Optimization-Based Interactive Motion Synthesis

SUMIT JAIN, YUTING YE, and C. KAREN LIU
Georgia Institute of Technology

We present a physics-based approach to synthesizing motion of a virtual character in a dynamically varying environment. Our approach views the motion of a responsive virtual character as a sequence of solutions to the constrained optimization problem formulated at every time step. This framework allows the programmer to specify active control strategies using intuitive kinematic goals, significantly reducing the engineering effort entailed in active body control. Our optimization framework can incorporate changes in the character's surroundings through a synthetic visual sensory system and create significantly different motions in response to varying environmental stimuli. Our results show that our approach is general enough to encompass a wide variety of highly interactive motions.

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

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Character animation, physics-based animation, nonlinear optimization

ACM Reference Format:

Jain, S., Ye, Y., and Liu, C. K. 2009. Optimization-based interactive motion synthesis. ACM Trans. Graph. 28, 1, Article 10 (January 2009), 12 pages. DOI = 10.1145/1477926.1477936 http://doi.acm.org/10.1145/1477926.1477936

1. INTRODUCTION

To date, animating the behavior of human characters in a dynamic environment remains primarily an animator-driven activity. Unlike simulation of passive phenomenon such as smoke, water, and clothing where automated algorithms have seen wide commercial adoption, the reaction of a virtual human character depends largely on the interaction between her own goals and environmental factors, in addition to the laws of physics. For example, when losing balance a real person will reposition her body to slow the fall while grabbing onto any nearby object that appears stable. Even for such a simple task, the sheer scale of possible objects and environments a character can interact with makes designing a generic simulation algorithm challenging. Consequently, to date, character animation is still primarily done through key-framing or blending prerecorded motion sequences.

Physical simulation via robotic controllers has the potential to be a general framework for simulating believable character interactions without need of extensive data or user effort. In the past, specialized control algorithms have proven capable of generating diverse motions such as balancing, running, and diving. Despite these successes, robotics controllers exhibit two main drawbacks. First, designing robotics controllers is a difficult and time-consuming process. Good controller design requires modeling of the musculoskeletal system and tuning of model parameters that have nonlinear relationships with the output motion. Second, once designed, controllers are often brittle, only working under a narrow range of conditions. Changes in the environment often necessitate significant tuning of control parameters or a redesign of the controller itself.

We explore an alternative framework for active character simulation, physics-based optimization, which formulates motion synthesis as an optimization problem. Up to now, physics-based optimization has been applied to generating motions in preplanned situations where all the constraints and objectives are known a priori. Within this domain, the framework has proven capable of synthesizing a wide class of realistic human motions, from walking to complex gymnastics. In addition, user control of the animation is straightforward. To specify a motion, the programmer only needs to describe the goals of the motion (e.g., jump to this position). The optimization framework handles how the motion is achieved. These traits make physics-based optimization an appealing framework to synthesize character animations.

This article describes a physics-based optimization framework for *interactive* character animation. In our system, the motion of responsive virtual character is a sequence of solutions to a constrained optimization problem formulated at every time step. In this framework, the controller is a process that directs the motion by providing kinematic goals, such as desired body position or velocity, into the optimization problem at each time step. Instead of explicitly solving for internal joint torques and numerically integrating them to solve for motion, our approach directly optimizes the joint configurations subject to the laws of physics, environment constraints, bio-mechanical limitations, and task-level control strategies. When the character is in contact with the environment, we also explicitly

This work was supported by NSF grant CCF-CISE 0742303.

Authors' address: S. Jain, Y. Ye, and C. K. Liu, Georgia Institute of Technology, Atlanta, GA 30332; email: {sumit, yuting, karenliu}@cc.gatech.edu.

optimize the contact forces to achieve desired tasks while maintaining physical realism. For a passive dynamic system, there is no benefit in using the optimization to solve for the motion, since the problem is well constrained and can be solved efficiently by a standard forward simulator. The real advantages of our method are exposed when simulating an active dynamic system in sustained contact with the environment, such as most everyday human activities. The unknown actuation and contact forces in such a dynamic system pose an underdetermined problem. By solving the actuation (implicitly), contact forces, and the final motion all in one procedure, our method allows the control policies to be intuitively formulated into functions of joint configurations without referring to forces and torques.

In this article, we demonstrate several benefits that arise from casting interactive motion synthesis as an optimization problem. First, the programmer can specify active control strategies through a controller using kinematic goals, thus retaining the intuitive user-level control that optimization offers in offline motion synthesis. Second, the programmer can compose complex control strategies by combining simple control strategies in a finite-state machine-like structure. We demonstrate a versatile virtual character coping with a series of unexpected disturbances, using a combination of control strategies in an autonomous fashion (Figure 10). Third, the character can perceive changes in the environment through a synthetic visual sensory system. Our optimization incorporates this information along with other high-level decisions as additional objectives and constraints. Consequently, the same controller creates significantly different motions in response to different environmental stimuli. As examples, we demonstrate a character that can use environment features to retain her balance while dodging flying objects. Finally, as our results show, this framework is general enough to encompass a wide variety of highly interactive motions. We demonstrate this scope, from a simple balance controller, to wall climbing and gymnastics.

2. RELATED WORK

Designing a virtual human character that actively responds to the physical environment is a long-standing challenge in computer animation. The variational optimization-based approach directly solves for the entire motion trajectory according to energy considerations and user specifications. Solving for nonlinear dynamic constraints and energy-based objective functions produces good results on simple skeletons [Witkin and Kass 1988; Cohen 1992; Liu et al. 1994], as well as on abstract human models [Popović and Witkin 1999] with simplified dynamics [Liu and Popović 2002; Fang and Pollard 2003]. With the aid of motion data, researchers have formulated the optimization problem in a reduced space biased towards natural human motion [Safonova et al. 2004; Sulejmanpašić and Popović 2004], or extracted parameters from the data that capture muscle preferences and joint stiffness [Liu et al. 2005]. The optimization approach allows the programmer to describe the motion task by providing key-frame-like constraints in the joint space. However, standard optimization-based approaches are not suited for interactive applications because all the constraints and objectives need to be specified a priori. Our method treats every simulation time step as an independent optimization problem with a new set of constraints and objectives. Any unscripted events in the current time step, such as user input or collisions, will be responded to appropriately in the next time step.

Active body control with physical simulation presents many obvious advantages over the optimal trajectory approach in the domain of creating responsive virtual characters. Researchers have designed basic balance controllers for bipedal systems [Raibert 1986; Laszlo et al. 1996; van de Panne and Lamouret 1995; Sharon and van de Panne 2005; Abe et al. 2007; Kudoh et al. 2006], as well as more versatile motions such as running, vaulting, and cycling [Hodgins et al. 1995]. Yin et al. reduce control design to a few kinematics poses with the help of a robust balance strategy for walking and running [Yin et al. 2007]. Wooten [1998] and Hodgins et al. [1995] concatenated a sequence of transition controllers that generate successive motion sequences. Faloutsos et al. [2001] demonstrated that a virtual character can be simulated by composing multiple primitive robotic controllers. Several companies have successfully applied similar technologies to commercial products by providing a repertoire of motor skills [NaturalMotion 2006]. The specific details regarding their implementation are unknown, but it is likely that each individual controller requires fine-tuning of the physical parameters. To synthesize animations involving interactions with the environment, they rely on users to establish contact constraints at the right timing. Robotic controller simulation yields physically plausible motion, often in real time, but requires an expert to tune the parameters properly. Our work provides a generic framework for rapidly designing active control procedures that require minimal physical parameter tuning, yielding an adaptable controller for characters of arbitrary design.

