles, root position and root orientation. Articulated rigid-body dynamics and integration provides a function for mapping the state $\mathbf{s}_t$ at time t to the state at the next time instant $\mathbf{s}_{t+1}$:

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \boldsymbol{\tau}_t, \mathbf{e}_t)\,. \tag{1}$$

This mapping is a function of state $\mathbf{s}_t$, the internal joint torques $\boldsymbol{\tau}_t$ produced by the character controller, and the relevant world parameters $\mathbf{e}_t$, such as external forces (e.g., wind) acting on the character.

The control torques $\boldsymbol{\tau}_t$ are determined by a controller. A controller, π, defines a mapping from the current state, $\mathbf{s}_t$, to the desired control signal:

$$\boldsymbol{\tau}_t = \pi(\mathbf{s}_t, \mathbf{c}_t;\, \mathbf{w})\,, \tag{2}$$

where $\mathbf{w}$ is a vector of control parameters, and $\mathbf{c}_t$ is a vector of user inputs at time t. Given a start state $\mathbf{s}_1$, the control parameters $\mathbf{w}$, any user inputs $\mathbf{c}_{1:T}$ and environmental parameters $\mathbf{e}_{1:T}$, a deterministic simulation is performed by recursively applying (2) to determine $\boldsymbol{\tau}_t$, and then (1) to compute $\mathbf{s}_{t+1}$ for each time-step $1 \leq t < T$. This yields an animation sequence $\mathbf{s}_{1:T}$.

We formulate the optimal control problem with the specification of a *return function* (e.g., see [Sutton and Barto 1998]), denoted $R(\mathbf{s}_{1:T})$, which measures the quality of a simulation. In the deterministic case, optimal control seeks the control parameters $\mathbf{w}$ which produce motions that maximize the return function [van de Panne and Fiume 1993; Grzeszczuk and Terzopoulos 1995; Wang et al. 2009]. We use a return function based on a simplified, but more general, version of the energy terms in [Wang et al. 2009]. These are described in the Appendix.

A controller optimized for a specific run-time environment without uncertainty can sometimes be robust to moderate perturbations, but typically not to large perturbations, changes in environmental conditions, or variations in start state that might occur when controllers are composed. For example, a carefully chosen $\mathbf{w}$ that allows the character to walk in a straight line may fail when the character is pushed in the chest, or when a user attempts to change the heading or the speed of the character.

Character and controller details. Our character model is identical to that of Wang et al. [2009], which is based on SIMBICON [Yin et al. 2007]. There are 30 internal degrees-of-freedom (DOF). Mass distributions are computed from an automatically generated mesh [Hasler et al. 2009] that approximates a male with height 180 cm, and weight 70 kg. The controller has four states, each with linear proportional-derivative (PD) joint control combined with SIMBICON-like balance control. The control parameterization $\pi(\mathbf{s}_t, \mathbf{c}_t; \mathbf{w})$ is like that in [Wang et al. 2009], but with several minor differences. First, the target angles that rotate the shoulders in the coronal plane are not manually specified; instead they are optimized. Second, to reduce the number of optimized parameters, and to encourage smoothness of the body's motion, several control parameters are constrained to share the same value in all four controller states. These include the damping and stiffness constants for each upper body DOF, and the target values for the elbow angles and the shoulder angle of rotation in the coronal plane.

We make one final change to the controller. Although the target angle for the stance hip in the transverse plane (θ_t) can be specified as a user input to control the character's heading direction [Yin et al. 2007], direct manipulation of this parameter, e.g., by the user, can easily lead to failure. Instead, we define an additional time-varying *target heading direction* θ_d, which determines the change in θ_t from one time-step to the next, and is better suited to be a user input. If the new θ_d is close to the current target heading, θ_c, we immediately change the target angle to θ_d. When θ_d differs significantly from θ_c, the target stance hip angle is adjusted gradually, to reduce instability. In particular, it is updated from one time-step to the next using

$$\theta_{t+1} = \alpha_{hip}(\theta_d - \theta_t) + (1 - \alpha_{hip})(\theta_c - \theta_t) + \theta_t \,, \qquad (3)$$

where θ_t is the stance hip target angle at time t, and α_{hip} is a weight parameter. Each controller state has two values for α_{hip}, for when $\theta_c > \theta_d$ or $\theta_c < \theta_d$, respectively. All these parameters are optimized along with the other controller parameters.

3.2 Random environments and optimal control

When user inputs and environmental variables are uncertain, we do not have specific values for $\mathbf{c}_t$ and $\mathbf{e}_t$ *a priori*. Rather, we characterize our limited prior knowledge by specifying a probability distribution over each unknown variable. As examples, here we consider four general types of uncertainty. First, environmental uncertainty is represented by $p(\mathbf{e})$; this might represent, for example, the distribution over wind forces applied to the character's torso. User inputs, such as commands to change the character's heading, are also unknown *a priori*. We therefore include a distribution over possible user inputs $p(\mathbf{c})$. Third, the precise initial state of a controller is often unknown during run-time applications. For example, we may wish to compose controllers where transitions from one controller to another occur at variable time instants. Accordingly, we can specify a distribution over start states for a given controller with a second distribution $p(\mathbf{s}_1)$. Finally, human motor neurons are subject to signal-dependent noise [Faisal et al. 2008], which is thought to play a significant role in human motion planning [Harris and Wolpert 1998]. We incorporate motor noise by perturbing the joint torques produced by the controller. To this end we specify a distribution over joint torques, $\boldsymbol{\tau}$, given the parameters of the controller, the current state, and the user inputs, i.e., $\boldsymbol{\tau}_t \sim p(\boldsymbol{\tau}|\mathbf{s}_t, \mathbf{c}_t, \mathbf{w})$.

