Simulation of Human Motion Data using Short-Horizon Model-Predictive Control

by

Marco da Silva

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2008

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
February 1, 2008

Certified by .... [illegible] ....
Jovan Popović
Associate Professor
Thesis Supervisor

Accepted by .... [illegible] ....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Simulation of Human Motion Data using Short-Horizon Model-Predictive Control

by

Marco da Silva

Submitted to the Department of Electrical Engineering and Computer Science
on February 1, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Many data-driven animation techniques are capable of producing high quality motions of human characters. Few techniques, however, are capable of generating motions that are consistent with physically simulated environments. Physically simulated characters, in contrast, are automatically consistent with the environment, but their motions are often unnatural because they are difficult to control. We present a model-predictive controller that yields natural motions by guiding simulated humans toward real motion data. During simulation, the predictive component of the controller solves a quadratic program to compute the forces for a short window of time into the future. These forces are then applied by a low-gain proportional-derivative component, which makes minor adjustments until the next planning cycle. The controller is fast enough for interactive systems such as games and training simulations. It requires no precomputation and little manual tuning. The controller is resilient to mismatches between the character dynamics and the input motion, which allows it to track motion capture data even where the real dynamics are not known precisely. The same principled formulation can generate natural walks, runs, and jumps in a number of different physically simulated surroundings.

Thesis Supervisor: Jovan Popović
Title: Associate Professor

Acknowledgments

I would like to thank my research advisor, Jovan Popović, for his guidance and ideas. Yeuhi Abe was an invaluable source of insight and knowledge on this project. I would like to thank my family, Telmo, Cila, and Marta, for their constant support and encouragement. Most of all, I would like to thank my wife, Lisa. This project would not have been completed without her love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many data-driven animation techniques are capable of producing high quality motions of human characters. These approaches extend the usefulness of captured motions by allowing applications to adapt existing motions to meet different needs. Applications can create motions that satisfy new user constraints while maintaining the input motion style [2, 16] or exhibit new styles while preserving content [15]. Interactive applications such as games can respond to user input and synthesize new results in real-time.

Few techniques, however, are capable of generating motions that are consistent with physically simulated environments. The implicit assumption made by all kinematic synthesis approaches is that the performance environment is the same as the capture environment. This assumption is invalid when motions are performed in physically simulated environments. In a physical simulation, the character can encounter new or unpredictable circumstances such as being hit by a ball or standing on a shaky platform. Ignoring these interactions leads to physically inconsistent motion.

In contrast, physically simulated character motions are automatically consistent with the environment but are often unnatural because they are difficult to control. Recorded motions provide an intuitive control specification but simulating any such motion remains a difficult problem. Human characters, in particular, have many degrees of freedom (dofs) subject to non-smooth, non-linear dynamics. This makes it hard to find the forces that reproduce a desired motion, particularly in new environments.

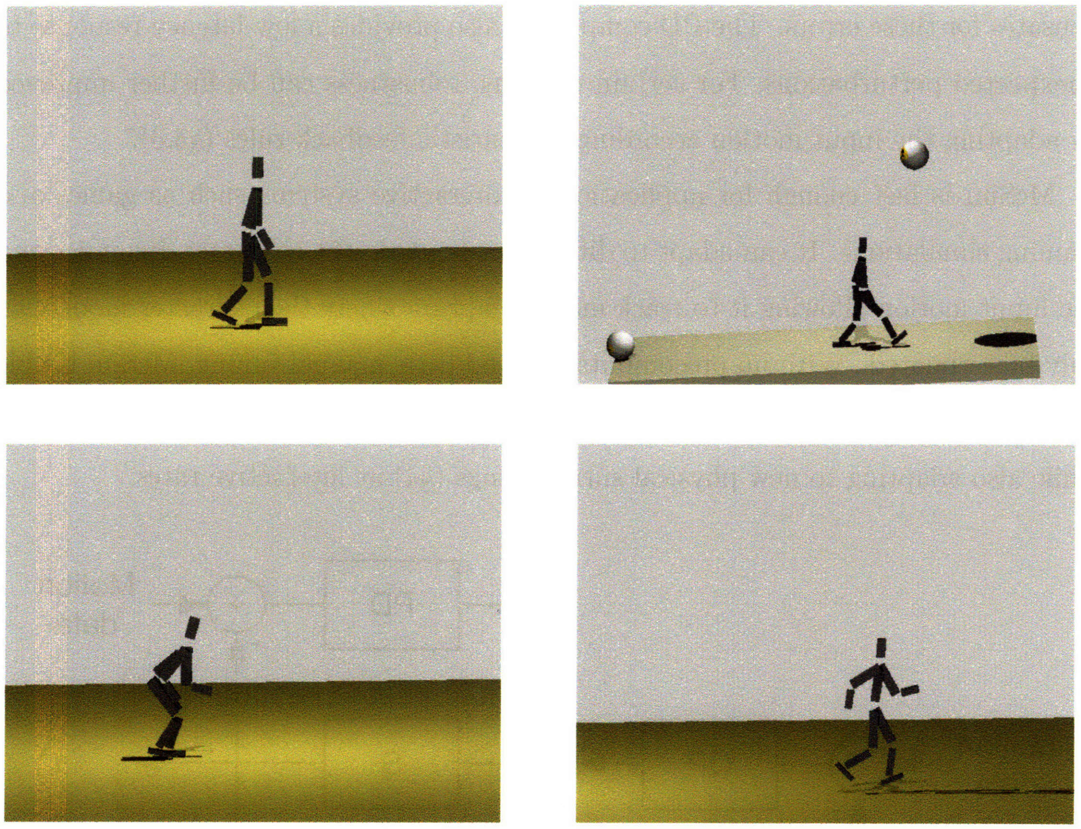


Figure 1-1: McSim is a controller for tracking reference motions in simulated environments. It is capable of tracking walking motions in different environments and styles. It can also track running and jumping.

We present a controller, McSim (motion capture in simulation), that yields natural motions by guiding simulated humans toward real motion data. McSim can be categorized as an instance of model-predictive control (MPC). In MPC, the controller predicts a control signal that achieves a desired change in system state based on the current system state and a model of the system's dynamics. Our controller uses a predictive component (§3.1) based on a linearized model of linked rigid body and contact dynamics. The linear dynamics model is used as a constraint in a quadratic program (QP) that solves for the joint and external forces that track the provided input motion for a short window of time into the future.

McSim combines the predictive component with a low gain proportional-derivative (PD) component (§3.2) as depicted in Figure 1-2. The predictive component's control has errors due to high latency and modeling assumptions. The PD component compensates for these errors. The PD component also provides a low-latency response to unexpected perturbations. For certain motions, robustness can be further improved by adapting the input motion according to heuristic feedback rules (§3.3).

McSim is fast enough for application in interactive systems such as games and training simulations. It can adapt to differences between the character dynamics and the input motion allowing it to track motion capture where the character model can only be estimated. With no precomputation and little manual tuning, McSim is able to produce walking, running, and jumping motions similar to the reference motion while also adapting to new physical surroundings (§4) at interactive rates.

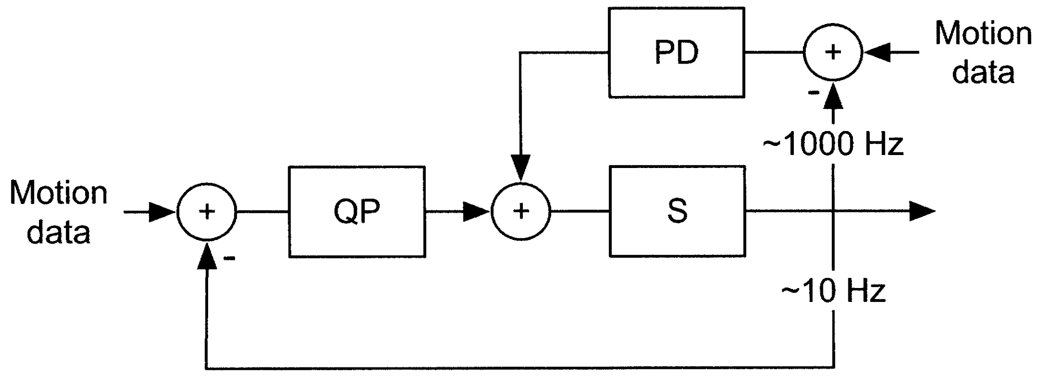


Figure 1-2: An overview of McSim's design. At 40 to 100 Hz, the predictive model solves a quadratic program for the joint and external forces that track the reference motion. These forces are adjusted by a low gain proportional derivative component that computes feedback forces at every simulation time step (1-10kHz). Our controller can work with any black-box simulator.

Chapter 2

Related Work

Most prior online control techniques in the graphics literature have been based on manually designed PD controllers [7,13,24,25,29,41]. These approaches are typically sensitive to gain parameters and not intuitively directed. In contrast, off-line authoring tools based on continuous optimization leverage the benefit of time to search for physical motions that are optimal according to some metric and satisfy user constraints [22,31,38]. The predictive component of McSim is inspired by these off-line approaches but sacrifices global optimality for computation speed by restricting the search to a short amount of time into the future. The predictive component of our controller allows the PD component to use relatively small gain parameters, resulting in more stable simulations and more natural motion [40].

McSim can be guided by an arbitrary input motion. Recently, both off-line and online physically based character animation have used data to produce life-like animations, though the role of data differs for each approach. Since the goal of an off-line approach is to produce a new motion with new content, data is used to restrict the search space of possible solutions [26], to model simplified equations of motion [4,33], and to learn parameters of motion style [19]. In online control, the goal is often to simply track a provided input motion in a dynamically simulated environment [42]. Recent approaches, however, have been limited to special cases of motion such as cyclic motions [41] or standing [1, 42]. Our approach can track arbitrary motions exhibiting stylistic variations and transitions such as walking to standing.

Many recent approaches to tracking motion data find globally optimal control policies using off-line precomputation methods such as feedback error learning or simplex methods [28, 29, 34, 41]. However, global optimality is only guaranteed if the playback environment does not differ significantly from the environment simulated during optimization. Furthermore, these global search methods are not easily applicable to 3D animation where the number of dofs is large. While McSim could incorporate a precomputed feedforward control signal, it produces plausible motions without precomputation. This enables it to be coupled with kinematic motion synthesis techniques [20, 21] to track newly created 2D or 3D motions at run-time.

Among instantaneous optimization approaches, McSim is closely related to Multiobjective Control [1]. McSim adds the ability to track motions where the contact state changes regularly as in locomotion. Furthermore, we illustrate how the input motion can be modified to improve tracking performance. There are many previous approaches from robotics that propose some form of optimization over a short time horizon to achieve a motion objective [10, 14, 36, 37], each with key differences in the details. In this work, we propose an alternative formulation of the tracking problem that is capable of handling arbitrary motions and couple it with robust low-latency feedback mechanisms. Others have argued that this form of control is employed by biological systems [40].

Chapter 3

Controller Design

McSim's design is guided by three goals. The output motion should be directed by specifying any input motion. It needs to work at interactive rates without requiring expensive precomputation. Finally, it has to work with existing black-box simulators. We would like our controller to work as a plug-in module with any simulator without any modification to the simulator itself. Achieving these objectives would make the system suitable for tracking kinematically specified motions in interactive applications such as games and training simulations. In the following sections, we describe how McSim achieves these three goals.

At each time step, t, McSim computes a control signal of the form:

$$u(t, x, x_r(t)) = u_f(t, x, x_r(t)) + u_b(t, x, x_r(t)) \tag{3.1}$$

where u is the control signal, x is the current system state consisting of joint values and velocities, $[q, \dot{q}]$, and x_r is the desired state. The total control signal consists of the predictive component's signal, u_f, added to the PD component's signal, u_b. A predictive dynamics model computes u_f. The PD controller computes u_b which provides low-latency feedback to deal with unexpected perturbations. Stability is achieved by tracking the velocity of the root of the character and modifying the reference motion, x_r as described in Section 3.3.