To circumvent the issues of overspecialization, many researchers suggested exploiting online, local optimization techniques that adjust the current dynamic parameters to new situations [Abe et al. 2007; Yamane and Nakamura 2000; Stewart and Cremer 1992a, 1992b]. Our method is inspired by the same idea, but instead of adjusting the parameters in the force domain and obtaining the motion by means of numerical integration methods, we directly optimize the joint configurations according to the control policies. By directly controlling the joint configuration instead of joint acceleration, we can formulate constraints that are satisfied exactly in the joint space, without numerical errors due to the integration. Furthermore, we do not require the constraints and their first derivative to be satisfied initially. This flexibility allows us to arbitrarily add or change constraints in the position space.

Much previous work in robotics has addressed the problem of controlling multiple tasks for robots or manipulators [Liegeois 1977; Maciejewski and Klein 1985; Sentis and Khatib 2006, 2005]. In computer graphics, Abe and Popović [2006] demonstrated a prioritized control approach that allows a virtual character to execute tasks at different priority levels without interfering with a posture tracking controller. They later proposed an optimization-based control that formulates the multiple objectives into a quadratic programming problem. This formulation allows for a compromise between several conflicting objectives, such as balancing and pose tracking [Abe et al. 2007]. Our framework also addresses the problem of multiple objectives by formulating an optimization. Instead of solving for the control, however, we directly solve for motion that achieves the coordination among multiple objectives. The weights of the objectives directly influence the task priority in the motion without interference from other physical parameters. Furthermore, because we use a constrained optimization to solve for motion, we can formulate a primary task that can never be violated as a constraint instead of an objective.

Using key-frame-like control to create physically responsive animation provides a practical tool for many computer animation applications [Isaacs and Cohen 1987; Stewart and Cremer 1992b]. Our simple framework for specifying control strategies is inspired by earlier systems designed by Stewart and Cremer [1992a]. Their proposed control schemes allow the programmer to control any linear combination of the state variables in the second derivative domain, such as the acceleration of the center of mass. Our optimization

formulation further allows programmers to control variables themselves, rather than using a more complicated second derivative domain. Liu [2008] described a similar optimization framework for synthesis of hand animation based on specifications of the manipulated objects. She demonstrated that simple grasping-like tasks can be produced with a few key-frames on the object. Our approach addresses more complicated issues such as postural balance and coordination in full body motion. In addition, our approach aims for designing an autonomous dynamic system by incorporating the sensory information to the control system.

3. OVERVIEW

We view responsive character motion as a sequence of solutions to a constrained optimization formulated at every time step. Each optimization yields an optimal joint configuration based on the user-specified goals and energy considerations, subject to the laws of physics. We break down our motion synthesis framework into following main components.

(1) *Motion Synthesizer.* The motion synthesizer forms the core of the framework. Given the current dynamic state of the character, the motion synthesizer formulates an optimization that solves for the joint configuration of the next time step. To ensure physical realism in the synthesized motion, we enforce Lagrange's equations of motion and Coulomb's friction model as constraints and minimize the change of muscle force usage as the objective in the optimization.

(2) *User-Specified Controller.* To synthesize an active character behavior, the controller adds kinematic goals to the objective function of the optimization at the current time step. The programmer creates this controller by specifying goals conforming to the character's internal dynamic state and/or the external environmental state.

(3) *Environment Knowledge from Visual Senses.* Complex control strategies often depend on sensory inputs that the character gathers from the environment. We endow our virtual characters with a synthetic visual sensor and allow the programmer to formulate control strategies that depend on the sensor input.

(4) *User Interaction.* The user can interact with the ongoing character motion by applying external forces or adding kinematic constraints or objectives in an interactive fashion.

Figure 1 illustrates the relationship between the components described before. At each time step, the user-specified controller formulates appropriate objectives and constraints based on the current dynamic state of the character and the environment information from the visual sensory system. The motion synthesizer formulates an optimization problem comprising the controller-generated objective and constraints, external disturbances, and objectives and constraints enforcing physical realism. The solution to the optimization problem yields the character's joint configuration for the next time step.

4. OPTIMIZATION SETUP FOR MOTION SYNTHESIS

The heart of our framework is the formulation of an optimization problem to synthesize the character's motion at each time step. Given physical constraints and objectives specified by the programmer, we solve for the character's joint configuration for the next time step and for external contact forces simultaneously in the optimization. Solving for external contact forces is equivalent to determining how much force the character applies at the point of contact. We do not solve for internal joint muscles explicitly in the optimization.

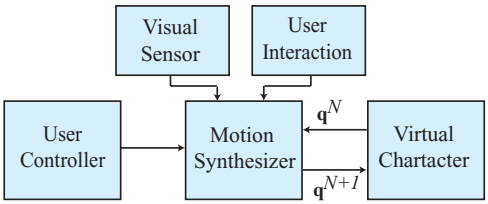


Fig. 1. The controller provides control strategies to the motion synthesizer, which takes in the current dynamic state of the character, $\mathbf{q}^N$, and outputs new joint configuration, $\mathbf{q}^{N+1}$, for the next time step. Additional inputs can be provided to the synthesizer in the form of user interactions and environment knowledge through a visual sensory system.

We represent the character's skeleton as a transformation hierarchy of 18 body nodes with 31 Degrees Of Freedom (DOFs) in reduced coordinates representing joints and 6 DOFs representing the global translation and rotation.

To enforce physical realism in the synthesized motion, we formulate Lagrange's equations of motion as constraints in our optimization. Lagrange's equations are a reformulation of Newton's equations of motion in generalized coordinates (DOFs in our case). We enforce Lagrange's constraint L_j on each DOF q_j of the root of the skeleton's hierarchy

$$L_j(\mathbf{q}, \boldsymbol{\lambda}) = \sum_{i \in N(j)} \left(\frac{d}{dt} \frac{\partial T_i}{\partial \dot{q}_j} - \frac{\partial T_i}{\partial q_j} \right) - Q_j^g - Q_j^c(\boldsymbol{\lambda}) - Q_j^{ext} = 0, \tag{1}$$

where $\mathbf{q} \equiv (q_0, q_1, \ldots)^T$ is a vector of all DOFs and $\boldsymbol{\lambda}$ are the parameters of contact forces. Further, Q_j^g, Q_j^c, and Q_j^{ext} represent the gravitational force, contact force, and an additional external force, respectively, in generalized coordinates. The formulation of these generalized forces is described in Appendix A.

The first two terms in Eq. (1) measure the inertia force due to the acceleration of DOF q_j in generalized coordinates. T_i denotes the kinetic energy of body node i and $N(j)$ is the set of body nodes in the subtree of DOF q_j. In the transformation hierarchy, the inertia force of node i due to the DOF q_j can be computed as

$$\frac{d}{dt} \frac{\partial T_i}{\partial \dot{q}_j} - \frac{\partial T_i}{\partial q_j} = \mathrm{tr} \left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i{}^T \right), \tag{2}$$

where tr () gives the trace of a matrix and $\mathbf{W}_i$ is the chain of homogeneous transformations from the root node to body node i. Moreover, $\mathbf{M}_i$ denotes the mass tensor of the body node i, defined in Appendix A.