Together, these sources of uncertainty and noise, in combination with the dynamics (1), define a probability distribution over animations $p(\mathbf{s}_{1:T}|\mathbf{w})$. Despite the complexity of this distribution, it is rather straightforward to draw fair samples from it. To sample an animation sequence from $p(\mathbf{s}_{1:T}|\mathbf{w})$, one first samples a start state $\mathbf{s}_1 \sim p(\mathbf{s}_1)$. Then, for each time-step t, the environmental variables $\mathbf{e}$ and the user inputs $\mathbf{c}$, if desired, are sampled from their distributions: $\mathbf{e}_t \sim p(\mathbf{e})$ and $\mathbf{c}_t \sim p(\mathbf{c})$. Joint torques $\boldsymbol{\tau}_t$ are sampled as $\boldsymbol{\tau}_t \sim p(\boldsymbol{\tau}|\mathbf{s}_t, \mathbf{c}_t, \mathbf{w})$. Finally, the next state $\mathbf{s}_{t+1}$ is computed according to the dynamics (1).

With random variables, we can no longer optimize the return of a single scenario, and then expect the controller to work during another simulation for which random variables take on different values.[1] Instead, we optimize the *expected return*. The expected return of a controller, with control parameters $\mathbf{w}$, is

$$\begin{aligned} V(\mathbf{w}) &\equiv E_{p(\mathbf{s}_{1:T}|\mathbf{w})}\left[R(\mathbf{s}_{1:T})\right] \\ &= \int p(\mathbf{s}_{1:T}|\mathbf{w})\, R(\mathbf{s}_{1:T})\, d\mathbf{s}_{1:T} \,. \qquad (4) \end{aligned}$$

The optimal control problem is to select $\mathbf{w}$ to maximize this expected return. By optimizing the expected return, we aim to find a controller that will work well, on average, with plausible values for the uncertain quantities. The same principles have been effective in modeling human motor control [Todorov 2004; Körding 2007].

3.3 Evaluation and optimization

Due to nonlinear dynamics and non-Gaussian noise, the expected return (4) cannot be computed analytically; hence we use Monte Carlo methods [Sutton and Barto 1998]. Specifically, N animation sequences are sampled, as described above. The approximation $\hat{V}(\mathbf{w})$ is then computed as the average return on these sequences:

$$\hat{V}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} R(\mathbf{s}_{1:T}^{(i)}) \,, \qquad (5)$$
$$\text{where } \mathbf{s}_{1:T}^{(i)} \sim p(\mathbf{s}_{1:T}|\mathbf{w}) \,.$$

For example, suppose the only source of randomness is an external force, $\mathbf{f}$, applied to the character's torso at time t, where the direction and magnitude of the force are uniformly distributed. We run the simulation N times, each time applying different forces drawn at random from the uniform distribution. To optimize $\hat{V}(\mathbf{w})$, we use the CMA algorithm [Hansen 2006].

Note that, because new motions are sampled for each evaluation, $\hat{V}(\mathbf{w})$ is a random quantity. Hence, even if $\mathbf{w}$ is fixed, one obtains a different result each time $\hat{V}(\mathbf{w})$ is evaluated. This can cause problems for optimization algorithms. One issue is as follows. Suppose, when comparing two controllers $\mathbf{w}_1$ and $\mathbf{w}_2$, we obtain estimates for which $\hat{V}(\mathbf{w}_1) > \hat{V}(\mathbf{w}_2)$. We cannot tell whether this is because $\mathbf{w}_1$ is really better than $\mathbf{w}_2$, or if the random forces sampled for the second evaluation were more challenging than those for the first. We address this by using the method of Common Random Numbers (CRN) [Spall 2003], also known as PEGASUS [Ng and Jordan 2000]. In CRN, one reuses the same random seed each time $\hat{V}(\mathbf{w})$ is evaluated. This makes $\hat{V}(\mathbf{w})$ deterministic: for the example above, the same sample of N random forces would be used in each evaluation. This resolves difficulties with many optimizers, and, under certain conditions, can be shown to yield better results with high probability. We find that effective controllers can be optimized with small values of N (e.g., see Fig. 4).

[1] However, even for a deterministic system, simulations of multiple walk cycles will exhibit small deviations from perfect gait periodicity. Controllers optimized for long simulations will therefore be somewhat more robust than those optimized for short durations.