3.1 The Predictive Component

The predictive component's task is to track the reference motion, $x_r(t)$. A long-horizon approach to tracking the reference motion would solve a single optimization for the control forces exerted over the entire motion [38]. For human motions, this form of tracking is a high-dimensional, non-linear, non-convex minimization problem. This makes an exact solution impractical at interactive rates. Furthermore, in interactive applications, long-horizon optimal plans are quickly invalidated by changes in the dynamic environment. Rather than plan optimally for situations that may never come to pass, we plan a over a small interval into the future using a linearized dynamics model and re-plan at regular intervals, incorporating changes in system state. We call this form of the problem, short-horizon tracking.

3.1.1 Short-Horizon Tracking

In a physical simulation, a character's motion is determined by integrating a dynamical system forward in time from some initial configuration,

$$x(T) = x(0) + \int_0^T \dot{x}(t)\, dt. \tag{3.2}$$

For an active character modeled as a system of linked rigid bodies with actuators between each joint, the equations of motion depend on the current state, $x(t)$, the control signal, $u(t)$, and the external forces, $u_c(t)$. The precise equations can be derived from classical mechanics [9] but are summarized here as

$$\dot{x}(t) = f(x(t), u(t) + u_c(t)). \tag{3.3}$$

A motion that perfectly tracks the reference satisfies $\ddot{q}(t) = \ddot{q}_r(t)$ for all t, where $\ddot{q}_r$ is the acceleration of the reference motion. The predictive component computes a u_f that tries to reproduce the reference acceleration over a window of size h. In practice, it is usually not possible to achieve the reference acceleration, $\ddot{q}_r$, exactly due to dynamics constraints of the character and environmental disturbances. As a result, the simulated motion will drift from the reference motion. To correct this drift, feedback terms are added to the reference acceleration to form the desired acceleration, $\ddot{q}_d$, as described in the next section. Once the desired acceleration is known, a constrained optimization is solved for the joint torques and external forces that achieve it.

3.1.2 The Desired Acceleration

The desired acceleration consists of the reference acceleration and a correction term. It is computed separately for each joint.

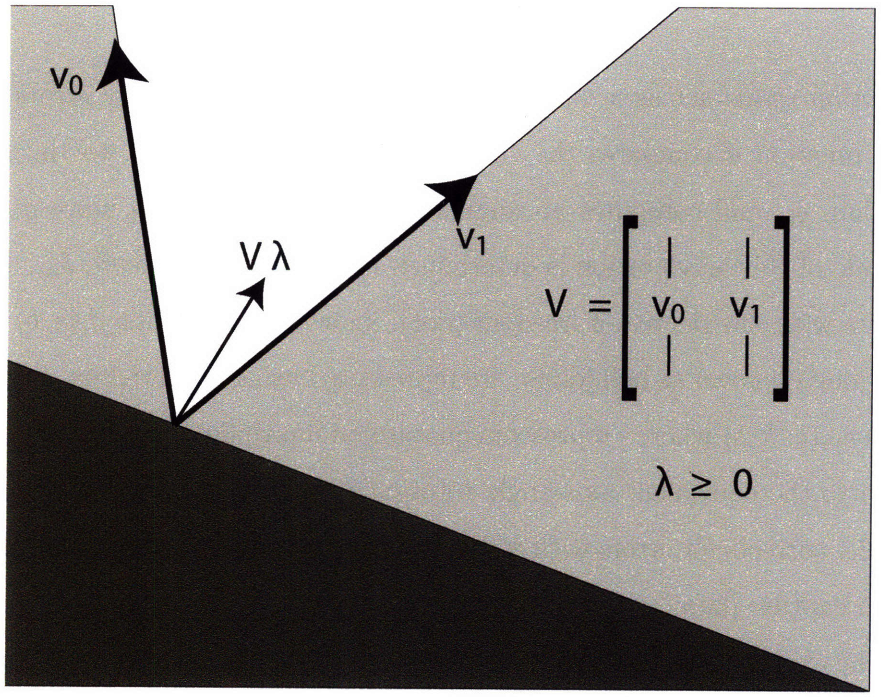
$$\ddot{q}_d = \ddot{q}_r + k_{os} d(q_r, q) + k_{od}(\dot{q}_r - \dot{q}). \tag{3.4}$$

The correction terms act as a damped feedback acceleration on any errors that occur. The function d compares the current joint configuration, q, to the reference configuration, q_r, and computes an angular acceleration that will move q closer to q_r. The scale of this acceleration is determined by the gain parameter, k_{os}. For rotational joints with one degree of freedom (dof), known as pin joints, $d_i(a, b) = a - b$. Three dof joints, known as ball joints, are represented using quaternions. In this case, $d_i(a, b) = vec_i(a^{-1} \cdot b)$ where $\cdot$ represents quaternion multiplication and vec_i maps the quaternion to the equivalent axis-angle rotation's i'th component. The last term in Equation 3.4 corrects for errors with respect to the reference velocity $\dot{q}_r$ obtained from the motion capture data.

With the exception of the root translation, all desired accelerations are computed using the same values of k_{os} and k_{od}. If $k_{os} = c$, then $k_{od} = 2\sqrt{c}$. Errors in the current position of the root are ignored when computing the desired acceleration of the root. Thus, for the root translation, $k_{os} = 0$. This prevents the controller from trying to correct for errors that are unavoidable due to the environment such as the character walking down hill. The velocity gain is not zero, however. This feedback uses the same gain as the other joints, $k_{od} = 2\sqrt{c}$. The controller is fairly insensitive to the particular value of c chosen as shown in section 4.

3.1.3 Dynamics Constraints

Computing the control input u needed to achieve the desired acceleration just described would be easy if we could simply invert Equation 3.3. Unfortunately, humans and animals have more degrees of freedom than forces to control them. Simple inverse dynamics algorithms such as those used for robotic arms rely on being rooted to the environment. Humans, however, are not rooted to the ground. They can use their feet to push, but not pull, on the ground. They must manipulate these unilateral contact constraints while respecting frictional limits to effect their overall motion [1,36]. These contact constraints are a key component of the dynamic model used by the predictive component.



v_0
$V\lambda$
v_1
$V = \begin{bmatrix} | & | \\ v_0 & v_1 \\ | & | \end{bmatrix}$
$\lambda \geq 0$

Figure 3-1: A friction cone in 2D. Legal contact forces lie within the cone which can be represented using a linear basis. Non-negative combinations of the basis vectors yield forces in the cone.

Contact forces are computed using a polygonal approximation to Coulomb's model of friction [8]. The model is depicted in 2D in Figure 3-1. Legal contact forces lie within a friction cone at each corner of the foot in contact with the ground. The cone is oriented normal to the contacting surface with a swept angle determined by the coefficient of friction. In 3D, we use a polygonal approximation (4 facets) to this cone which can be described with a linear basis, V. Contact forces are equal to $V\lambda$ with $\lambda \geq 0$. The non-negative bound on λ insures that the ground reaction force resides within the approximation to the friction cone and prevents contacting bodies from pulling on each other. The i'th contact force induces generalized torques on the character which are calculated as $J_i^T V_i \lambda_i$ where J is the gradient of the contact point with respect to the joint configuration of the character. The total contact force on the character, then, is $u_c = \sum_i J_i^T V_i \lambda_i$.

For this static contact model to hold, the contact forces must act only on contact points with zero acceleration [3]. This is known as the no-slip condition:

$$J_i \ddot{q} + \dot{J}_i \dot{q} = 0. \tag{3.5}$$

In addition to constraints on possible contact forces, achievable accelerations are constrained by the dynamics of the character. Since the predictive component plans over a short-horizon, the dynamics of the character are described by a linear relationship between applied forces and resulting accelerations:

$$\ddot{q}(t) = f(q(t), \dot{q}(t), 0) + W(u + u_c) \tag{3.6}$$

where W is the gradient of f with respect to the control input. Note that the internal torques, u, are limited by bounds on the strength of the character's actuators, $u_f \in U$, further restricting possible accelerations.

3.1.4 Quadratic Programming Optimization

Given all of these constraints, we can now formulate an optimization problem that solves for the joint and external forces, u_f and $J_i^T V_i \lambda_i$, that best achieve the desired acceleration, $\ddot{q}_d$:

$$\min_{u_f,\lambda_i} \quad \frac{1}{2}||\ddot{q} - \ddot{q}_d||^2 \tag{3.7a}$$

$$\text{subject to} \quad \lambda_i \geq 0 \tag{3.7b}$$

$$u_f \in U \tag{3.7c}$$

$$\ddot{q} = f(q, \dot{q}, 0) + W(u_f + \sum_i J_i^T V_i \lambda_i) \tag{3.7d}$$

$$J_i \ddot{q} + \dot{J}_i \dot{q} = 0. \tag{3.7e}$$

The predicted acceleration of the character is $\ddot{q}$. The objective penalizes accelerations different than the desired acceleration, $\ddot{q}_d$, which was chosen to track the reference motion. This minimization problem can be solved efficiently: it features a quadratic objective with a positive-semidefinite Hessian, and the constraints are linear. This yields a convex quadratic programming (QP) problem. The QP is solved at a much slower rate than the simulation. At time steps where it is not solved, the previously calculated forces are used.

3.2 Proportional-Derivative Component

Solving the QP in the predictive component is fast but not immediate. The drawback of this latency is that the predictive component cannot adapt to disturbances in between updates to its control signal. We resolve this problem with a PD control that adjusts the QP solution at each simulation step. The PD control guides the character through contact transitions and provides immediate responses to disturbances.

McSim's PD component computes u_b in Equation 3.1 at each step of the simulation. It is implemented using a critically-damped proportional-derivative (PD) controller [25]. The form of this control varies according to the particular joint. Since the root joint of the character is unactuated, no feedback forces are computed for the root dof's. Pin joints are computed using a standard critically damped feedback law

$$u_b = k_s(q_r - q) - 2\sqrt{k_s}\dot{q}. \tag{3.8}$$

To compute the feedback forces of a ball joint, we first compute the composite rotational inertia of all of its child links in world coordinates:

$$I_{c,j} = \sum_{l \in c(j) \cup j} R_l I_l R_l^T. \tag{3.9}$$

The resulting feedback force is then computed as

$$u_b = k_s I_c d(q_r, q) + 2\sqrt{k_s I_c}(\dot{q}_r - \dot{q}). \tag{3.10}$$

Note that the term $\sqrt{k_s I_c}$ means taking the square root of each element of the matrix $k_s I_c$. Multiplying by the world-space inertia matrix insures that the feedback force is scaled by the appropriate amount relative to the actual current distribution of mass supported by the joint. The resulting force, u_b, is added to the current predictive force u_f to give the total force at each time step.

3.3 Maintaining Balance

McSim maintains balance by tracking the input motion with forces that are consistent with the current contact environment. Other works employ a similar approach by using formulations specific to static contact [42] or infinite friction and planar contacts [14, 17, 35]. In contrast, McSim uses a model of contact dynamics that can account for more general geometric and frictional properties of the contacting surfaces [1, 36].

In certain cases, heuristic methods can adapt the input motion directly to improve tracking robustness. For example, one could track a parameterized family of motions rather than a single motion [37] or adapt the center of mass motion through a feedback [1]. For some of the 2D motions presented in the results section, we employed a feedback scheme similar to the heuristic used in the SIMBICON system [41]:

$$\theta_d = \theta_{d0} + c_d d + c_v v \tag{3.11}$$

where θ_d is the desired angle of the swing hip, θ_{d0} is the value of the swing hip in the reference motion, d is the horizontal distance between the root link and the support foot, and v is the horizontal velocity of the root link. Contrary, to SIMBICON's approach, we do not change the gains, c_d and c_v, with changes in contact state. They are fixed for a particular motion.