Because we represent time as discrete samples, all the functions of time-varying variables need to be represented in a discrete domain. We discretize the time into samples with small intervals Δt. We define the velocity and the acceleration of a DOF q_j, at current time sample N, by central finite differences.

$$\dot{q}_j^N \equiv \frac{q_j^{N+1} - q_j^{N-1}}{2\Delta t} \tag{3}$$

$$\ddot{q}_j^N \equiv \frac{q_j^{N+1} - 2q_j^N + q_j^{N-1}}{\Delta t^2} \tag{4}$$

For a body node i, $\dot{\mathbf{W}}_i$ and $\ddot{\mathbf{W}}_i$ at time sample N can be written as

$$\dot{\mathbf{W}}_i = \frac{\partial \mathbf{W}_i}{\partial \mathbf{q}}\dot{\mathbf{q}} = \sum_j \frac{\partial \mathbf{W}_i}{\partial q_j}\dot{q}_j \tag{5}$$

$$\ddot{\mathbf{W}}_i = \sum_j \left(\sum_k \left(\frac{\partial^2 \mathbf{W}_i}{\partial q_k \partial q_j}\dot{q}_k \right)\dot{q}_j + \frac{\partial \mathbf{W}_i}{\partial q_j}\ddot{q}_j \right). \tag{6}$$

For clarity, we drop the superscript henceforth for quantities at time sample N. The derivative terms $\frac{\partial \mathbf{W}_i}{\partial q_j}$ and $\frac{\partial^2 \mathbf{W}_i}{\partial q_k \partial q_j}$ can be computed analytically, since $\mathbf{W}_i$ is a differentiable function of $\mathbf{q}$.

In this discrete formulation, the optimization at each time step N solves for the DOFs at the next time step, $\mathbf{q}^{N+1}$, given the current and previous DOFs, $\mathbf{q}^N$, and $\mathbf{q}^{N-1}$. Using Eqs. (6) and (3), Lagrange's constraint (Eq. (1)) is reformulated as

$$L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) = \sum_{i \in N(j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i{}^T(\mathbf{q}^{N+1}) \right) - Q_j^g - Q_j^c(\boldsymbol{\lambda}^N) - Q_j^{ext} = 0. \tag{7}$$

The only terms depending on unknowns in Eq. (7) are $\ddot{\mathbf{W}}_i$ and Q_j^c. All other terms can be readily evaluated using $\mathbf{q}^{N-1}$ and $\mathbf{q}^N$, which serve as constants in the optimization. Once this optimization is solved, we advance our time by Δt, making $N+1$ as the current time sample.

Our formulation has the same order of accuracy as the second-order Runge-Kutta method in solving ordinary differential equations numerically. To improve the time performance in practice, we define the velocity of a DOF q_j using backward finite differences.

$$\dot{q}_j^N \equiv \frac{q_j^N - q_j^{N-1}}{\Delta t} \tag{8}$$

This definition of joint velocity sacrifices the second-order accuracy; however, Lagrange's equation becomes a linear function of unknowns $\mathbf{q}^{N+1}$ and $\boldsymbol{\lambda}^N$, allowing for a much more efficient quadratic programming formulation. We provide details on linearization of constraints in Appendix B.

4.1 Muscle Control

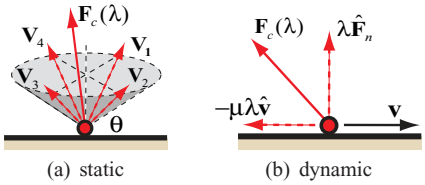
With only 6 equations (Eq. (7)) on the root DOFs, this system is largely underdetermined and has infinitely many solutions. We do not enforce Eq. (7) on joint DOFs, as they are implicitly equipped with muscles or actuators that can generate arbitrary forces to satisfy Eq. (7). This formulation is equivalent to computing the aggregate force and torque [Fang and Pollard 2003] and the low-order dynamic constraints [Sulejmanpašić and Popović 2004].

To bias the solution towards a more plausible configuration, we incorporate the minimal torque change model in the optimization [Kawato 1999; Uno et al. 1989]. Natural human motion tends to remain smooth in the acceleration domain with limited ability to change the muscle activation rapidly over time.

Therefore, minimizing the change of joint torques in time discourages the muscle forces from changing abruptly and excessively.

We define the generalized muscle force usage at each actuated joint DOF q_j as Q_j^m, which represents the sum of torques generated internally by musculoskeletal components. Lagrange's constraint for each actuated DOF can then be expressed as

$$L_j{}'(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) = L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) - Q_j^m = 0. \tag{9}$$



(a) static (b) dynamic

Fig. 2. Contact forces associated with different types of contacts.

Using this equation, change of muscle force is defined as

$$\dot{Q}_j^m = \dot{L}_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) \equiv L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) - L_j(\mathbf{q}^N, \boldsymbol{\lambda}^{N-1}), \tag{10}$$

where $L_j(\mathbf{q}^N, \boldsymbol{\lambda}^{N-1})$ is the muscle force at the previous time step and serves as a constant in the equation.

We propose a simple method to regulate the muscle forces to achieve natural human motion. We add objectives, $\dot{L}_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N)$, for each actuated DOF q_j, to minimize the change in muscle forces over time. When the character reacts to unexpected events, such as being pushed by the user, we simulate the activation delay in adjusting the muscles by minimizing the objective $L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) - L_j(\mathbf{q}^{N_0}, \boldsymbol{\lambda}^{N_0-1})$ for a small time interval ($\approx$ 200ms), where $L_j(\mathbf{q}^{N_0}, \boldsymbol{\lambda}^{N_0-1})$ is the muscle force usage at the moment of the push. This delay in muscle response is due to the delay in the internal spinal feedback loop and external visual feedback loop [Kawato 1999; Lockhart and Ting 2007] Such a simple feature results in a natural passive reaction to the push.

4.2 Contact Model

Our method explicitly optimizes the contact forces subject to Lagrange's equations of motion and Coulomb's friction model. This implies that the character can use any contact forces within correct range of the friction model to satisfy Eq. (1) and help achieve other objectives.

The standard optimization formulation usually handles a sustained contact by adding a positional constraint and Lagrangian multipliers parameterizing the contact force in the dynamic equations. The drawback of this setup is that, instead of breaking off the contact, the optimization will become infeasible when the equations of motion cannot be satisfied simultaneously with the constraints imposed by the friction model. We instead enforce a more relaxed nonpenetrating constraint that prevents interpenetration of the points in contact but allows for contact slippage and breakage. The formulation of the contact forces depends on whether the contact is static or dynamic.

—*Static.* A static contact has zero tangential velocity along the surface. According to Coulomb's friction model, repulsive contact forces should lie within the *cone* defined by the static friction coefficient μ, whose generatrix forms an angle $\theta = cot^{-1}\mu$ with the surface of contact. We approximate this friction cone by four basis vectors with non-negative basis coefficients (Figure 2(a)). The contact force is computed as a linear combination of these bases $\mathbf{V}$ as:

$$\mathbf{F}_c(\boldsymbol{\lambda}) = \mathbf{V}\boldsymbol{\lambda}, \ \boldsymbol{\lambda} \geq \mathbf{0}, \tag{11}$$

where $\boldsymbol{\lambda}$ represents the coefficient vector $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$.

—*Dynamic.* A dynamic contact slips along the surface with the friction force directed opposite to the velocity, $\mathbf{v}$, of the contact point (Figure 2(b)). The contact force is computed as

$$\mathbf{F}_c(\boldsymbol{\lambda}) = (\hat{\mathbf{F}}_n - \mu\hat{\mathbf{v}})\boldsymbol{\lambda}, \ \boldsymbol{\lambda} \geq \mathbf{0}, \tag{12}$$

where λ is a vector of one coefficient representing the magnitude of the normal contact force $\mathbf{F}_n$.

The generalized contact force, Q_j^c, for q_j, is given by

$$Q_j^c(\lambda) = \left(\frac{\partial \mathbf{W}_b}{\partial q_j}\mathbf{p}_b\right).\mathbf{F}_c(\lambda), \tag{13}$$

where $\mathbf{p}_b$ is the contact point in the local coordinates of body node b. Depending on the type of contact, $\mathbf{F}_c$ is parameterized by λ consisting of either one or four coefficients.

When a contact k is established, we add a nonpenetrating inequality constraint, $\mathbf{C}_{np}^k(\mathbf{q}^{N+1}) > \mathbf{0}$, to the optimization, and solve for the contact forces that help satisfy the constraint. In addition, these contact forces are constrained either by static or dynamic friction forces, based on the tangential velocity at the contact point. If the normal velocity is nonzero, the contact breakage occurs and we simply remove the contact from optimization for the next time step.