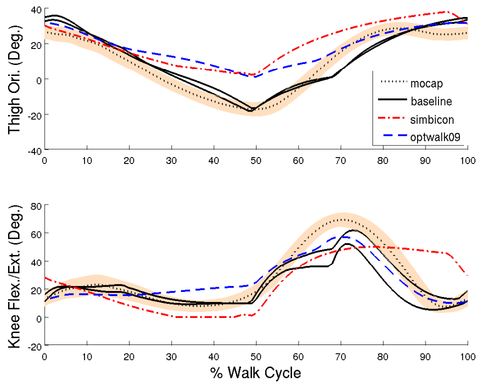


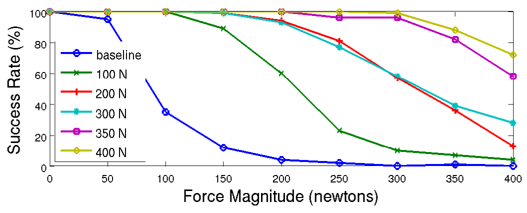
Figure 2: ***Comparison of baseline controllers to motion capture data. Top:*** *Thigh orientation w.r.t. down vector.* ***Bottom:*** *Angle between thigh and shin. The dotted curve and orange region represent the mean and standard deviation of walk cycles from 115 subjects. Dashed blue and red curves represent the mean of optimized controllers of [Wang et al. 2009] and SIMBICON, respectively. Solid lines represent baseline controllers: optimized with no user constraints and the 1.6 m/s controller described in Sec. 4.4.*

With each application, the controllers are optimized using the method of CRN with $N = 10$ simulations, and CMA. The simulator frequency is 2400 Hz (time-steps are about 0.00042 s), and each simulation run is 10 s long. Following our previous work [Wang et al. 2009], we run 19 CMA samples in parallel per iteration. The optimization here is, however, more expensive, because each evaluation of the return function requires N simulations. Convergence typically requires a few hundred iterations. Running the optimizations overnight on a cluster of 20 CPUs is usually sufficient. Further parallelization of the N independent samples is possible.

Because the optimization is nonconvex, a good initial guess is required and local minima are problematic. We first optimize a controller with no uncertainty, as in Sec. 3.1. This *baseline controller* is then used as the initial guess for controller optimization under uncertainty. When uncertainty or variability in environmental conditions is extreme, it is easy for the optimization to get trapped in poor local minima. In these cases, we first optimize controllers for smaller noise levels or perturbations (often with early stopping). These are then used to initialize optimizations for higher levels of noise. We find that incremental optimization, while slowly changing conditions from baseline controllers to the desired scenarios, typically produces effective controllers.

4 Applications

To demonstrate the impact and generality of optimization under uncertainty, we consider several applications where different environment conditions and sources of uncertainty combine to produce different strategies for robustness. These controllers are compared against their corresponding baseline controllers to evaluate the significance of uncertainty. The baseline controllers are similar to those described in our previous work [Wang et al. 2009], but with small differences in control parameterization, objective function, and mass distribution, the gaits produced by baseline controllers appear more natural. Many features, such as knee angle (Fig. 2(bottom)) behave as before, while others, such as the thigh orientation trajectory (Fig. 2(top)) bear greater similarity to motion capture data.



direction	0 N (baseline)	100 N	200 N	300 N	350 N	400 N
$(1, 0)$	125	175	275	375	350	425
$(1, \pm1)$	75	125	225	275	300	325
$(0, \pm1)$	75	125	325	375	425	475
$(-1, \pm1)$	25	100	175	225	250	275
$(-1, 0)$	75	200	350	325	375	375

Figure 3: ***Controller robustness to pushes.*** *During optimization and testing, random forces to the torso are parallel to the ground with uniformly distributed directions over $(0, 2\pi]$. At each simulation time instant, a force lasting 0.4 s is initiated with probability 0.025% (approximately 6 pushes in a 10 s simulation). Controllers were optimized in sequence, with those for smaller force magnitudes F used to initialize optimizations for larger F.* ***Top:*** *Success rate vs. F, averaged over 100 trials at each magnitude F. Controllers optimized with larger forces (see legend) are more robust to a wider range of pushes.* ***Bottom:*** *Maximum force (in newtons) tolerance for controllers pushed in different directions. Each column represents a controller trained for a particular F.*

4.1 External disturbances

We begin with a scenario in which external forces are applied to the torso of the character, but with unknown timing and direction, such as gusts of wind. Other quantities are assumed known: the generation of internal joint torques is deterministic (2), there are no user inputs $\mathbf{c}$, and the start state $\mathbf{s}_1$ is optimized along with the control parameters.

Walking under random pushes to the chest. Consider a baseline controller optimized for walking in a fixed direction, efficiently and with a human-like speed to step-length ratio [Wang et al. 2009]. While such controllers produce gaits that appear loose and relaxed (e.g., Fig. 1a), they are not particularly robust. Strong pushes to the chest can easily make the character fall. Conversely, controllers optimized under uncertainty, where $p(\mathbf{e})$ represents random pushes to the chest, are significantly more robust. Further, the larger the magnitudes of the pushes during optimization, the more robust the resulting controller (see Fig. 3).