McSim is largely insensitive to the particular choice of the gains. Normally, McSim tracks the input motion even when the gains are set to zero. However, adding this form of balance feedback improved the robustness of a character walking on a moving platform and allowed the controller to track a run cycle indefinitely. A drawback to using this particular form of balance feedback, however, is that it is specific to walking and running motions. Similar methods of adapting the input motion have been applied to other motions [39].

Chapter 4

Results

McSim produces life-like character motion similar to a provided input motion. In the following section we highlight results that demonstrate McSim's ability to adapt motion capture data to new physical environments and track a variety of input motions. We also explore the sensitivity of the approach to various modeling errors and discuss the quality of the results. Finally, we provide implementation details. Please refer to the video accompanying this paper for the animations corresponding to the presented results [6].

4.1 New Environments

An exciting application of McSim is adapting motion data to new physical environments. This is illustrated in several demos presented in the accompanying video [6]. For example, a motion recorded on flat ground can be adapted to walk up or down an inclined ground plane. In our experiments, successful walks were created for uphill slopes as large as five degrees and downhill slopes as large as 10 degrees. Simple kinematic playback of the motion would walk through the ground or into the air.

In a physical simulation, the environment can change dynamically and a character must react to maintain plausibility. Our controller allows motion data to adapt to its environment. We generated several animations where the character is perturbed by flying balls or obstructed by blocks (Figure 4-1). The ground too can evolve

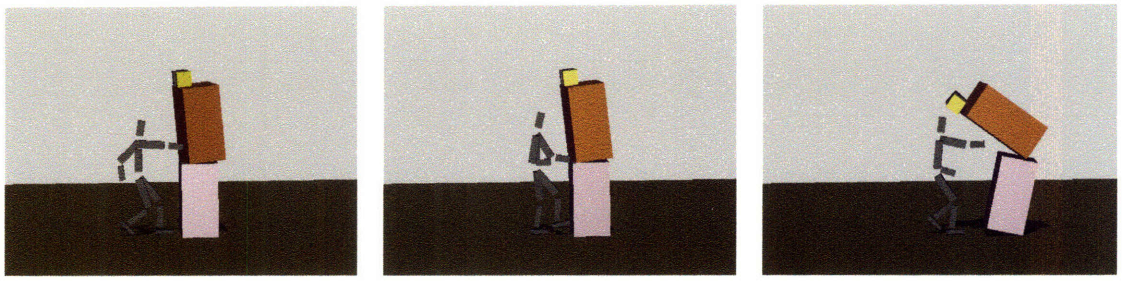


Figure 4-1: A sequence of frames from a walking sequence. The reference motion is a person marching in a military style. In this simulation, the character punches through some heavy blocks blocking the path.

dynamically as evidenced by simulations of the character walking over a moving platform and a see-saw (Figure 4-2).

4.2 Tracking

McSim is capable of tracking a wide range of motions in 2D and 3D. The video accompanying this paper presents results of tracking walking, running, and jumping motions [6]. These motions exhibit stylistic variations and transitions between modes such as from standing to walking and walking to standing.

A key feature of McSim is that there are few parameters that require tuning. To generate the results, two parameters were tuned manually: the optimal feedback gain used in Equation 3.4 and a scale factor on the intrinsic joint stiffness parameters used in Equations 3.8 and 3.9. In most cases, it was not difficult to find a satisfactory setting of these parameters as a large range of values led to satisfactory results as explained in Table A.1. Even across different types of motion, identical parameter values lead to good results.

Though McSim does not satisfy any global optimality criterion, it achieves good tracking results in practice. In the absence of large disturbances to the physical system or large errors in the physical character model, McSim will succeed in tracking the input motion. The plots in Figure 4-4 depict the squared tracking error (squared Euclidean distance between the actual state vector and the desired state vector) over time for selected motions. The plots illustrate several interesting features of the

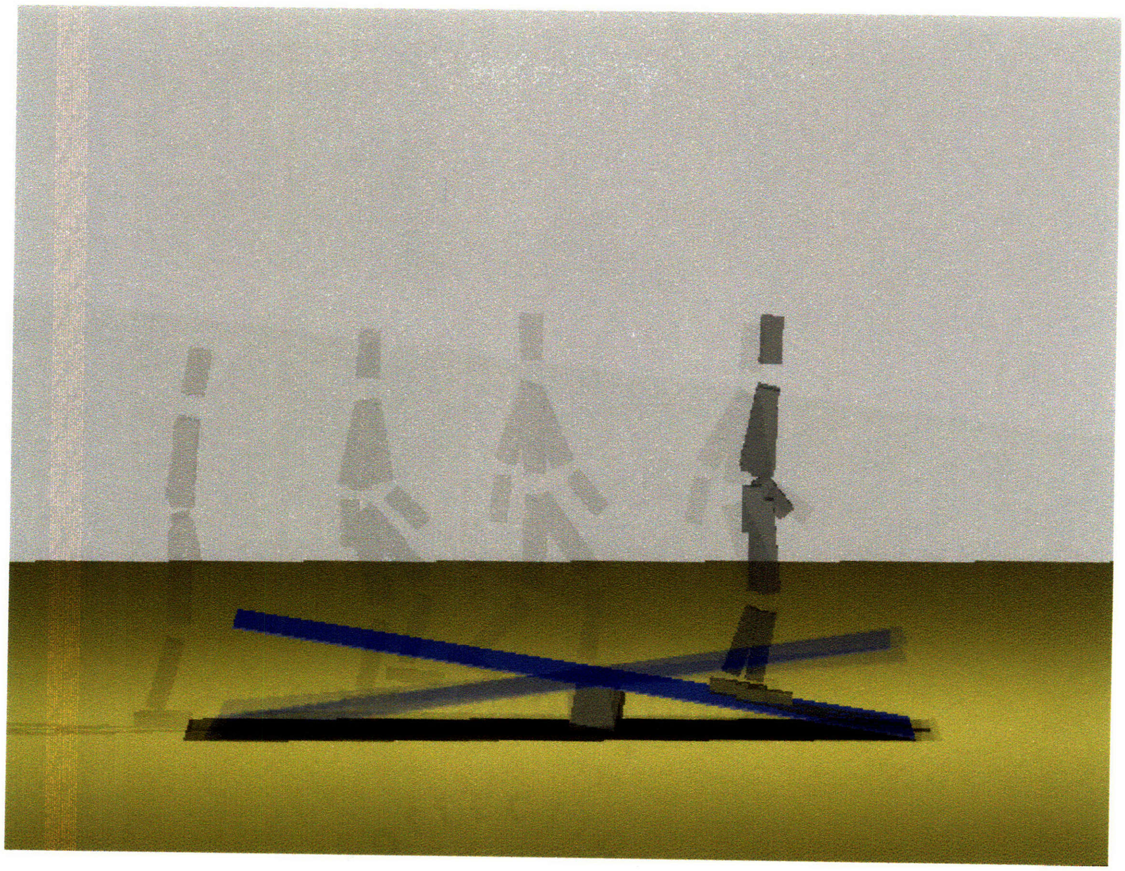


Figure 4-2: A time-lapse image of a walking sequence. The reference motion was captured from a person walking on flat ground. Our controller was able to adapt this motion to walk over a moving see-saw.

tracking system. First, the beginning of the walk motion is a period of standing. The system has little trouble tracking this portion of the motion. More energetic motions lead to more error. The spikes in the error curves coincide with changes in contact state suggesting that the predictive model could be improved by accounting for mismatches in the current contact state and the contact state in the reference motion.

4.3 Modeling Errors

The tracking quality of McSim is adversely effected by physical mismatches between the character model and the capture subject. To explore the effect of modeling errors we introduce various modeling changes and measure the change in tracking

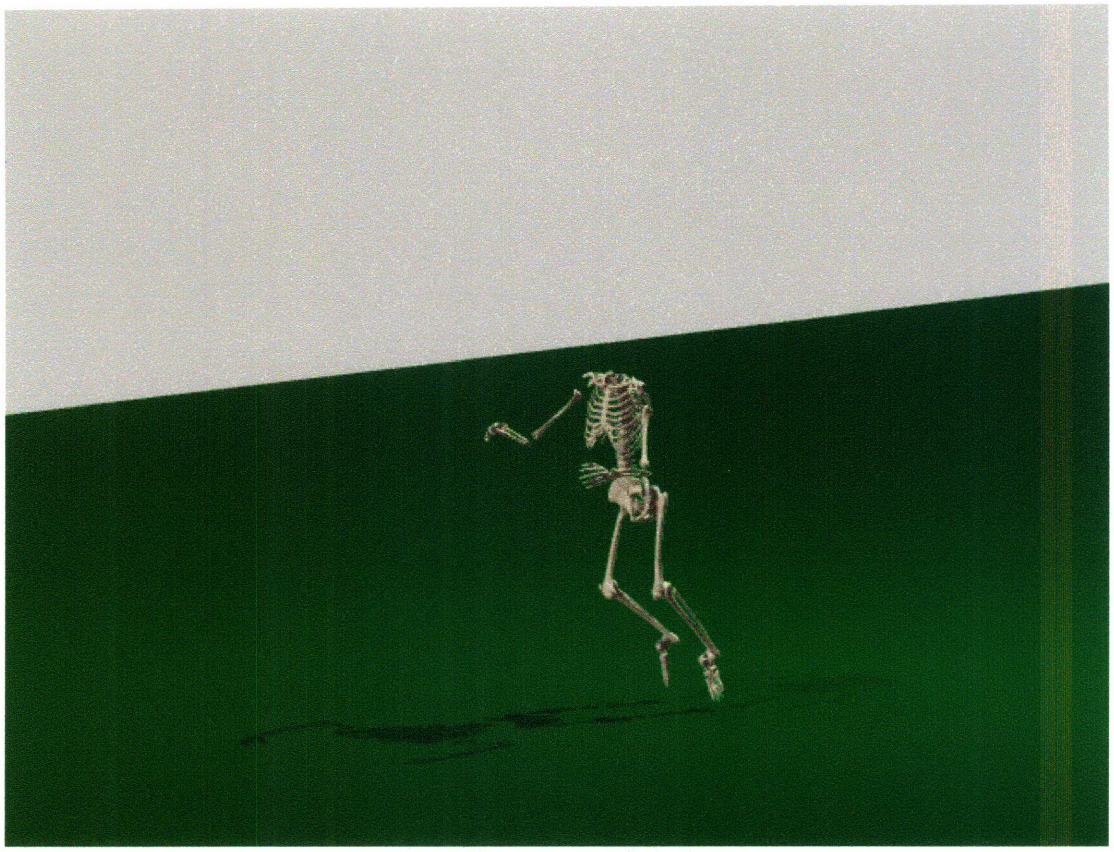


Figure 4-3: A rendering from a frame of a jumping sequence in 3D. McSim was able to track the character through three consecutive jumps.

performance.