Apart from these contacts based on the unilateral friction constraint, the character can be in contact with an object by grabbing onto it. In such a case, the contact forces are unconstrained due to the bilateral grip and we simply add three unconstrained coefficients for the forces in the optimization.

Because we only enforce a nonpenetrating constraint on the contact, the character might choose to use her own muscles to unnecessarily slide along the surface or even to break off the contact. Therefore we need an incentive for the character to move the contact point only when it is physically impossible to maintain the contact, or when an overpowering conflicting objective occurs. To this end, we add an objective $G_c^k(\mathbf{q}^{N+1})$ to minimize the movement of the contact point. By using these objectives to reduce voluntary slippage and breakage, we can make the character behave in a more human-like manner without sacrificing physical correctness.

One drawback of this contact model is that it does not enforce the principle of zero virtual work for contact forces. In other words, the contact force is not guaranteed to be zero at the moment of the breakage, resulting in a nonzero amount of work done. When this situation is detected, we roll back our motion to the previous frame and enforce the contact force as zero.

4.3 Optimization Summary

Eq. (14) summarizes the formulation of the optimization problem at each time step using the constraint and objective notations introduced in this section.

$$\begin{aligned} \underset{\mathbf{q}^{N+1},\lambda^N}{\text{argmin}} \quad & E = \sum_{j=6}^{36} \|\dot{L}_j(\mathbf{q}^{N+1}, \lambda^N)\| + \sum_k \left\| G_c^k(\mathbf{q}^{N+1}) \right\| \\ \text{subject to} \quad & \begin{cases} L_j(\mathbf{q}^{N+1}, \lambda^N) = 0, \ j = 0, \cdots 5 \\ \mathbf{C}_{np}^k(\mathbf{q}^{N+1}) > \mathbf{0}, \ \forall k \end{cases} \end{aligned} \tag{14}$$

5. FRAMEWORK FOR ACTIVE CONTROL

The motion synthesizer described in Section 4 produces physically plausible motion with regards to frictional contacts and smooth changes in muscle activations. Without any active control, however, the character will quickly fall on the ground under the influence of gravity. The goal of the controller is to direct the virtual character's active motion in reaction to the environment to achieve a specified task.

In our framework, the controller comprises a user-specified control strategy that maps the character's dynamic state and the environment state to an appropriate set of objectives and constraints. These objectives and constraints describe the desired goal of an active motion as a function of the character's joint position and derivatives. At each time step, the controller determines appropriate control strategies and adds the desired objectives and constraints to the current optimization problem. Consequently, the optimizer must generate a motion that follows the dictates of the controller while also satisfying the physical constraints in the environment.

Our framework allows for a more intuitive specification of controller behavior than prior optimal control algorithms for dynamic systems. In prior optimal control algorithms, a controller is a model of the physical actuators responsible for generating the internal force that creates a desired motion. In our framework, a controller is essentially a statement of the kinematic goal of the motion, specified as functions of joint DOFs $\mathbf{q}^{N+1}$ and the external contact forces λ^N.

To design a controller for complex interaction with the environment, our approach allows the programmer to intuitively describe control strategies as a sequence of kinematic actions, such as desired poses and potential contacts with the environment. The entire process of controller design does not require the programmer to fine-tune the physical parameters representing the joint actuators. However, to create a specific output motion, the programmer has to find a balanced set of weights for the objectives. In our experience, tuning objective weights is relatively easier because the weights only determine the high-level relative importance among competing objectives, rather than the physical properties of joints and muscles. Consequently, one set of weights is consistently applied across all joints and can be used for different characters in different environments. Furthermore, a wide range of weights can produce different but equally plausible motion sequences. We provide the exact weight settings used in our examples in Section 6, but the programmer can vary these values to create a variety of motions.

5.1 Controller Specification

Formally we define a controller as a finite-state machine (FSM) $\mathbf{M} = (\mathbf{S}, \mathbf{T})$ with states $\mathbf{S}$ and allowed transitions $\mathbf{T}$. Each state is a basic control strategy comprising a set of control objectives $(G_1, G_2, \ldots)$ representing the kinematic goals of the desired motion, and a Boolean condition D representing when the strategy is applicable (a condition returns **false** when it fails or is not applicable). Control objectives, G_i's, can be represented as functions of the character's joint positions: $G_i = \|f(\mathbf{q}^{N+1})\|$. Each objective function has an associated weight which indicates its relative importance in the optimization. We assign these weights based on our understanding of human locomotion and on experimentation. In our experiments, we never needed to scale the weights for different joints. Once these values are tuned, we do not need to change them for different characters or environments.

Transitions determine allowable changes in control strategy within a single simulation time step. If the current control strategy is not applicable, then the state machine searches for an adjacent control strategy that is reachable through a transition. Our formulation is inspired by an approach described by Faloutsos et al. [2001]. The main difference is that in Faloutsos's work, a state has both a precondition for entering and a postcondition for leaving. In all our examples, a single condition suffices for the behaviors we wish to implement. At each simulation time step, the FSM searches for a control state in $\mathbf{S}$ that is applicable to the current dynamic state of the character and the environment state described by the condition D associated with each control state. The motion synthesizer then adds the state objectives to the current optimization.

To demonstrate the ease of controller design using our API, we show the implementation of a balance controller, a climb controller, and a swing controller in the next section.

5.2 Environment Knowledge

Realistic virtual characters incorporate sensory information about their surrounding environment to determine the appropriate action according to the desired task. We demonstrate the capability of our framework to model this behavior for a simplified synthetic sensory system. We endow our character with a sensor that can evaluate the reachability of environment objects and surfaces with respect to the character. For example, the character can regain balance using anything she can reach and grab onto in her immediate surroundings, such as handles, poles, or walls. In terms of the control algorithm, this entails augmenting the environment state with a list of objects that are reachable by the virtual character's end effectors. We can then specify control strategies where the condition incorporates information about the reachability of particular objects, and the objectives can describe desired spatial or derivative relationships between the character and object.

6. IMPLEMENTATION AND RESULTS

We now discuss the design of several controllers that enable the character to perform various actions in a varying environment.

The motion for all the examples discussed in this section is simulated at 2 to 10 frames/s on a single core of 2.93 GHz Intel Core 2 Duo processor. The variance in simulation time is primarily due to the complexity of the controllers. Full animations can be seen in the supplemental video available online in the ACM Digital Library. We used SNOPT [Gill et al. 1996] to solve the optimization problem at each time step. The time step used for simulation is 0.01s. Our framework does not require any motion capture sequences.

6.1 Balance Controller

Balancing is the basis for all locomotion tasks for bipedal characters. In this section, we describe the implementation of a balance controller using the controller specification described in Section 5.1. We start by describing a basic balancing strategy that allows the character to stand on the ground and maintain balance. We then enhance this basic control strategy with complex ones that allow the character to take protective steps when required, or to utilize nearby surfaces to recover balance. The same balance controller can be applied to different behaviors (dodging incoming objects, standing one foot), different physical models (child character), and different environments (balancing on the ice) by adding a few high-level objectives.

6.1.1 *Basic Balance Strategy.* We implemented a basic balance strategy with the following high-level objectives.

(1) Support the COM, $G_{cp} = \|proj(\mathbf{COM}(\mathbf{q}^{N+1})) - \mathbf{C}_{sp}(\mathbf{q}^{N+1})\|$, where **COM** is the center of mass, *proj*() projects a point to the ground, and $\mathbf{C}_{sp}$ is the center of support polygon evaluated at time sample $N+1$.
(2) Keep upper body upright, $G_{spine} = \|\mathbf{d}_{spine}(\mathbf{q}^{N+1}) \times \hat{j}\|$, where $\mathbf{d}_{spine}$ is spine orientation at time sample $N+1$ and $\hat{j}$ is the direction of gravity.
(3) Avoid sudden movements, $G_{qv} = \|\dot{\mathbf{q}}^{N+1}\|$.