To model pushes to the chest, forces of magnitude F newtons from random directions parallel to the ground are applied to the torso COM. We learned controllers with $F = 100, 200, 300, 350$, and 400, all of which differ significantly from the baseline controller. The knee swings are less passive, making each step appear more deliberate. The arm swings are more pronounced, making the gaits appear more energetic. At $F = 200$, the upper body leans forward slightly, resulting in a fast gait (e.g., see Fig. 1b). At $F = 400$, the upper body is bent almost parallel to the ground, lowering the average COM from 1.02 m (baseline) to 0.96 m. We also find that, as F increases, the average squared torque over time (E_{power} in [Wang et al. 2009]), grows quickly from 34671 (baseline) to 194847 ($F = 400$).

Fig. 3(top) shows the success rate for different controllers under different force magnitudes during simulation, where a simulation trial is deemed successful if the character does not fall within 10 s.

Controllers optimized for larger forces are clearly more robust. Fig. 3(bottom) reports the maximum force tolerated by each controller from 8 directions. Following [Wang et al. 2009], we apply pushing forces to the torso once every 4 s. A controller succeeds if the character does not fall within 40 s.

Walking on a narrow beam. When walking on a narrow beam high above the ground (e.g., see Fig. 1d), the consequence of even a slight misstep can be catastrophic. If the environment is deterministic and known, one's gait on the beam might not differ from that used on the ground plane. In the presence of uncertain forces applied to the body, however, one must be more conservative to avoid taking a bad step and falling. To demonstrate this, we first optimized a baseline controller to walk on a narrow beam that is 0.5 m in width. We enable collision detection between body parts here, so that gaits with legs passing through each other are not possible. As expected, the resulting controller is like that for walking on the ground plane, except that the width of its gait is less than 0.5 m.

We then apply random forces to the torso, like those in the previous experiment, but with $F = 30$. Under pushes of this magnitude, the baseline controller quickly takes a wrong step and falls off the beam. Unlike the ground plane, where extending the width of a step in the sagittal or coronal direction can prevent falling even with large forces, here, just a light push is enough to cause the character to fall. Nevertheless, a successful controller for this environment can be learned through optimization. As depicted in Fig. 1d, the resulting controller takes smaller, more deliberate steps, and keeps the feet closer to center of the beam. The average step length decreased from 0.82 m (baseline) to 0.53 m, and E_{power} increased from 44998 to 100657.

4.2 Interactive user control

The control parameterization allows a user to specify heading (Sec. 3.1) and hence the walking direction. Nevertheless, the baseline straight-walking controller does not handle changes in heading successfully. To improve this, we view user input as a source of uncertainty, and optimize a controller to cope with random changes in desired heading direction. The optimized controller is slower (1.1 m/s) than the baseline controller optimized without turning (1.6 m/s). Unlike our results in the pushing experiment, this controller has the upper body leaning back slightly. We created an interface where the user changes the desired heading direction at will using the keyboard, 0.5 radians at a time. When used with the baseline controller, the character falls frequently. When optimized with random orientation changes, however, the user can easily learn to interactively navigate the 2D plane without the character falling.

We also use this task to examine the effect of N on the optimized solutions. We optimized 10 controllers with stochastic heading changes for each of $N = 1, 3, 5, 10, 20$. The results in Fig. 4 show that for small values of N there is higher sampling variability, and hence less robustness on average. As N increases, the average performance increases, as does the reliability of the controllers. Around $N = 10$ the marginal gain in controller performance decreases significantly compared to the added computational expense during optimization for this task.

4.3 Motor noise

We now consider the effects of motor noise together with different environments and return functions. In a deterministic setting, control optimization may succeed at challenging tasks with relative ease, even if there is little margin for error. In the presence of randomness, however, controllers must become more careful.

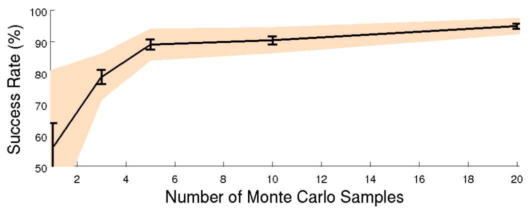


Figure 4: ***Effect of N on controller robustness.*** *We use the turning task as an example to show that optimization with CRN is effective with small N. To model $p(\mathbf{c})$, heading changes occur at each time instant with probability 0.05%, each of which is drawn from $\mathcal{N}(0, 0.5)$, a mean-zero Gaussian density with a standard deviation of 0.5 radians. To assess controller performance we use the fraction of 100 simulations that do not fall within 10 s. The curve shows the mean success rate for 10 controllers, at each value of N, with standard error bars. The orange region depicts the sample standard deviation for 10 controllers.*

Biological neural control systems exhibit noise. This seemingly random variability is found in the measurement of many biological quantities, even in highly repetitive tests [Faisal et al. 2008]. For example, all neurons, including motor neurons, exhibit variability in their output potentials even when the same stimuli are presented on repeated trials. Such neural noise is often signal dependent. In the motor system, larger control signals exhibit greater noise; i.e., in motor neurons that control muscle activation, the standard deviation scales in proportion to firing rates. There is strong evidence that motor noise plays a major role in determining human motor control strategies [Harris and Wolpert 1998]. Motor noise may also provide robustness under a wide range of otherwise unmodeled phenomena, including numerical errors in computer simulation.