4.3.1 Inertial Modeling Errors

One potential source of error in tracking motion capture data is an incorrect physical model of the subject. The mass distribution and inertial properties are often based on statistical models that are often quite different than the actual properties of the recorded subject. This mismatch can make an input motion physically infeasible for the character. To illustrate the sensitivity to errors in mass distribution, we plot the squared error for different versions of the 2D model for the walking motion in Figure 4-5. The mass of the character was redistributed to create three new versions of the original. One version of the character has a left leg that is twice as heavy as the right leg. In the next version, the upper body's mass is doubled while the lower body mass

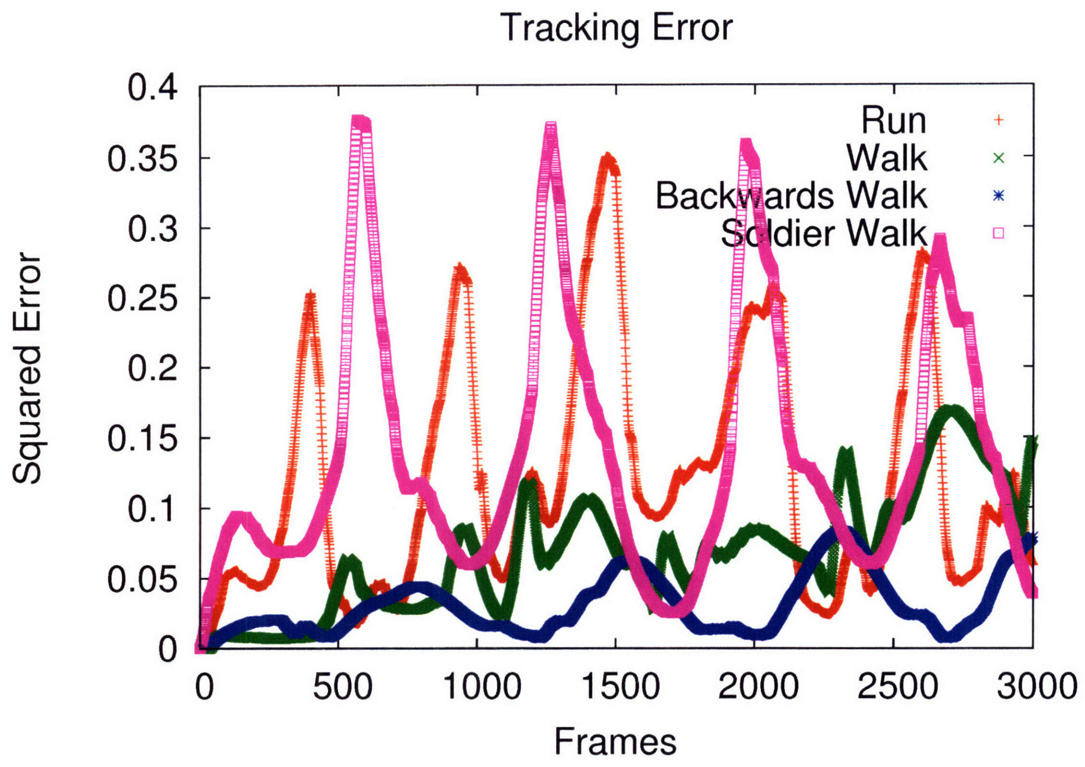


Figure 4-4: Shown are plots of squared error over time for four selected motions. The plots illustrate several interesting features of the tracking system. First, the beginning of the walk motion is a period of standing. The system has little trouble tracking this portion of the motion. More energetic motions lead to more error. The spikes in the error curves coincide with changes in contact state suggesting that the predictive model could be improved by accounting for mismatches in the current contact state and the contact state in the reference motion.

is cut in half. Finally, we double the mass of both legs. For walking motions, McSim is more sensitive to errors in the mass properties of the legs.

4.3.2 Contact Modeling Errors

Contact geometry was modeled using four small spheres placed at the corners of each foot. The controller is somewhat insensitive to the simulator's contact dynamics. To illustrate this, we compared the performance of the controller on a walking motion with varying coefficients of friction in 4-6. Tracking performance was not greatly effected. Contact dynamics were approximated using a friction cone model with a coefficient of friction ranging from 0.75 to 2.0 or stiff springs as in [29].

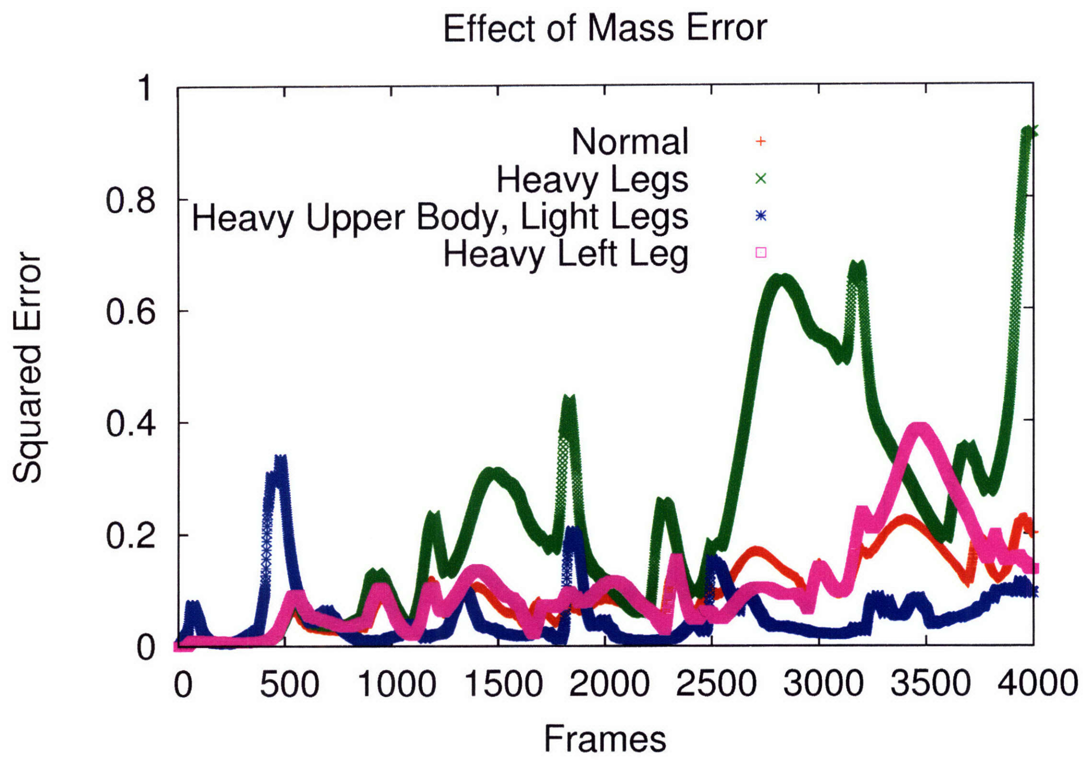


Figure 4-5: Shown are plots of squared error over time for four versions of the 2D model tracking the walking motion. The modifications are described in the legend. McSim is more sensitive to errors in the mass properties of the legs.

4.3.3 Foot Modeling Errors

The feet present another difficulty when tracking motion capture data. Our motion capture data for the ankle is fairly inaccurate. We offset the ankle angle by a constant so that the character's contact points are flush with the ground while standing. To get a feel for how sensitive McSim is to variations in foot geometry, we varied the foot size of the 2D model and plotted the results in Figure 4-7. The big feet were 4 centimeters larger than the standard feet used in most of the results in this work. The small feet were 4 centimeters shorter than the standard while the smallest feet were 8 centimeters shorter. The results indicate that McSim is robust to small discrepancies in foot size.

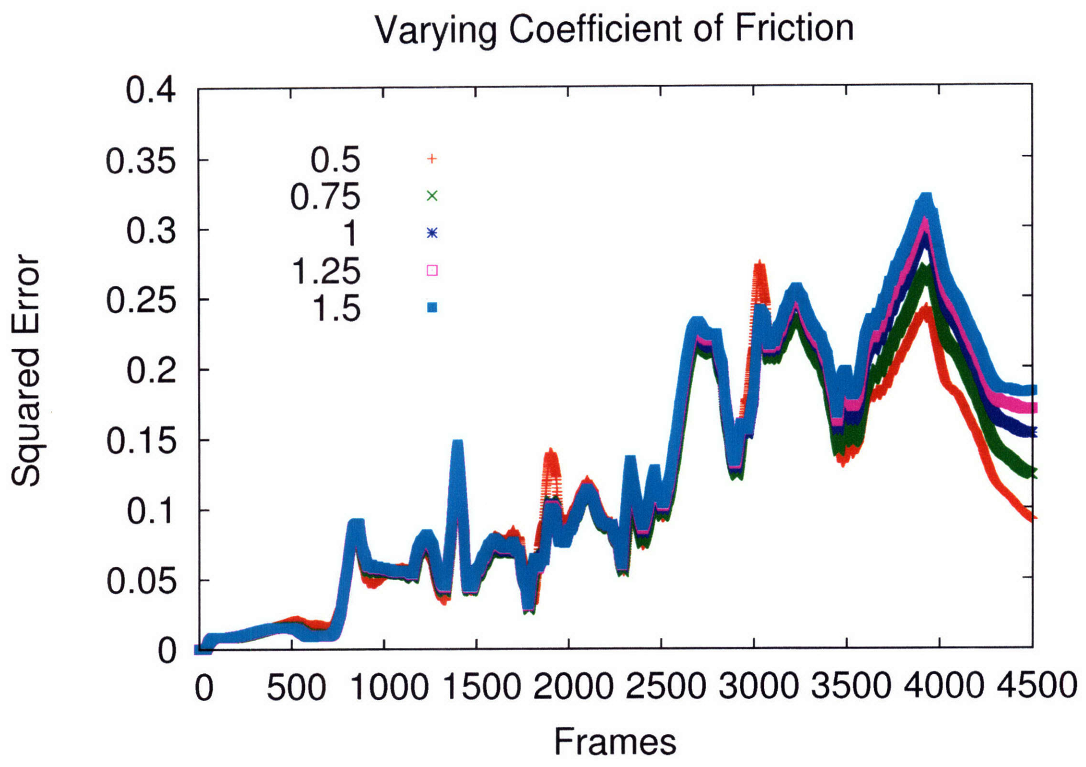


Figure 4-6: In these plots, the squared error is shown for a walking motion where the coefficient of friction in the predictive component is varied from 0.5 to 1.5. The simulator's coefficient of friction was fixed at 1. For walking motions, the error is not greatly effected by the coefficient of friction used in the model. When the predictive model's coefficient of friction exceeds the actual coefficient of friction, performance is worse, but only slightly.

4.3.4 Motion Quality

The results of McSim's tracking often look robotic and abrupt. For example, the 3D marching motion makes hard contacts with the ground that are not present in the reference motion. The 2D walk uphill sways a bit unnaturally as well. There are a couple of factors that affect the quality of the results. The first is that the short-horizon approach to tracking is a greedy approach. It applies large torques to try and immediately cancel any errors. These large forces can lead to unnatural accelerations and motion. The other factor effecting quality is the fact that gain parameters are manually set by hand. This was more of an issue for the 3D examples which were more sensitive to the gain parameter settings.