We create a state representing the balance strategy, *balance*, with the weighted sum of objectives mentioned previously.

balance Objective: $5.0G_{cp} + 70.0G_{spine} + 0.5G_{qv}$
balance Condition: **if** COM is outside the support polygon, **return false**; **else return true**

This state machine comprising only one state with three simple strategies is capable of maintaining a balanced pose for the character, even under small perturbations (see supplemental video).

The exact same balance strategies also allow the character to balance on one foot. By reducing the supporting polygon to an arbitrarily chosen supporting foot, the character automatically shifts her weight toward the supporting foot. Once the character balances herself with one foot, we add kinematic objectives to make the character mimic one given pose (Figure 6(a)).

6.1.2 *Enhanced Balance Strategies.* The *balance* state fails when the COM of the character falls outside the support polygon. To handle this failure, we add two states, *relaxFoot* and *takeStep*, to the basic balance controller that enable the character to take protective steps, by automatically deciding when and where to place the foot for recovering balance. The more robust balance controller is represented next.

BALANCE Machine:
States:
balance, *relaxFoot*, *takeStep*
Transitions:
balance → *relaxFoot*
relaxFoot → *takeStep*
takeStep → *balance*

—*RelaxFoot.* The *relaxFoot* state is responsible for reducing the ground contact forces on the foot about to be lifted before taking a step. The decision of which foot is to be lifted depends on a simple heuristic that assumes the foot farther from the ground projection of the COM is easier to lift. To relax the forces on the foot, we add the following objectives.

(1) Move COM to the supporting foot, G_{cf}.
(2) Relax contact forces on the foot to be lifted, $G_c = \|\boldsymbol{\lambda}^N\|$, where $\boldsymbol{\lambda}^N$ are the contact force parameters for the contact points on the foot.

This helps the character to shift her weight away from the foot to be lifted and eventually reduce the contact forces. The rationale behind moving the COM towards the supporting foot comes from our observations of recorded human motion in which the COM accelerates towards the supporting foot before the subject breaks the contact from the other foot.

relaxfoot Objective: $0.2G_{qv} + 50.0G_{cf} + 1.0G_c$.
relaxfoot Condition: **if** contact forces on foot are relaxed, **return false**; **else return true**

—*TakeStep.* In the *takeStep* state, we have an objective to move the lifted foot to the desired position. The desired foot position is updated at each time step when the character is in this state. It is based on the simple heuristic that the COM should lie in the center of support polygon. Thus, the new foot position, $\mathbf{p}_f$, is chosen such that ground projection of COM lies midway between the feet. The objective G_p for moving a body point, $\mathbf{p}_i$, defined in local coordinates of body node i, to any desired position $\mathbf{p}_0$ is written as $G_p = \|\mathbf{W}_i^{N+1}\mathbf{p}_i - \mathbf{p}_0\|$. Thus, we substitute the desired position $\mathbf{p}_0$ in this equation by $\mathbf{p}_f$.

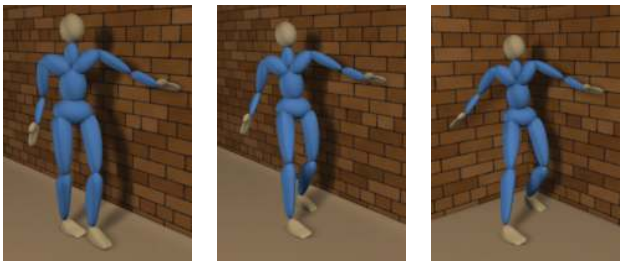
takeStep Objective: $3.0G_{qv} + 20.0G_{spine} + 0.8G_p$
takeStep Condition: **if** distance between the moving foot and the desired position is increasing, **return false**; **else return true**

(a) pushed backwards (b) trying to balance (c) taking step (d) still not balanced (e) taking additional step (f) finally balanced

Fig. 3. Character going through a series of states of the balance controller after she is pushed and finally balancing after taking a couple of steps (red arrow depicts the applied force).

(a) pushed backwards (b) slipping while trying to balance (c) taking small steps

Fig. 4. Character trying to balance on icy surface when pushed by taking small steps and slipping occasionally.



(a) nearer wall (b) further wall (c) two nearby walls

Fig. 5. Character's reaction to same push in different environment settings.

At each time step, progress of the moving foot is monitored and when the preceding condition fails that is, when the situation is getting worse as the character is not able to move her foot as fast as it should, the state machine decides to place the foot on the ground and transition to the *balance* state once again.

This completes one protective step to recover balance and if the *balance* fails again, this cycle is repeated; for example, in case of a strong push, the character has to take multiple steps to recover (see Figure 3).

To demonstrate the robustness of the balance controller, we just change the friction coefficient of the floor from 1.0 to 0.2 to model an icy surface. When the character tries to recover from a push on a slippery surface, she often slips and cautiously takes smaller steps to reduce this slipping (see Figure 4).

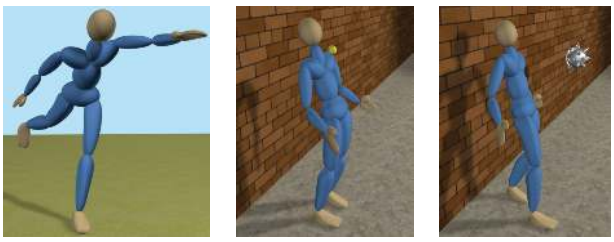
6.1.3 *Support Using Environment Features.* By incorporating the balance controller with information from a synthetic visual sensory system, we develop a balance controller that synthesizes significantly different motions in response to different environmental stimuli.

The visual sensory component takes as input the character's current configuration and the current state of all objects in the environment, and outputs a list of objects reachable by any end effector on the character's body. We define an end effector e_i as a point on the body that can be used for support (e.g., a hand or a foot) against a reachable object, o_j. For each (e_i, o_j) pair, the character evaluates whether o_j is reachable by e_i. If so, we add an objective in the motion synthesizer that moves e_i towards o_j.

We demonstrate that the character autonomously determines to use the nearby wall for support when pushed by a large force. The character automatically decides when to move her hand for support and tries to reach for the nearest point on the wall (projection of her hand on the wall). By increasing the distance between the wall and the character, she takes extra steps before reaching the wall. The reaction changes significantly when there are two walls available for support (Figure 5). When there are multiple available contact points, the character simply reaches for the closest one. Nonetheless, more sophisticated strategies can be encoded in the controller. We refer the reader to the supplemental video for full animations. This capability of using environment features for support is added on top of the same balance controller as described earlier.

6.1.4 *Balance and Dodge.* To create more interesting behavior in a dynamically varying environment, the programmer can add simple high-level objectives to the balance controller. For example, we create a behavior where a character dodges objects thrown at her while maintaining balance. We add a dodging objective that keeps the character's body parts away from the object. For an object at position $\mathbf{p}$ and traveling with velocity $\mathbf{v}$, we add objectives to maximize (by putting negative weight for objective) distance of some points p_i (defined in the local frame of body node i) on the body which lie near to the line $\mathbf{l}$ passing through $\mathbf{p}$ with direction $\mathbf{v}$ (parametric representation $\mathbf{l}(t) = \mathbf{p} + \mathbf{v}t$). Thus, the objective can be written as $G_{dodge} = dist(\mathbf{W}_i^{N+1}\mathbf{p}_i, \mathbf{l})$, where $dist()$ evaluates the distance of a point $\mathbf{p}_i$, when expressed in world coordinates, to the line $\mathbf{l}$. We weight the dodge objectives as inversely proportional to the distance to the line (-0.05 to -0.15) and these are active when the object is within a certain distance (e.g., 1 meter) from the character.