We employ a simplified model in which motor noise affects the joint torques produced by the controller. Motivated by neural noise, we assume the standard deviation of the noise increases with torque, and decreases with the strength of the joint (i.e., the maximum torque that can be generated at that joint). The particular form of our model is a modified version of that developed by Hamilton et al. [2004]. For each joint i, given a desired (noiseless) torque of the controller $\bar{\tau}_i$, the noisy torque τ_i is drawn from a Gaussian density with mean $\bar{\tau}_i$ and standard deviation $\sigma(\bar{\tau}_i)$:

$$\tau_i \sim \mathcal{N}(\bar{\tau}_i; \sigma(\bar{\tau}_i)), \tag{6}$$

$$\sigma(\bar{\tau}_i) = \bar{\tau}_i \beta \exp(-2.76) MVT^{-0.25}, \tag{7}$$

where MVT is the maximum voluntary torque output at the particular joint and β is a scale factor that allows one to adjust the noise level. For each joint DOF, we use $MVT = k_p$, the spring stiffness constant of the PD-controller at that joint DOF.

Walking on a slippery surface. Walking on a surface with a low coefficient-of-friction μ requires caution. When optimizing a baseline controller for walking in a straight line on a surface with $\mu = 0.4$, with no noise or uncertainty in the environment, we obtain a controller that is somewhat more cautious, taking smaller steps than a comparable controller trained with greater friction. Nevertheless, more pronounced differences are evident when controllers are optimized with uncertainty due to motor noise. We optimized three controllers, each with a different amount of motor noise, by varying the scale factor in (7), i.e., $\beta = 50, 75, 100$. We find that, for high noise levels, the character raises his arms wide in the coronal plane and lowers his center of gravity, producing a gait much like that of a person treading carefully on ice (e.g., see Fig. 1c).

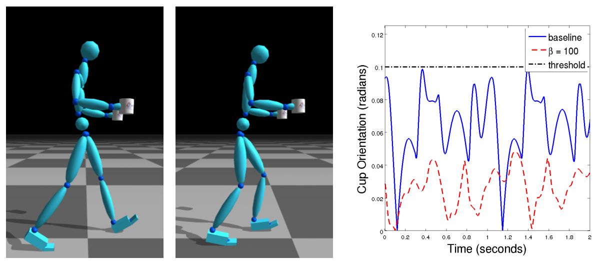


Figure 5: ***Carrying hot beverages. Left:*** *Poses from controllers optimized without and with motor noise ($\beta = 100$).* ***Right:*** *Mug orientation as a function of time from the two controllers, both simulated without motor noise.*

Carrying hot beverages. When designing a controller to carry a mug of hot coffee, we want to ensure the character will not spill the coffee. Accordingly, the controller receives large penalties (negative rewards) at every time-step for which the orientation of the mug deviates too far from vertical. Fig. 5(left) shows one pose of the baseline controller where the arms are not particularly stiff and the step length is relatively long. By comparison, a corresponding pose of the controller optimized with $\beta = 100$ shows a shorter step length with arms that remain steady and level. Fig. 5(right) plots the mug tilt as a function of time. The baseline controller walks in a relaxed fashion, but allows the mug orientation to reach the spillage threshold. Even small disturbances will therefore cause coffee to spill. In contrast, the strategy optimized with motor noise creates a margin for safety below the threshold.

To model this task, let $\mathbf{\Theta}_t = [\theta_{cor}, \theta_{sag}, \theta_{trans}]$ represent the orientation of a hand in the world frame at simulation step t, such that $\mathbf{\Theta}_t = [0, 0, 0]$ corresponds to a level, upright mug. Suppose that spills occur whenever $m(t) = \sqrt{{\theta_{cor}}^2 + {\theta_{sag}}^2} > 0.1$. Note that this model ignores acceleration of the mug, which could also be included for realism. For the baseline controller, optimized in a deterministic setting, the mug orientation quickly exceeds the 0.1 threshold, even for small amounts of motor noise (e.g., with $\beta = 25$). For a controller optimized with $\beta = 50$, the walking speed slows, and the cup orientation is more stable. For $\beta = 100$, the gait appears cautious and hip-driven (63% of the lower body power output is from the hip, compared to 43% in the baseline and $\beta = 50$ case). The latter two controllers, optimized to handle larger noise levels, are both able to walk without spillage when simulated with $\beta = 25$, unlike the baseline controller.

4.4 Recovery controllers

Optimization with random external perturbations, like that in Sec. 4.1, produces controllers that anticipate external disturbances. For example, they achieve robustness by remaining stiff and keeping the COM relatively low. A complementary approach is to design a reactive controller, where a basic controller π is active under normal circumstances, but a *recovery controller* π_r is invoked when a disturbance is detected. The recovery mechanism helps the character return to a normal gait so that π can be restarted. For example, one might walk with a relaxed gait until pushed, at which point the recovery controller takes over until the basic controller with the relaxed gait can resume control.