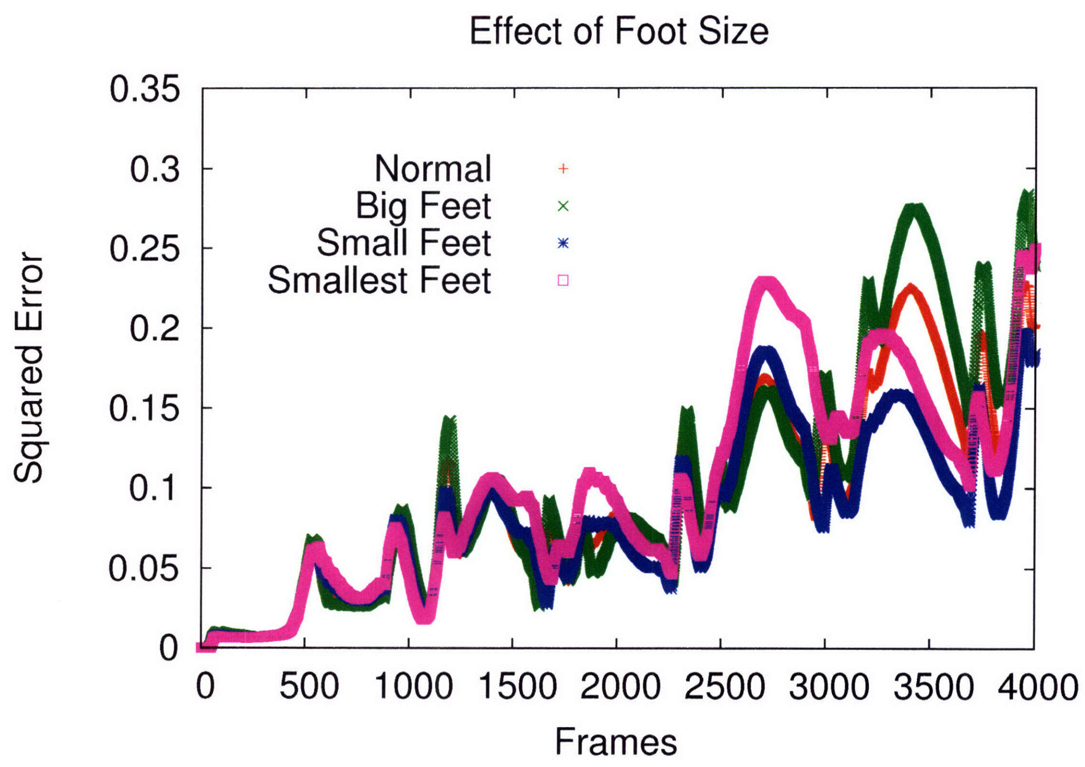


Figure 4-7: Shown are plots of squared error over time for versions of the 2D model with different sized feet. The big feet were 4 centimeters larger than the standard feet. The small feet are 4 centimeters smaller than the standard while the smallest feet are 8 centimeters smaller. These plots show that, at least for walking, slightly better results can be achieved by shrinking the foot. This might suggest that the actor performing the motion had slightly smaller feet. However, the results indicate that McSim is robust to small discrepancies in foot size.

4.4 Experimental Setup

The motion data for this work came from two sources. The 2D examples were downloaded from `http://mrl.snu.ac.kr/research/ProjectSimulBiped/SimulBiped.html`. This data was converted to 2D from motion capture data as described in [29]. The 3D data was captured and processed using a standard motion capture system.

A prerequisite of simulating character motion is a physical model of the inertial and stiffness properties of the character's limbs and joints. A good model is important as significant errors make the input motion physically infeasible for the model. For the 2D examples, the physical model (see Figure 4-8) has the same properties as the

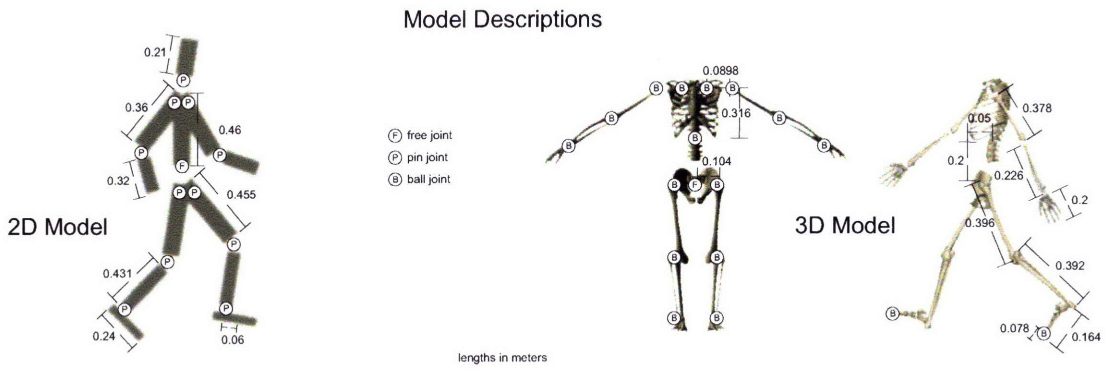


Figure 4-8: The models. A free joint has six degrees of freedom and is represented by a position and a quaternion. A pin joint has one degree of freedom and is represented by an angle of rotation. The center of mass of each link is located at the center of the link. Inertial and joint stiffness properties are listed in Table A.2 and A.3.

one used in [29]. The root link, however, is three dimensional. Its state is represented with a position vector, an orientation quaternion, and linear and angular velocity. The resulting model has 18 dofs. The inertial properties and joint stiffness parameters are presented in Table A.2. These stiffness parameters are first scaled uniformly by a gain factor that is smaller than one and then used as the PD gains in 3.8 and 3.9. The 3D model has 57 dofs. The parameters for the 3D model are presented in Table A.3.

The simulations were executed in DANCE [27] using the Open Dynamics Engine (ODE) as the simulator. The step size was 1 ms for the 2D examples and 0.1ms for the 3D examples. We use a smaller step size for 3D examples as ODE was unstable with larger step sizes. A simulator using an implicit or semi-implicit integration scheme could presumably use a larger step size.

The controller implementation sets up the QP problem described in section 3.1 using the current contact state from the simulation. It uses our C++ implementation of recursive dynamics equations [9] to compute various dynamical quantities needed for the optimization such as the inertial matrix of the system and gravitational and centrifugal forces on the system. The QP is solved using SQOPT [12]. Timings for the QP solver on a Pentium 4 2.8 Ghz processor are presented in Table A.4. The code for the PD component took roughly 0.4 ms on the 3D character and 0.05 ms on the 2D character.

Chapter 5

Conclusions

Motion data is an intuitive way to direct the actions of a physically simulated character. Determining the forces that track the motion faithfully while respecting physical and environmental constraints is a difficult problem. The controller presented in this work, McSim, offers advantages over previous kinematic and physical approaches to the problem of tracking motion data. First, McSim finds forces that track motion data while remaining physically consistent. Also, McSim requires less manual tuning than previous controllers. Finally, McSim can run at interactive rates making it suitable for the control of characters in interactive applications such as games and training simulations. McSim has limitations with respect to quality and robustness which point to interesting directions of future work.

Physically Consistent McSim produces a motion that is physically consistent with the environment. The reference motion provided to McSim may have been captured in a physical setting that differs from the simulated setting. In the simulated world, kinematic playback of the reference motion may hover over the ground or penetrate it. A kinematically driven character is unnaturally stiff and does not react to external disturbances. McSim allows the animator to change the environment and add external disturbances to produce a new motion that is like the reference motion but physically consistent. This physically based approach to motion tracking extends the utility of recorded motion data.

Parameters McSim requires less manual tuning than previous controllers. All physically based character animation tools require parameters that model the intrinsic physical properties of the character and the environment. In many previous approaches to control, a change in these intrinsic properties would require tuning many control parameters such as PD gains. McSim has two manually tuned parameters that were used to generate the results in this work: a gain on desired accelerations in the QP and the scale factor on PD gain. Changes in the physical parameters of the environment such as the mass of the characters limbs or the coefficient of friction required little or no adjustment to these parameters. McSim lessens the need for manual tuning by incorporating a dynamics model in its control decisions.

Speed McSim can run at interactive rates making it suitable for the control of characters in interactive applications such as games and training simulations. Furthermore, it requires no expensive precomputation step. The quadratic programs used by the controller can be solved efficiently by currently available hardware. However, solving the QP is the bottleneck in the controller and limits how often it can be run in an interactive system. While a parallel system could solve multiple QP's for decoupled characters, the computation effort does not scale well for multiple coupled characters or equivalently characters with many more degrees of freedom.

Robustness McSim is fairly robust to changes in motion type or style and changes in the physical environment. McSim tracks motions that stop and and start, walk, run, and jump. The motions were tracked thru obstacles and changes in the ground terrain. With the help of a heuristic balance mechanism in 2D, the controller tracked a running motion indefinitely and tracked a walk on a moving platform.

There are several cases where McSim can fail. McSim is not a high-level motion planner and cannot make large changes to the reference motion to adapt to large changes in the environment. For example, it cannot alter a walk to account for steps placed in its path. As a result, tracking the reference motion may result in the feet colliding with the steps and a fall. Also, tracking a single input motion is not a good strategy for robust and stable control of a physically simulated character. In this work, we experimented with a simple heuristic in 2D that adjusts the desired angle of the swing hip to help stabilize walking and running. In addition to finding an equivalent balance scheme in 3D, we would like to incorporate long range planning to allow for anticipatory actions that should increase robustness.

Quality While McSim's robustness can be improved, its main limitation is quality. Tracked motions can often appear more abrupt and robotic than the reference motion. While some changes to the motion are necessary to make it physically feasible, we believe that the quality of the controller can be improved by addressing some of the factors that currently limit quality. The first factor that limits quality is the tradeoff between computational performance and optimal performance over the entire motion. We are exploring ways making the controller optimal over longer time ranges without sacrificing performance. Quality was also impacted by the manually set parameters of the controller. For some 3D motions, it was more difficult to find parameters that produced nice results. Also, there were certain motions that we could not track well such as turning motions. An interesting area of future work would be to apply global optimization techniques that automatically tune the manually set parameters. In addition to reducing dimensionality, parameterizing control with our approach may help smooth the energy landscape, making it easier to find solutions.

Preliminary Improvements A key to addressing McSim's limitations is incorporating long-range optimal planning into the control. Incorporating long-range planning without adversely impacting computational effort is difficult. Some promising preliminary experiments indicate that long range planning can indeed improve tracking performance. Furthermore, the added cost comes in the form of precomputation, allowing the run-time controller to operate at similar rates as McSim. This preliminary work is based on linear quadratic regulators (LQR) [30, 32]. While LQR is a well-known approach to optimal control, applying it to the control of human characters is non-trivial because the dynamics are non-linear and non-smooth. In the sections that follow, we review LQR, describe approaches to applying LQR to human locomotion, and demonstrate improvements over McSim.

5.1 Linear Quadratic Regulator

Optimal control problems, when tractable, are typically solved using expensive numerical approximation techniques. LQR is a special case of optimal control than can be solved analytically and computed efficiently because it assumes linear dynamics and quadratic costs. Interested readers should reference more complete discussions of optimal control theory for background [5,30,32]. Here we sketch the LQR solution using dynamic programming on the Bellman equation.

The Bellman equation defines the optimal value function. A value function, $v(x, k)$, is the minimum cost of completing a task starting from the given state, x, at step k. The optimal value function is defined recursively as the minimum over all possible actions, u, of the current cost, $\phi(x, u, k)$, and the optimal value function evaluated at the next state, $v(F(x, u, k), k + 1)$, where $F(x, u, k)$ maps the current state and action onto a new state.

$$v(x, k) = \min_u \phi(x, u, k) + v(F(x, u, k), k + 1). \tag{5.1}$$

In LQR, the cost function is quadratic

$$\phi(x, u, k) = \frac{1}{2} x^T Q_k x + \frac{1}{2} u^T R_k u \tag{5.2}$$

with terminal cost $\phi(x, u, n) = \frac{1}{2} x^T Q_n x$ and the transition function is linear

$$F(x, u, k) = A_k x + B_k u. \tag{5.3}$$

Substituting Equations 5.2 and 5.3 into Equation 5.1 yields

$$v(x, k) = \min_u \frac{1}{2} x^T Q_k x + \frac{1}{2} u^T R_k u + v(A_k x + B_k u, k + 1). \tag{5.4}$$

This equation can be solved by guessing a quadratic form for the optimal value function, $v(x,k) = \frac{1}{2}x^T P_k x$ with $P_n = Q_n$. Using this guess, Equation 5.4 becomes

$$\frac{1}{2}x^T P_k x = \min_u \frac{1}{2}x^T Q_k x + \frac{1}{2}u^T R_k u + \frac{1}{2}(A_k x + B_k u)^T P_{k+1}(A_k x + B_k u). \tag{5.5}$$

The optimal control action is found by minimizing the right side

$$u = -(R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T V_{k+1} A_k x \tag{5.6}$$

which is abbreviated as $u = K_k x$. Substituting Equation 5.6 into Equation 5.5 yields a discrete *Ricatti* equation for V_k which can be solved for all k by starting at $k = n$ and iterating backwards.