In addition, by adjusting the relative importance of each objective, the same controller can produce a variety of behaviors. In the synthesized example, the character easily dodges a tennis ball by bending her spine (Figure 6(b)) but gets hit on the arm by the object coming from behind. The programmer can adjust the importance of the dodging objective based on the incoming object. If the character sees a flying object that appears harmful, she quickly moves out of way (Figure 6(c)). To realize this, we changed the



(a) balancing on one foot (b) dodging a less harmful object (c) dodging a more harmful object

Fig. 6. Character performing variety of tasks while balancing.

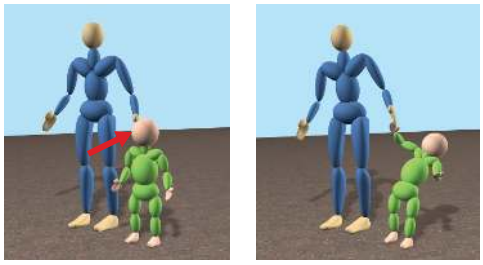


Fig. 7. Adult character preventing the child character from falling by holding hands when the child character is pushed.

weights of objective function in the *balance* state (Section 6.1.1) to $2.0G_{cp} + 30.0G_{spine} + 0.5G_{qv}$ for a harmful object.

Better dodging strategies or hazard assessments rely on domain knowledge in the controller design. Zordan et al. [2007] learn different anticipation strategies from motion capture data and automatically choose which to employ at runtime according to a damage and energy assessment calculated from simulation results. Their results exhibit a wide variety of dodging strategies with compelling realism. Metoyer et al. [2008] use psychological insights and motion capture data to formulate protective anticipatory movement parameterized by a model of an approaching object. This is combined with a physically-based dynamic response to produce animations with anticipation and reaction to impacts. We do not aim at thoroughly solving a particular problem of anticipation, but rather emphasize the ease of designing control strategies with limited domain knowledge.

6.1.5 *Multiple Character Interaction.* The same balance controller can operate across characters with different mass distribution and skeletal structures. In addition, the programmer can simulate interactions between multiple characters by adding objectives or constraints that model the physical contacts. We synthesize a child reaching out for an adult's hand for support in the event of balance loss (see Figure 7). The natural reaction for both the characters is synthesized automatically, since the objective of holding hands and force exchange affects both characters' joint configuration.

All the examples described in this section took 2 to 4 frames/s to simulate. The slow simulation speed is due to the complexity of the balance problem as objectives conflict and compete with each other in the optimization.

6.2 Climb Controller

We next move to a control task that requires a much more sophisticated interplay between character and environment. We implement a climbing controller that facilitates wall climbing using attached holds. The holds are placed at random positions and the character automatically decides which ones to grab in order to progress upwards. A complex wall climbing motion can be generated by specifying kinematic constraints at the hand holds and foot holds.

We create five states, *allSupport*, *relaxHand*, *moveHand*, *relaxFoot*, and *moveFoot* (discussed in the following), and define the state machine for the climb controller as follows.

CLIMB Machine:
States:
allSupport, *relaxHand*, *moveHand*, *relaxFoot*, *moveFoot*
Transitions:
allSupport → *relaxHand*
relaxHand → *moveHand*
moveHand → *allSupport*
allSupport → *relaxFoot*
relaxFoot → *moveFoot*
moveFoot → *allSupport*

—*AllSupport.* In this state, the character grabs both the hand holds and places her feet on the foot holds. This state consists of following objectives:
(1) to raise her COM to the highest possible position, so that she is comfortable to stretch out her hand and grab the next hand hold (the objective for the COM can be written as $G_{com} = \|\mathbf{COM}(\mathbf{q}^{N+1}) - \mathbf{C}_0\|$, where $\mathbf{C}_0$ is the center of hand holds that is high enough for the COM to reach); and
(2) to reduce the joint velocities for smooth movements, G_{qv}.

allSupport Objective: $0.2G_{qv} + 20.0G_{com}$
allSupport Condition: **if** the character is stable, **return false**; **else return true**

—*RelaxHand.* The goal of the character is to relax the contact forces on the hand so that it can release the hold and move to the desired position. We achieve this by setting objectives for reducing the contact forces from the corresponding hand hold (*moveFoot* has similar objectives for relaxing the contact forces on the foot).

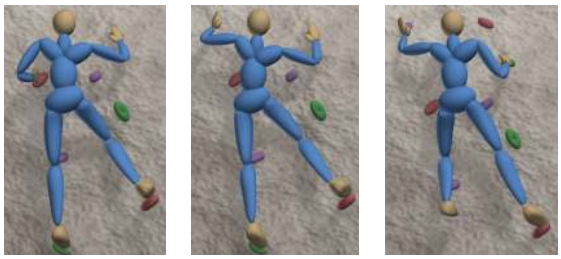
relaxHand Objective: $0.2G_{qv} + 1.0G_c$
relaxHand Condition: **if** the contact forces fall below a small threshold, **return false**; **else return true**

—*MoveHand.* In this state, the controller adds the position of next nearest hand hold, along with minimization of joint velocities as control objectives (see Figure 8(b)).

moveHand Objective: $0.2G_{qv} + (0.1 \text{ to } 0.25)G_p + 10.0G_{com}$
moveHand Condition: **if** hand reaches the hand hold, **return false**; **else return true**

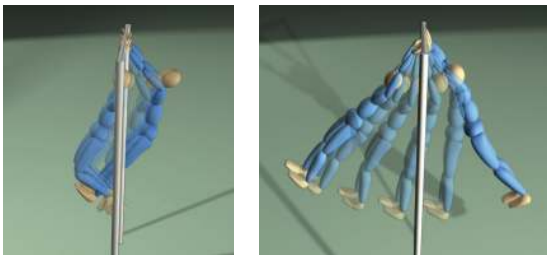
We vary the weights for G_p according to the distance of the hand to the desired position (higher weight when nearer). When the condition fails, that is, when the hand reaches the hand hold, grasping contact is established and the state transitions to *allSupport*. Next, to raise her body up and move her foot, the character relaxes the forces on her foot by making a transition to *relaxFoot* state. Once relaxed, she moves to *moveFoot* state.

—*MoveFoot.* In this state, the position of the next foot hold is set as an objective for the moving foot and the character begins to move her foot to the desired position (see Figure 8(c)). When the foot reaches close to the hold, we add an objective,



(a) *allSupport* (b) *moveHand* (c) *moveFoot*

Fig. 8. Character in different states of the climb controller.



(a) start from rest position (b) start from high angle

Fig. 9. Character swinging with different initial swing angle.

$G_{pose} = \|\mathbf{q}^{N+1} - \mathbf{q}_0\|$, to guide the joint angles close to the starting pose, $\mathbf{q}_0$, to make the character assume a realistic-looking pose. This starting pose (see Figure 8(a)) is created using simple inverse kinematics.

moveFoot Objective: $0.2G_{qv} + (0.01 \text{ to } 0.30)G_p + 10.0G_{com} + 2.0G_{pose}$
moveFoot Condition: **if** foot reaches the foot hold, **return false**; **else return true**

This completes one cycle of the state machine which moves the character up by one hold and makes her reach a stance similar to the starting pose. By looping over this cycle, the character can climb an arbitrary number of holds.

In the synthesized example for climbing a wall (see supplemental video), the programmer only needed to specify the starting pose for the character, designed using simple inverse kinematics, and the placement of holds on the wall. With the help of simple strategies and kinematic constraints as described before, the character automatically climbs up the wall using the required amount of external contact forces from these randomly placed hand and foot holds. The simulation rate for this example varied from 5 to 10 frames/s.