Ideally, π_r would bring the character to states with high expected return with respect to the basic controller π, but this is costly to evaluate. Instead, we aim for states that are typical of π. We define several key features $\hat{\mathbf{s}}$ of a character's state, and then estimate $p(\hat{\mathbf{s}})$, the distribution of features observed during normal walking under π. Here, $\hat{\mathbf{s}}$ comprises the horizontal distance from the stance ankle to the COM, and the COM velocity, projected into both the sagittal and coronal planes. Finally, we combine π and π_r to form a new reactive controller π_{new} as follows:

$$\pi_{new}(\mathbf{s}) = \begin{cases} \pi(\mathbf{s}) & p(\hat{\mathbf{s}}) > \kappa \\ \pi_r(\mathbf{s}) & \text{otherwise}, \end{cases} \tag{8}$$

where κ is a threshold. This controller runs π when the input is in π's typical states, and runs π_r otherwise.

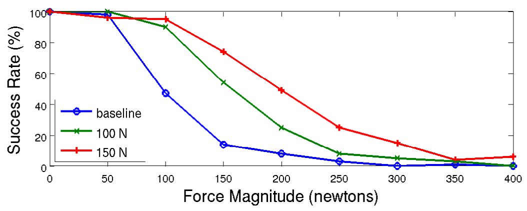
For the recovery task we begin with a controller π optimized to walk comfortably at 1.6 m/s, with step-length 0.8 m. We generate motions of duration 100 s from π with random heading changes (see Sec. 4.2) that occur with probability 0.025% at each time-step, drawn from $\mathcal{N}(0, 0.3)$. We then fit an axis-aligned Gaussian to these motion features to approximate $p(\hat{\mathbf{s}})$.

The goal of the controller π_{new} is to walk using π where possible, using π_r to return the character to states with high $p(\hat{\mathbf{s}})$ (i.e., typical states from π). We model random external forces using random pushes to the torso (as in Sec. 4.1), of 100 N and 150 N. The optimization variables are κ and the parameters of π_r, where π_r is initialized to π, κ is initialized to e^{-7}. The reward for the optimization penalizes time-steps not spent in the basic controller, and heavily penalizes falling:

$$r(\mathbf{s}_t) = -recover_t - failed_t\,, \tag{9}$$

where $recover_t$ is 0.01 if $p(\hat{\mathbf{s}}_t) < \kappa$, and 0 otherwise, and $failed_t$ is defined in the Appendix.

Following pushes, the recovery controllers successfully return the character to states with $p(\hat{\mathbf{s}}) > \kappa$ after a few steps at most, thus reactivating π. Adding the recovery controller improves robustness (Fig. 6), to a degree comparable to directly optimizing the basic controller, as in Sec. 4.1. However, direct optimization still provides larger improvements in robustness, since the entire walking style is allowed to be modified.



direction	baseline	rec. 100 N	rec.150 N
$(1, 0)$	50	125	200
$(1, \pm 1)$	50	75	175
$(0, \pm 1)$	150	175	175
$(-1, \pm 1)$	25	50	125
$(-1, 0)$	75	125	125

Figure 6: ***Robustness of recovery controllers. Top:*** *Recovery controllers optimized to return to the baseline controller (Sec. 4.4) for larger pushes show greater robustness. Success rate is estimated from 100 random trials.* ***Bottom:*** *Maximum disturbance force components (in newtons) tolerance for recovery controllers pushed in different directions.*

4.5 Transition between speeds

The ability to compose controllers is essential for characters to perform interesting activities in sequence. Previous work either assumes the existence of low-level controllers to or from which one can reliably transition, or has focused on identifying specific states where it is safe to switch [Faloutsos et al. 2001; Coros et al. 2009]. In general, between very different controllers (e.g., walking fast and slow), finding reliable switching states is difficult. We can however exploit the recovery controllers (8) above to facilitate such transitions.

More concretely, to facilitate transitions from controller π_A to controller π_B, we define a new controller

$$\pi_{AB}(\mathbf{s}) = \begin{cases} \pi_B(\mathbf{s}) & p_B(\hat{\mathbf{s}}) > \kappa_{AB} \\ \pi_{r,B}(\mathbf{s}) & \text{otherwise}\,, \end{cases} \tag{10}$$

where $p_B(\hat{\mathbf{s}})$ characterizes the key features of motions produced by π_B, and κ_{AB} is a threshold. When a command to switch from π_A to π_B is received, π_{AB} is activated. Since the states produced by π_A are not necessarily typical of π_B, i.e., $p_B(\hat{\mathbf{s}}) < \kappa_{AB}$, the transition will usually activate $\pi_{r,B}$ directly. The recovery controller $\pi_{r,B}$ will then be active until control under π_B can begin. To determine π_{AB}, like the recovery controller in (8), we optimize κ_{AB} and the parameters of $\pi_{r,B}$ to transition the character from a start state produced by π_A, to a state where π_B may be activated. To generate a range of possible start states for transitions, we simulate motions from π_A along with random switching times.

We demonstrate transition controllers to change speed while walking. First, we learned controllers for walking at 0.8, 1.6 and 2.4 m/s, and modeled their typical states with $p_B(\hat{\mathbf{s}})$ as above. For optimization and testing, we run each simulation for 7 s, with switching times drawn from a uniform distribution between 1 and 2 s. A transition is deemed successful if the character is still walking after 7 s. Unlike previous experiments, we use CRN with $N = 20$.