5.2 Full-State Feedback

The first application of LQR we have tried linearizes the dynamics of the character around the reference motion. At each sample of the reference motion, we assume an Euler integration step yielding the A_k and B_k matrices defining the linear transition function. Furthermore, we assume that the linearized dynamics are fully-actuated. The control cost is $R_k = rI$. The positional cost penalizes deviations from the reference motion and is $Q_k = diag(0, 0, 0, q, q, \ldots, q)$ where $diag$ is a diagonal $2N\,x\,2N$ matrix. The first three entries are 0 meaning that deviation in the root position is not penalized.

The optimal gain matrices, K_k, from Equation 5.6 are found in a precomputation step by solving the Ricatti equations. At runtime, the current gain matrix is multiplied by the current tracking error to yield the desired feedback force. Unfortunately, applying the resulting feedback force directly is not possible because of two assumptions. First, since we assumed a fully-actuated model the feedback forces will have forces on the root of the character. Applying these forces directly will likely render the motion physically implausible. The real character is not fully-actuated and must make use of contact forces to accelerate the center of mass. Second, the linear approximation to the dynamics is only good in a neighborhood about the nominal trajectory. In our prototype system, these assumptions are maintained through the use of a QP. The QP tries to reproduce the desired force as closely as possible using feasible internal torques and ground reaction forces. It also tries to match the observed acceleration of the reference motion. This extra force acts as a feedforward control that maintains the validity of the linearized dynamics model used to compute the LQR controller.

In 2D, applying this full-state feedback control through the QP improves the visual quality of the animation and reduces tracking error as seen in Figure 5-1. These results were somewhat surprising given the crudeness of the dynamics model used. In particular, we thought that assuming full actuation would be a poor approximation. These results would indicate that at least for planar walking models, the character does have good control over the center of mass through the use of ground contacts.

5.3 Under-Actuated Approximations

The fully-actuated assumption turned out to be too crude for the control of 3D systems. For our next experiment, we tried to find a better model for LQR. LQR has been shown to maintain balance of simple under-actuated systems such as the acrobot [18]. We leveraged this capability by mapping the state of the character onto a simple under-actuated system geometrically, computing control forces using LQR, and then mapping these forces onto the full character through the use of a QP. This control scheme improved not only static balance but also the stability of controllers for walking in 2D and 3D.

The particular simple model we used for balance was a three-link model as shown in Figure 5-2. After it is constructed, the dynamics equations for this model were linearized about the upright equilibrium point where the model is statically balanced. This provides the A_k and B_k matrices needed to solve for the LQR control policy as in Equation 5.6. In this particular case, the Ricatti equations were solved over a long interval to approximate the optimal control over an infinite time interval. This yields

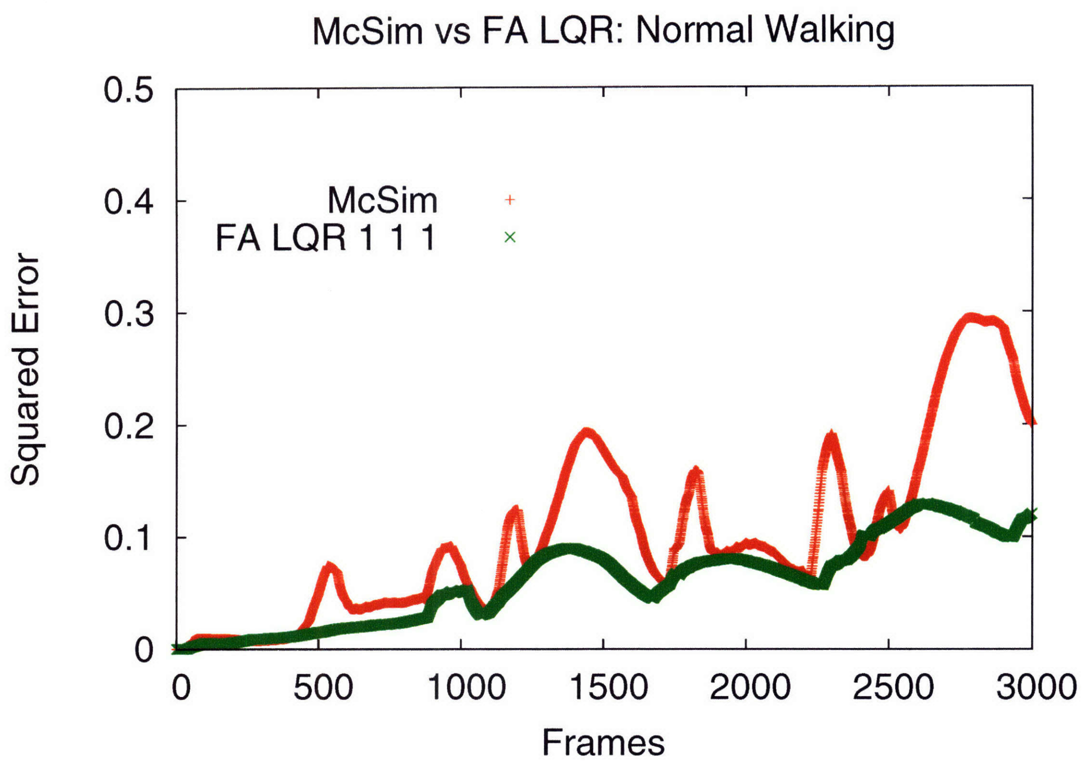


Figure 5-1: Shown are plots of squared error over time for two different tracking controller designs: the design discussed in §3 and the design discussed in §5.2. The LQR based controller has less error and is much smoother. FA stands for fully-actuated and 1,1,1 indicates the relative scaling of the terminal, position, and control costs used when solving for the LQR policy.

a single optimal gain matrix, K.

The performance of this simple control policy was evaluated on the simple model by perturbing it slightly and applying K directly to the deviation of the model from the equilibrium point. Qualitatively, the behavior of this controller is simple to describe. If the model is falling, it can rapidly swing its upper body in the direction of the fall and rotate the swing leg in the opposite direction to raise the center of mass over the swing leg. Previous heuristic approaches to balance have demonstrated a similar strategy on the linear inverted flywheel model [23].

Next, we examined whether this successful balance controller for the simple model could be used to balance a complex character. The QP from Equation 3.7 is altered

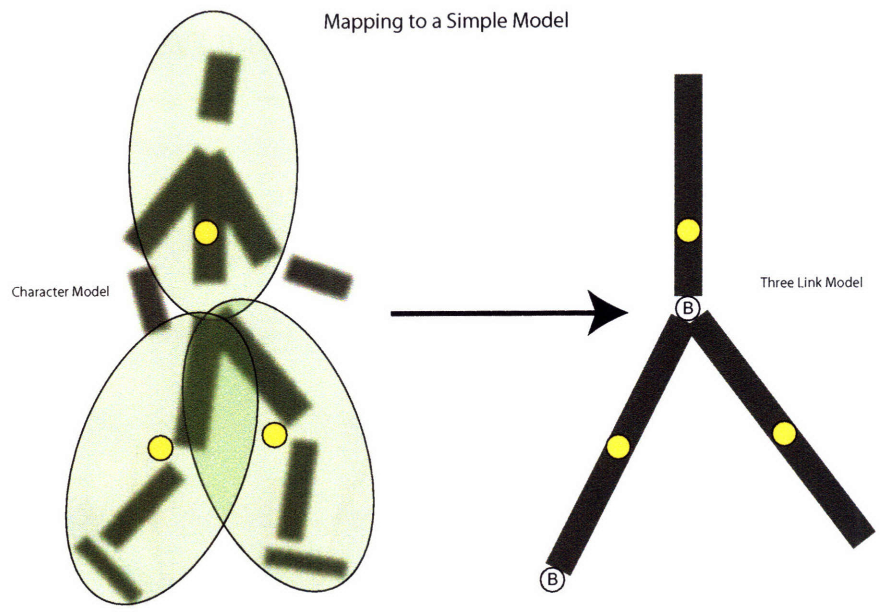


Figure 5-2: The mapping from a simulated character to a simple model. The first link in the simple model is anchored to the ground and is unactuated. It represents the support leg of a character balancing on one foot. The second link represents the upper body. The final link is anchored at the same location as the joint between the upper body and the support leg and represents the swing leg. The lengths, relative positions, and inertial properties of each link on the simple model are determined by aggregate properties of links on the original character. For example, the foot, lower leg, and upper leg links of one leg are used to determine the properties for the swing link. Given the state of the simple model, we can compute LQR control forces and apply them to the full model using a QP.

with three new objectives of the form

$$||J_i\ddot{q} + \dot{J}_i\dot{q} - a_d||^2 \tag{5.7}$$

where J_i is the Jacobian of the center of mass of the links associated with the i'th link of the simple model and a_d is the desired acceleration for that point as determined by the LQR policy computed on the simple balance model. These objectives are weighted to trade off between maintaining a rest pose and matching the accelerations of the simple model. This control strategy was able to maintain balance under external disturbances as shown in Figure 5-3.

5.4 Future Approximations

To move beyond balancing, we need to examine new simple linearized models. Consider walking. The three-link model used for balancing should be able to stabilize walking motions by manipulating the center of mass trajectory as it does when balancing. Linearizing about a static equilibrium point, however, is insufficient since the character must briefly lose balance to propel the center of mass forwards. Instead, we could linearize about a reference trajectory of the simple model as created by mapping the character's stepping motion onto the simple model. For jumping and running motions, we need a model that can operate while in contact and in free-flight such as the slip model [11].

To stably reproduce an arbitrary human motion, multiple simple models would have to be used over the duration of the motion. For a simple walking motion, at least two models are needed: one for each support foot. If the walking motion is more complex, exhibiting steps of different speeds, lengths, and styles, it is possible that a simple model would need to be constructed for each corresponding segment of the motion. More generally, all reference motions would need to be annotated with the appropriate control policy that should be used during tracking. Furthermore, the controller should only be allowed to switch between simple models at predefined nodes. It may not be advisable to switch from a right foot step to a two foot jump for example. These control policies and the associated switching points together define an *action graph* as shown in Figure 5-4. Defining this graph and deciding which edges to take at run time given the character's state and high-level goals is an exciting area of future research.