6.3 Swing Controller

Our framework facilitates synthesis of natural motion by defining a few high-level objectives. Thus, it can serve as a testbed for designing new motor skills. In this section, we describe a simple swing controller based on only one objective function: Maximize the center of mass velocity in the direction tangential to her movement. The character starts from a rest pose holding a high bar with both hands. We create a simple state machine *SWING* consisting of two states, *trySwing* and *passiveSwing*.

In the *trySwing* state, the character tries hard to increase her COM velocity in the direction perpendicular to the plane joining her COM and grasps onto the bar. The objective for increasing the COM velocity is defined as $G_{cv} = \|\dot{\mathbf{COM}}(\mathbf{q}^{N+1}) - \mathbf{v}_d\|$, where $\mathbf{v}_d$ is the desired velocity (a value more than the current COM velocity).

trySwing Objective: $0.1G_{qv} + 4.0G_{cv}$
trySwing Condition: **if** the angle of swing increases a threshold, **return false**; **else return true**

When the angle of swing increases above a threshold, a transition to *passiveSwing* occurs. This state's objective is to maintain constant velocity in the perpendicular direction to create a smooth passive swing.

Based on only one objective function and no knowledge in gymnastics, the character tries hard to increase her velocity and is able to start swinging. However, she is not able to increase her angle of swing beyond a certain limit because of the lack of coordination of her joints and skills possessed by gymnasts (see Figure 9(a)). When started from a higher angle, the character still fails to maintain the momentum and the velocity diminishes rapidly.

Adding some more objectives to improve the coordination of joints helps the character maintain her velocity. To achieve this, we added joint position objectives to make her body more stiff and legs straightened, then let her start from a higher swing angle. These objectives are meant to keep her legs close to a specific pose (straight legs) and the stiffness is achieved by setting a relatively higher weight for these objectives.

The objective function now becomes $0.1G_{qv} + 15.0G_{cv} + 5.0G_{pose}$. Specifically, G_{pose} is responsible for stiffening the body and straightening the legs. We do not add pose objectives for DOFs of the abdomen to allow easier bending of the abdomen.

The character is now able to swing more smoothly and maintain her velocity (see Figure 9(b)). The examples were synthesized at 5 to 10 frames/s.

6.4 Composition of Multiple Controllers

Primitive controllers can be easily composed to create an autonomous and versatile virtual character. The cable car example (Figure 10) highlights realistic behaviors and responsive reactions of the character to unexpected events in a dynamically varying environment. Initially, the character comfortably counteracts small disturbances of the cable car with her balance strategies. When the car shakes violently, she decides to take protective steps and grab onto nearby walls and bars to prevent herself from falling. When the ground breaks, she resorts to holding the bar and applying her swing motor skills to hang on (see supplemental video). The programmer just key-framed the events, like rocking the cable car and breakage of walls and the floor, and the character autonomously decides what and when to grab, depending on her immediate surroundings and her dynamic configuration.

7. DISCUSSION

We have described a new approach to synthesize reactive virtual characters in a physical environment based on constrained optimization. Our approach provides a generic framework for rapidly designing a variety of controllers by formulating high-level objectives and tasks. This approach gives us the following advantages.

(1) Our goal-driven formulation of control strategies expedites the design of physics-based motion controllers, enabling the programmer to rapidly create a wide range of motion repertoires for virtual characters.
(2) The controllers designed in this framework can be robustly adapted to different virtual characters (e.g., adult or child

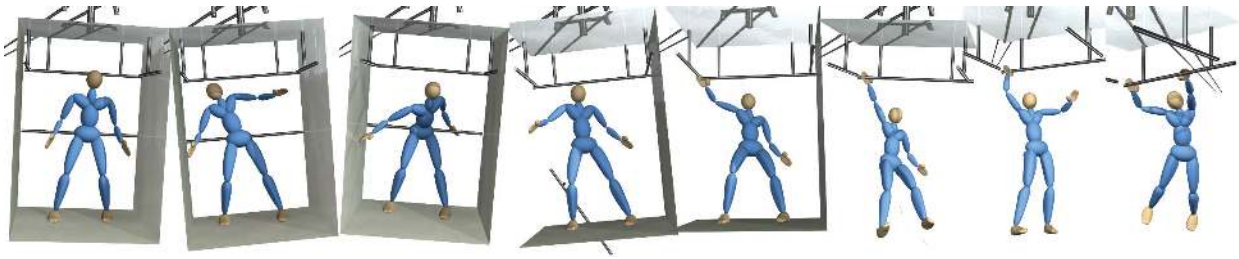


Fig. 10. An autonomous character reacting to unexpected events inside a cable car and trying hard to prevent herself from falling.

characters) and environments (e.g., slippery surface, cable car settings, etc.).

The design of our approach raises some important issues and questions in practice. First, what degree of physical realism can the system provide when the user-specified objectives are conflicting or unrealistic? Second, is the weight adjustment in the objective function any easier than parameter tuning for designing robotic controllers? Third, what types of motions/tasks are most appropriate to our framework?

7.1 Physical Realism

We enforce exact equations of motion on the global dynamics of the character, that is, at the root of the character's hierarchy where no actuators exist. These equations (Eq. (7)) alongside the contact model (Section 4.2) are physically correct up to discretization error, similar to numerical integration methods used for forward simulation.

For all the other joints, we do not enforce these equations, implying that the actuators at these joints can assume arbitrary values. However, we add an additional objective for minimizing the change in joint torques (see Section 4.1). This restricts arbitrary changes in joint torques, leading to smooth and plausible muscle forces. We chose not to enforce explicit joint torque limits (constraints) in the optimization because, in practice, the values of these torques remain within reasonable limits. Therefore, removing these constraints helps reduce the computation time without affecting the output motion.

The ratio of the weight of muscle minimization to the weight of kinematic objectives specified by the programmer indicates the responsiveness of the character to the control goals. The user can adjust the ratio to explore the trade-off between the naturalness of the movement and the satisfaction of control goals. However, global physical realism is ensured regardless of the value of this ratio.

7.2 Weight Adjustment

Our framework facilitates easy design of control strategies for articulated characters (see Section 5). The parameters requiring to be tuned in our framework are the weights associated with each objective function used in the optimization. These objectives indicate high-level behaviors that can be described by the joint angles, such as basic balance strategies or reaching for objects. Consequently, the user can express controllers at kinematic level without knowing the mechanical details of each joint. For example, the user only needs to tune two weights to maintain a supported center of mass and to achieve a specific location for an end effector. In other methods such as robotic controller framework, the physical parameters for each joint need to be individually tuned and tested. Often, the values of those parameters cannot be easily translated to a high-level task description. Moreover, when the physical properties of the articulated body system change, readjusting physical parameters is likely required. In our framework, however, the same weights can be reused as long as the relative importance of the objectives remains the same.

7.3 Suitable Range of Tasks

In principle, with careful design and sufficient engineering effort, most tasks demonstrated by our framework can also be achieved by robotics controllers. However, we believe our method significantly reduces the engineering effort for the following types of tasks.

(1) *Tasks that Require Precise Positional Control.* For example, a hand reaching out for a moving point in space requires such control. Our method directly imposes positional control as objectives or hard constraints, rather than employing additional inverse kinematics and inverse dynamics computation to obtain the required joint torques. This type of control was frequently applied in our examples, such as step taking in the balance controller (Section 6.1.2).

(2) *Tasks that are Highly Constrained by the Environment through Resting Contacts.* In general, having more resting contacts complicates the computation in a dynamic system. Our approach, on the contrary, works particularly well with multiple resting contacts. This is because contacts introduce additional degrees of freedom, contact forces, in the optimization that "help" the character meet the various objectives with ease. Furthermore, contacts provide additional kinematic hints for solving an underdetermined joint configuration.