Without transition controllers, transitions from high to low speeds fail frequently. Out of 500 random trials, switching from 2.4 m/s to 1.6 m/s failed 203 times (40.6%), while switching from 2.4 m/s to 0.8 m/s failed 238 times (47.6%). The number of failures is reduced dramatically by the transition controllers, down to 7 (1.4%) and 16 (3.2%), respectively. Solely in terms of failure rates, transition controllers do not make much difference in other cases. For example, the failure rates for 0.8 m/s to 2.4 m/s and 1.6 m/s to 2.4 m/s were lowered from 5.4% to 2.2% and from 6.2% to 5.8%. However, they often still served to bring the character into a stable state more quickly and gracefully than direct switching, resulting in smoother transitions. Comparisons are included in the supplemental video.

4.6 Composing many controllers at run-time

The ability to transition from one controller to another facilitates the implementation of a character that can switch control strategies on demand. In the supplemental video, we demonstrate a character switching between several walking controllers, each optimized using methods described for fast speeds, slow speeds, a slippery surface, recovery, turning, and the high beam. Most of the switching did not require explicit transition controllers, since the controllers optimized with uncertainty are often robust enough as they are. All switching was determined by user commands within a single interactive session.

5 Discussion

We have presented a unified framework that captures many sources of uncertainty in a single optimization process. Our work shows the value of explicitly representing uncertainty in character controllers: control strategies automatically adapt to specific sources of randomness, making them more robust and composable, while creating natural stylistic variations. A main limitation of our method is that the quality of results still falls short of kinematic methods, and it could be argued that some of our adaptations appear unusual. While we have tested with a simple four-state PD controller, we believe these observations are general and should be useful for more sophisticated control parameterizations as well, but optimization may also become more difficult.

There remain other sources of uncertainty that could be handled with our model; each of these could lead to new forms of robustness and control strategies. Perhaps most significant is perceptual uncertainty [Körding 2007], namely, the incomplete picture of the world we get from our senses. For example, visual estimation of depth and motion are inherently noisy, a crucial fact for anyone attempting to catch or avoid a fast-moving object. Other sources of uncertainty include proprioceptive error (e.g., pose uncertainty), ground roughness [Byl and Tedrake 2009], and the behaviors of other agents.

Incorporating uncertainty into other approaches to optimization of character control should be straightforward. For example, many optimal control formulations used in recent animation research — including the Bellman equations, policy iteration, and the linear-quadratic regulator — can be formulated with probabilistic dynamics. However, restrictive dynamics and uncertainty models (e.g., linear dynamics and Gaussian noise) are normally required for optimal closed-form solutions in continuous models.

Acknowledgements

Thanks to Zoran Popović for early discussions, and the reviewers for their suggestions. This research is supported in part by NSERC, CFI, the Ontario MRI, and CIFAR. Part of this work was done while AH was on a sabbatical visit to Pixar Animation Studios.

References

ABBEEL, P., COATES, A., QUIGLEY, M., AND NG, A. Y. 2007. An application of reinforcement learning to aerobatic helicopter flight. In *Adv. NIPS 19*. MIT Press, 1–8.

BYL, K., AND TEDRAKE, R. 2009. Metastable walking machines. *Int. J. Rob. Res. 28*, 8, 1040–1064.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2009. Robust task-based control policies for physics-based characters. *ACM Trans. Graphics 28*, 5, 170.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive simulation of stylized human locomotion. *ACM Trans. Graphics 27*, 3, 82.

FAISAL, A. A., SELEN, L. P. J., AND WOLPERT, D. M. 2008. Noise in the nervous system. *Nature Rev. Neurosci. 9*, 292–303.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proc. SIGGRAPH*, ACM, 251–260.

GRZESZCZUK, R., AND TERZOPOULOS, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proc. SIGGRAPH*, ACM, 63–70.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. NeuroAnimator: Fast neural network emulation and control of physics-based models. In *Proc. SIGGRAPH*, ACM, 9–20.

HAMILTON, A. F. D. C., JONES, K. E., AND WOLPERT, D. M. 2004. The scaling of motor noise with muscle strength and motor unit in number in humans. *Exp. Brain Res. 157*, 4, 417–430.

HANSEN, N. 2006. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*. Springer, 75–102.

HARRIS, C. M., AND WOLPERT, D. M. 1998. Signal-dependent noise determines motor planning. *Nature 394*, 780–784.

HASLER, N., STOLL, C., SUNKEL, M., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. A statistical model of human pose and body shape. *Computer Graphics Forum 28*, 2, 337–346.

KÖRDING, K. 2007. Decision Theory: What "Should" the Nervous System Do? *Science 318*, 606–610.

LEE, J., AND LEE, K. H. 2006. Precomputing avatar behavior from human motion data. *Graphical Models 68*, 2, 158–174.

LEE, Y., LEE, S. J., AND POPOVIĆ, Z. 2009. Compact character controllers. *ACM Trans. Graphics 28*, 5, 169.