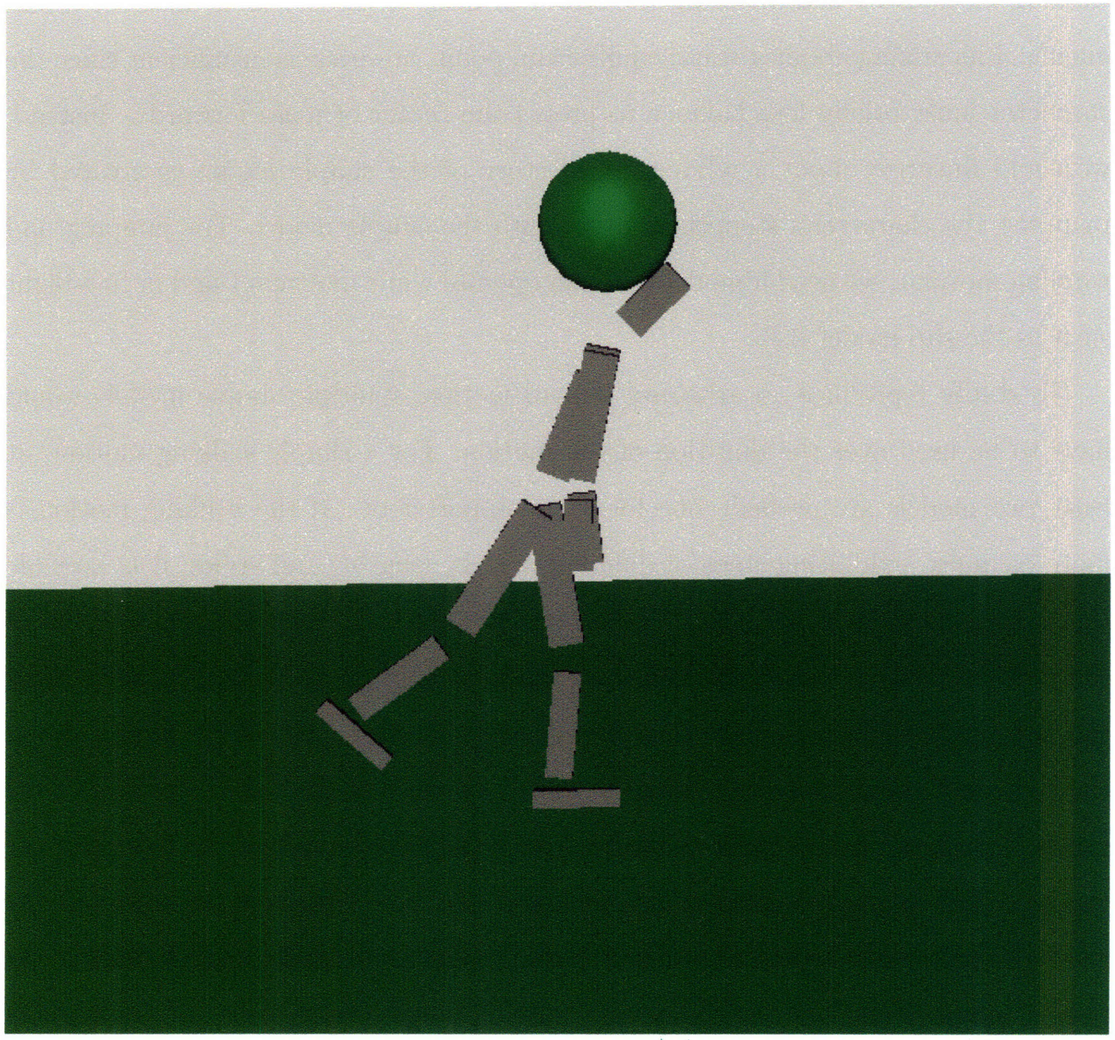


Figure 5-3: A character is hit on the head with a ball but maintains balance. The state of the character is mapped onto a three-link model. A precomputed LQR policy is executed on the simple model to generate desired accelerations for three center of mass positions on the character. These desired accelerations are achieved as well as possible by a QP.

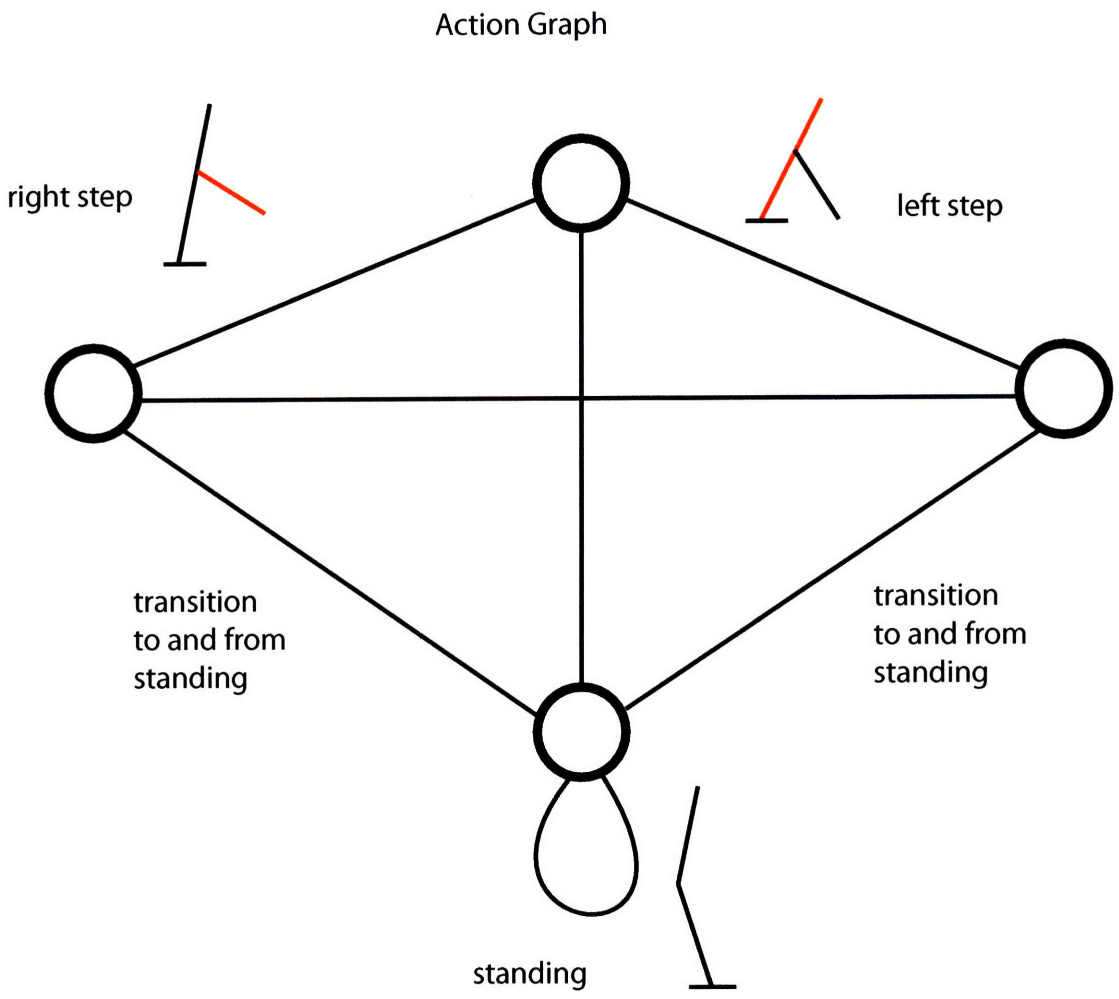

Action Graph
right step
left step
transition to and from standing
transition to and from standing
standing

Figure 5-4: An action graph. Edges represent reference motions (which may be empty) and associated control policies. Nodes indicate allowable transitions. At runtime, a higher level controller can decide which edge to take once each node is reached given the current character state and goals. Making these decisions optimal and automatic is an interesting area of future work.

Appendix A

Tables

Motion	k	k_{os}
2D Punchy	0.02	1000
Downhill	0.02	300
Walk Wave	0.02	1000
Sneaky*	0.005	500
2D Jump	0.05	1000
Run	0.2	600
Backwards	0.05	1000
Soldier	0.01	600
March	0.05	1000
Limp	0.08	1000

Table A.1: This table lists the relevant parameters used to generate selected results. k is a scale factor that multiplies the intrinsic joint stiffness parameters of the character listed in Tables A.2 and A.3 which are then used in the PD feedback component of the system. k_{os} is a gain used to calculate a modification to the acceleration from the input motion as in Equation 3.4. These two parameters were tuned manually to achieve a desired tracking result but reasonable results are achieved for a range of settings. For most 2D motions, values of k in the range between 0.005 and 0.5 worked. The setting of k_{os} is also flexible. Values in the range of 300 to 2000 typically work for this parameter. In many cases, the same settings achieved good results for many different motions. Starred motions were simulated using stiff springs at contacts. Despite using a different contact model, McSim tracks these motions well.

Link	k_s	Mass	Inertia
head	3000	3	0.011
upper arm	4000	2	0.022
lower arm	3000	1	0.009
torso	N/A	10	0.176
thigh	4000	7	0.121
shin	4000	5	0.077
foot	4000	4	0.019

Table A.2: This table lists the inertial properties of each link in the 2D model and the stiffness of the associated joint. Note that there is no stiffness for the unactuated root joint. It is also important to note that the stiffnesses listed here are not directly used by the PD feedback component. They are first scaled by a single scale parameter that is typically much less than one. This scaled value is used to calculate a critical damping gain. The units are as follows: newtons per radian for the gains, kilograms for the mass, and kilogram meters cubed for the inertias.

Link	k_s	Mass
trunk	N/A	12.92
thigh	4000	9.0853
shin	4000	3.944
foot	1000	1
toes	4000	0.3
thorax	3000	17.155
clavicle	4000	2.535
upper arm	4000	1.435
lower arm	3000	0.575
hand	3000	0.5

Table A.3: This table lists the inertial properties of each link in the 3D model and the stiffness of the associated joint. Again, there is no stiffness for the unactuated root joint.

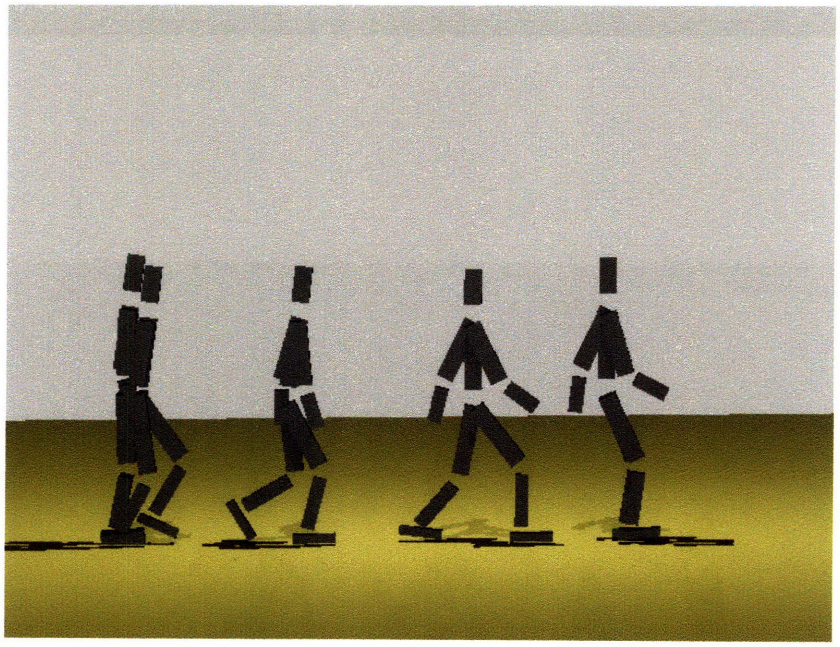
Num. Vars.	QP Solve Time (secs)
36	0.0013
44	0.0015
52	0.0023
68	0.003
150	0.007
154	0.0075
158	0.0097

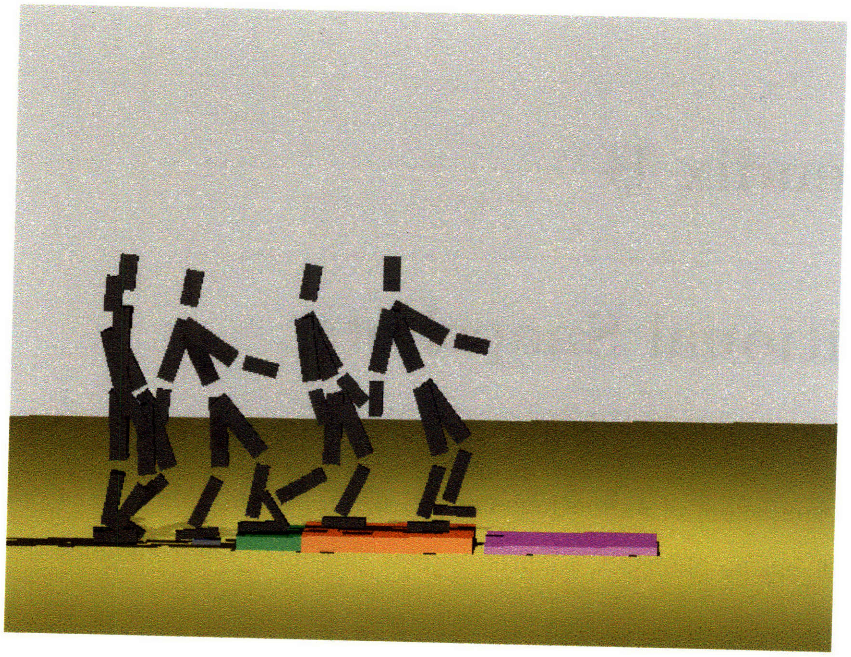
Table A.4: Timing results for the QP solver as a function of the number of variables in the QP. The number of variables is a function of the number of degrees of freedom in the character and the current contact state. Note that, for ease of implementation, we used dummy variables for the acceleration of each degree of freedom. This is not strictly necessary and would result in a much smaller QP problem.