LO, W.-Y., AND ZWICKER, M. 2008. Real-time planning for parameterized human motion. In *Proc. Symposium on Computer Animation*, ACM SIGGRAPH/Eurographics, 29–38.

MCCANN, J., AND POLLARD, N. S. 2007. Responsive characters from motion fragments. *ACM Trans. Graphics 26*, 3, 6.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graphics 28*, 3, 81.

NG, A. Y., AND JORDAN, M. I. 2000. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proc. UAI*, AUAI, 406–415.

SHARON, D., AND VAN DE PANNE, M. 2005. Synthesis of controllers for stylized planar bipedal walking. In *Proc. ICRA*, IEEE, 2387–2392.

SIMS, K. 1994. Evolving virtual creatures. In *Proc. SIGGRAPH*, ACM, 15–22.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graphics 26*, 3, 107.

SPALL, J. C. 2003. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley.

SUTTON, R. S., AND BARTO, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

TEDRAKE, R., ZHANG, T. W., AND SEUNG, H. S. 2004. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proc. IROS*, vol. 3, IEEE/RSJ, 2849–2854.

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

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. *ACM Trans. Graphics 26*, 3, 7.

VAN DE PANNE, M., AND FIUME, E. 1993. Sensor-actuator networks. In *Proc. SIGGRAPH*, ACM, 335–342.

VAN DE PANNE, M., AND LAMOURET, A. 1995. Guided optimization for balanced locomotion. In *Proc. CAS*, EG, 165–177.

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

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Trans. Graphics 26*, 3, 105.

Appendix: Return function details

Our return function is based on the energy function of Wang et al. [2009]. We use the term return instead of energy, for consistency with the policy search literature. The complete return function, for motion $\mathbf{s}_{1:T}$, is the sum of *rewards* over time, plus terms E_{power}, E_{ratio} [Wang et al. 2009] to encourage power usage similar to human walking:

$$R(\mathbf{s}_{1:T}) = \left(\sum_t r(\mathbf{s}_t)\right) - (w_p E_{power} + w_r E_{ratio}) \ , \qquad (11)$$

where r is a scalar reward function of the current state, $w_p = 10^{-5}(200/mass), w_r = 5$.

In practice, there's no need to compute rewards at the frequency of the simulation. We set $r(\mathbf{s}_t) = 0$ except for when t is a multiple of 20, or when a state switch due to ground contact occurred at time t (the end of a stride). The reward is defined as the negative sum of a number of energy terms (i.e., $r(\mathbf{s}_t) = -\sum_i E_i$). The individual energy terms are defined by a thresholded quadratic penalty (i.e., $Q(d;\epsilon) = d^2$ if $|d| > \epsilon$, 0 otherwise).

A user may specify the desired average forward speed, v_x, or step length, s. This is done by penalizing differences between these quantities from the previous stride and user-specified targets ($\hat{v}_x$, $\hat{s}$, respectively):

$$E_{user} = Q(v_x - \hat{v}_x; 0.05) + Q(s - \hat{s}; 0.05) \, . \qquad (12)$$

Both of these terms are optional. If neither is specified, then we use

$$E_{step} = Q(s - \hat{s}; 0.05) \, , \quad \text{where } \hat{s} = l\left(v_x/\sqrt{gl}\right)^{0.42} \ , \qquad (13)$$

and l is the leg length of the character.

The target heading direction may be time-varying, with energy

$$E_{facing} = Q(v_y; 0.05) + 0.005 Q(\theta_d - \theta, 0.1) \, , \qquad (14)$$

where v_y is the average simulation velocity (in m/s) of the COM in the y direction in the current stride, θ_d, is the desired heading, θ is the current heading, and v_x, v_y are both computed by with respect to θ_d. Small angular momenta are also preferred:

$$E_{ang} = Q(L_x; 0.04) + Q(L_y; 0.05) + Q(L_z; 0.01) \, , \qquad (15)$$

where L_{xyz} are the maximum normalized angular momenta about the COM in the previous stride.

Let $\mathbf{\Phi}_t = [\phi_{cor}, \phi_{sag}, \phi_{trans}]$ represent the head orientation in the world frame at time step t. The objective to stabilize the head is

$$E_{head} = Q(v_{head}; 0.25) + h_t \, , \qquad (16)$$

where v_{head} is the maximum head velocity in the x direction during the previous stride, $h_t = 0.001$ when $\|\mathbf{\Phi}_t\|_2 > 0.1$.

Stable foot contact is rewarded by

$$E_{land} = stance_t + stubbed_t \, . \qquad (17)$$

If, during state 1 or 3 (toe-off), neither the stance toe nor the stance foot has 3 or more points of ground contact, then $stance_t = 0.001$; it is zero otherwise. We set $stubbed_t = 0.001$ at any time when the top of the swing toe is in contact with the ground.

Finally, we heavily penalize falling:

$$E_{fail} = failed_t \, , \qquad (18)$$

where for all states, if the COM falls below 0.7 m (i.e., the simulated character has fallen) then $failed_t = 100$, otherwise $failed_t = 0$. $E_{user}, E_{step}, E_{ang}, Q(v_y; 0.05), Q(v_{head}; 0.25)$ are all set to zero except for at the end of every stride.