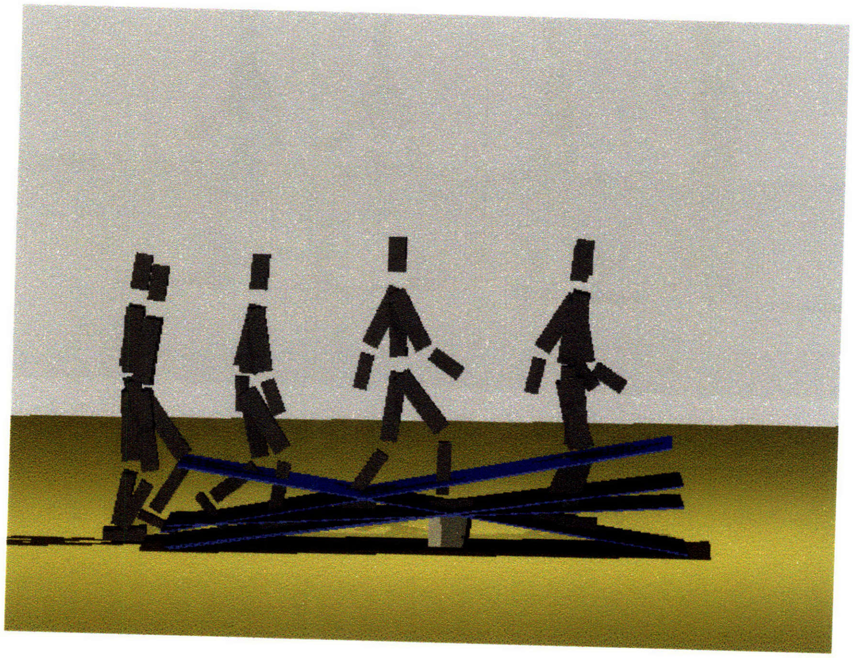
Appendix B

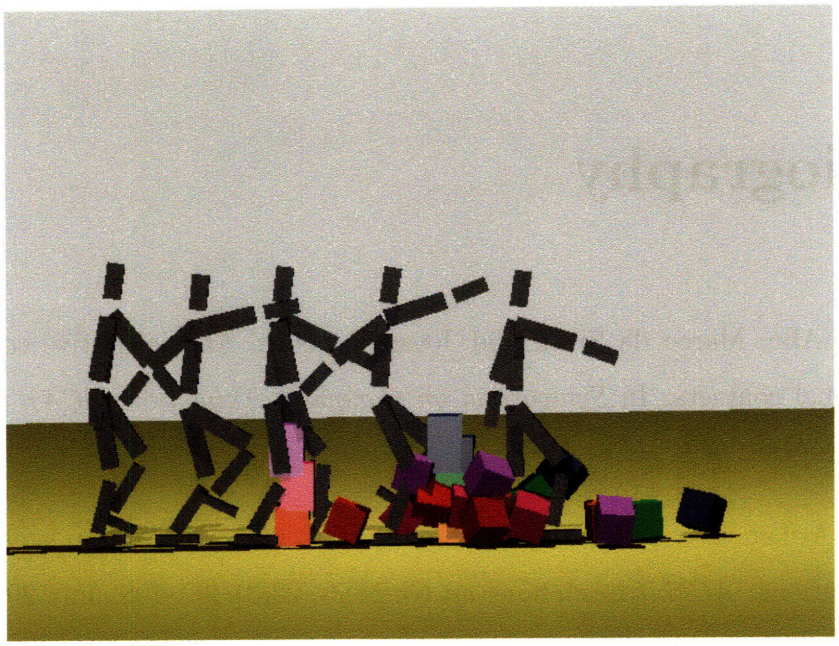
Additional Snapshots

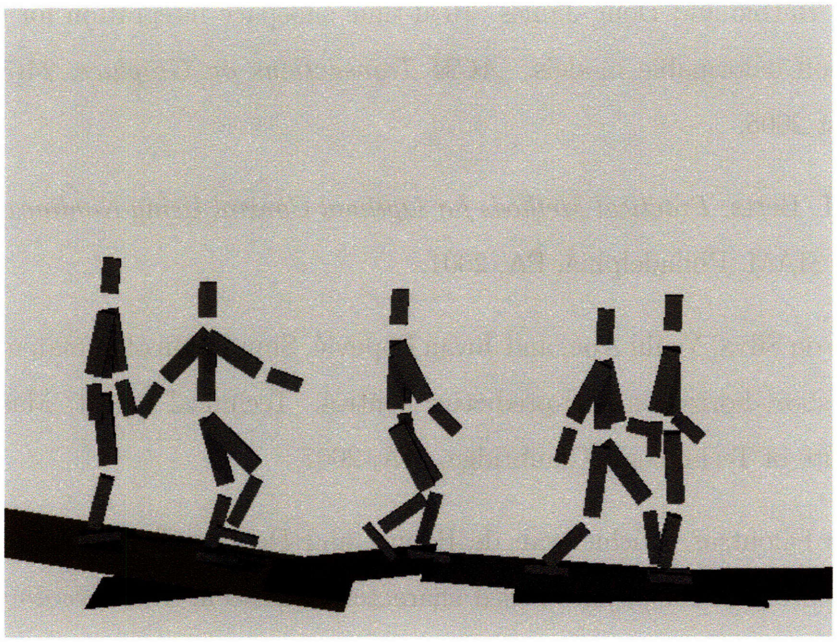
Additional snapshots of animations created using McSim. Motions of different style and content can be plausibly adapted to new simulated environments.











Bibliography

[1] Yeuhi Abe, Marco da Silva, and Jovan Popović. Multiobjective control with frictional contacts. In *Symposium on Computer Animation (SCA)*, pages 249–258, August 2007.

[2] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, July 2003.

[3] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, Annual Conference Series, pages 223–232. ACM SIGGRAPH, July 1989.

[4] Jernej Barbič and Doug James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Transactions on Graphics*, 24(3):982–990, August 2005.

[5] John T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM, Philadelphia, PA, 2001.

[6] Marco da Silva, Yeuhi Abe, and Jovan Popović. Simulation of human motion data using short-horizon model-predictive control. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2007.

[7] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, pages 251–260, August 2001.

[8] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics*, 22(3):417–426, July 2003.

[9] R. Featherstone and D. E. Orin. Robot dynamics: Equations and algorithms. In *International Conference on Robotics and Automation (ICRA)*, pages 826–834, 2000.

[10] Y. Fujimoto, S. Obata, and A. Kawamura. Robust biped walking with active interaction control between foot and ground. In *International Conference on Robotics and Automation (ICRA)*, pages 2030–2035. IEEE, 1998.

[11] R. J. Full and D. E. Koditschek. Templates and anchors: Neuromechanical hypotheses of legged locomotion on land. *The Journal of Experimental Biology*, 202:3325–3332, 1999.

[12] Philip E. Gill, Walter Murray, and Michael A. Saunders. User's guide for SQOPT 5.3: A fortran package for large-scale linear and quadratic programming. Technical Report NA 97–4, University of California, San Diego, 1997.

[13] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, pages 71–78, August 1995.

[14] A. Hofmann, S. Massaquoi, M. Popovic, and H. Herr. A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1952–1959. IEEE/RSJ, 2004.

[15] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089, August 2005.

[16] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[17] Shunsuke Kudoh, Taku Komura, and Katsushi Ikeuchi. The dynamic postural adjustment with the quadratic programming method. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2563–2568, 2002.

[18] Kangsik Lee and Victoria Coverstone-Carroll. Control algorithms for stabilizing underactuated robots. *Journal of Robotic Systems*, 15(12):681–697, December 1998.

[19] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics*, 24(3):1071–1081, August 2005.

[20] James McCann and Nancy Pollard. Responsive characters from motion fragments. *ACM Transactions on Graphics*, 26(3):6, 2007.

[21] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070, August 2005.

[22] Zoran Popović and Andrew P. Witkin. Physically based motion transformation. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, August 1999.

[23] J.; Drakunov S.; Goswami A. Pratt, J.; Carff. Capture point: A step toward humanoid push recovery. *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200–207, 4-6 Dec. 2006.

[24] Marc H. Raibert. *Legged Robots That Balance.* MIT Press, Cambridge, MA, 1986.

[25] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, Annual Conference Series, pages 349–358. ACM SIGGRAPH, July 1991.

[26] Alla Safonova, Jessica Hodgins, and Nancy Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3):514–521, August 2004.

[27] Ari Shapiro, Petros Faloutsos, and Victor Ng-Thow-Hing. Dynamic animation and control environment. In *Proceedings of Graphics Interface (GI)*, pages 61–70, 2005.

[28] Daniel Sharon and Michiel van de Panne. Synthesis of controllers for sylized planar bipedal walking. In *International Conference on Robotics and Automation (ICRA)*, pages 2387–2392, 2005.

[29] Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. In *Computer Graphics (Proceedings of SIGGRAPH 07)*, Annual Conference Series, page 107. ACM SIGGRAPH, aug 2007.

[30] Robert F. Stengel. *Optimal Control and Estimation.* Dover Publications, New York, 1994.

[31] Adnan Sulejmanpasić and Jovan Popović. Adaptation of performed ballistic motion. *ACM Transactions on Graphics*, 24(1):165–179, January 2005.

[32] E. Todorov. *In Bayesian Brain: Probabilistic Approaches to Neural Coding, Doya K (ed).* MIT Press, 2006.

[33] Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. In *Computer Graphics (Proceedings of SIGGRAPH 06)*, Annual Conference Series, pages 826–834. ACM SIGGRAPH, 2006.

[34] M. van de Panne and A. Lamouret. Guided optimization for balanced locomotion. In *Eurographics Workshop on Computer Animation and Simulation*, pages 165–177, 1995.

[35] Miomir Vukobratovic and Branislav Borovac. Zero-moment point – thirty five years of its life. *International Journal of Human Robotics*, 11(1):157–173, 2004.

[36] P. B. Wieber. On the stability of walking systems. In *International Workshop on Humanoid and Human Friendly Robotics*, pages 1–7, 2002.

[37] Pierre-Brice Wieber and Christine Chevallereau. Online adaptation of reference trajectories for the control of walking systems. *Robotics and Autonomous Systems*, 54(7):559–566, July 2006.

[38] Andrew Witkin and Michael Kass. Spacetime constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, volume 22, pages 159–168, August 1988.

[39] J.K. Wooten, W.L.; Hodgins. Simulating leaping, tumbling, landing and balancing humans. *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 1:656–662 vol.1, 2000.

[40] K. Yin, M. Cline, and D. K. Pai. Motion perturbation based on simple neuromotor control models. In *Pacific Conference on Computer Graphics and Applications (PG)*, pages 445–449, 2003.

[41] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: simple biped locomotion control. In *Computer Graphics (Proceedings of SIGGRAPH 07)*, Annual Conference Series, page 105. ACM SIGGRAPH, 2007.

[42] Victor B. Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *Symposium on Computer Animation (SCA)*, pages 89–96, July 2002.