(3) *Multi-Objective Tasks with Conflicting Objectives.* Our optimization method resolves the trade-offs between conflicting objectives simultaneously with other dynamic and kinematic constraints imposed on the character. This flexible framework allows the programmer to compose simple tasks in the position domain to create complex behaviors. For example, dodge and balance tasks (Section 6.1.4) have conflicting preferred joint configurations. The programmer only needs to tune the weights of these two objectives to arrive at a solution satisfying both. In our experience, there is a wide range of weights that achieve the goal.

There are certain situations where our framework does not offer many advantages and where use of other approaches might be more appropriate.

(1) Existing forward simulation methods score over our approach in two situations. First, when the character does not exhibit active control in the motion (e.g., ragdoll), our optimization formulation adds unnecessary computation to a relatively trivial simulation problem. Second, when the motion involves frequent passive colliding contacts (e.g., falling off the stairs), our approach becomes very inefficient because each contact

increases the number of constraints and expands the dimension of degrees of the freedom in the optimization.

(2) We sacrifice the anticipatory property of space-time optimization for interactivity. Our system can only generate anticipatory motion enforced by kinematic constraints, such as changing the kinematic goals gradually, but is not able to create natural anticipation and follow-through involving a change of dynamics, such as a broad jump.

8. FUTURE WORK

We plan to incorporate motion planning into our framework by optimizing over a short window of time in the future, which would also help in controlling the timing of the motion. One potential approach is to combine our framework with motion capture data in a similar manner as the tracking techniques introduced by Zordan and Hodgins [1999]. We are particularly interested in a recent work of da Silva and his colleagues [da Silva et al. 2008], in which the joint torques that achieve the captured motion are solved by a short-horizon optimization.

The objectives in the optimization are required to be weighted relative to each other in order to correctly emphasize and de-emphasize various objectives coexisting in the motion synthesizer. The programmer has the freedom to choose a set of weights for the objectives that help meet his goals better. Although choosing weights in our framework is relatively easier than in other approaches, the right and robust choice for such weights requires domain knowledge to fulfill the task. One promising extension to our work would be to automatically learn the most robust set of objective functions and their weights from the motion capture data.

We demonstrated some preliminary results on synthesizing two-character interaction through contact forces in a collaborative manner. We are also interested in exploring the coupling between our framework and a physics-based simulation of passive systems, such as a spring-mass or fluid system. For example, our framework could be used to rapidly design and test swimming control strategies in various fluid conditions.

APPENDIX

A. LAGRANGIAN FORMULATION

The generalized forces Q_j^g and Q_j^{ext} incorporating gravity force $\sum m_i \mathbf{g}$ and user input force $\mathbf{F}_{ext}$, acting on body node k at point $\mathbf{p}_k$ in its local coordinate frame, respectively, are given by

$$Q_j^g = \sum_{i \in N(j)} \left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{c}_i \right) . (m_i \mathbf{g}), \tag{15}$$

$$Q_j^{ext} = \left(\frac{\partial \mathbf{W}_k}{\partial q_j} \mathbf{p}_k \right) . \mathbf{F}_{ext}, \tag{16}$$

where $\mathbf{c}_i$ is the center of mass (COM) of body node i in its local coordinate frame.

The mass tensor $\mathbf{M}_i$ for a body node i is defined as

$$\mathbf{M}_i \equiv \int \int \int \rho \mathbf{x}\mathbf{x}^T dx\, dy\, dz, \tag{17}$$

where an infinitesimal point $\mathbf{x} \equiv (x, y, z, 1)^T$ in the local coordinates of body node i has mass density ρ.

B. LINEARIZATION OF CONSTRAINTS

Our optimization framework deals with constraints that, in general, are nonlinear in $\mathbf{q}^{N+1}$ and λ^N.

However, solving a constrained nonlinear optimization problem is slow, in general, as compared to solving a quadratic programming (QP) problem which consists of linear constraints and quadratic objectives. Thus, it is desirable to have as many linear constraints as possible for the solver. Any constraint $\mathbf{C} = 0$ can be used as an objective function, for example, as $\mathbf{C}^T\mathbf{C}$ or $(\|\mathbf{C}\|^2)$ to be minimized in the optimization along with other objectives. Thus, a linear constraint can be used as a quadratic objective in a QP problem.

(1) *Lagrange's Constraint.* Eq. (7) gives the discretized Lagrange's constraint as a function of $\mathbf{q}^{N+1}$ (by using Eq. (6) and Eq. (4)) and λ^N. From Eq. (13), we see that the generalized contact force Q_j^c is linear in λ^N. The constraint is also linear in $\ddot{\mathbf{W}}_i$'s, which are functions of $\mathbf{q}^{N+1}$. Now if we use the definition of joint velocity as in Eq. (3), Lagrange's constraint becomes quadratic in $\mathbf{q}^{N+1}$. However, if we use the definition in Eq. (8), the constraint becomes linear in $\mathbf{q}^{N+1}$. This motivates us to sacrifice the second-order accuracy for a practical speedup in solving an optimization. Note that this makes the corresponding objective (as used in Section 4.1) quadratic.

(2) *Position Constraint.* A position constraint $\mathbf{C}_P$ which fixes the position of some point $\mathbf{p}_l$ on body node i to a world position $\mathbf{p}_0$ (i.e., $\mathbf{C}_P = \mathbf{W}_i^{N+1}\mathbf{p}_l - \mathbf{p}_0 = 0$) is nonlinear in $\mathbf{q}^{N+1}$. We linearize it by approximating the position of this point by using its position at current time sample N and velocity at time sample $N+1$.

$$\begin{aligned} \mathbf{C}_P(\mathbf{q}^{N+1}) &= \mathbf{W}_i^{N+1}\mathbf{p}_l - \mathbf{p}_0 \\ &\approx \mathbf{W}_i^N \mathbf{p}_l + \dot{\mathbf{W}}_i{}^{N+1}\mathbf{p}_l \Delta t - \mathbf{p}_0 \\ &= \mathbf{W}_i^N \mathbf{p}_l + \frac{\partial \mathbf{W}_i^{N+1}}{\partial \mathbf{q}^{N+1}} \dot{\mathbf{q}}^{N+1} \mathbf{p}_l \Delta t - \mathbf{p}_0 \\ &\approx \mathbf{W}_i^N \mathbf{p}_l + \frac{\partial \mathbf{W}_i^N}{\partial \mathbf{q}^N} \dot{\mathbf{q}}^{N+1} \mathbf{p}_l \Delta t - \mathbf{p}_0 \end{aligned} \tag{18}$$

Similarly, other functions of positions (e.g., COM position constraint) which can be computed as a linear combination of COMs of individual body nodes can be approximated in this fashion.

ACKNOWLEDGMENTS

We would like to thank H. Chong, S. Hardegree and M. Kuo for proofreading the manuscript.

REFERENCES

ABE, Y., DA SILVA, M., AND POPOVIĆ, J. 2007. Multiobjective control with frictional contacts. In *Proceedings of the Eurographics/SIGGRAPH Symposium on Computer Animation*, 249–258.

ABE, Y. AND POPOVIĆ, J. 2006. Interactive animation of dynamic manipulation. In *Proceedings of the Eurographics/SIGGRAPH Symposium on Computer Animation*.

COHEN, M. F. 1992. Interactive spacetime control for animation. In *SIGGRAPH*. Vol. 26, 293–302.

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

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

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

GILL, P., SAUNDERS, M., AND MURRAY, W. 1996. Snopt: An SQP algorithm for large-scale constrained optimization. Tech. rep. NA 96-2, University of California, San Diego.

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

ISAACS, P. M. AND COHEN, M. F. 1987. Controlling dynamic simulation with kinematic constraints. *SIGGRAPH*, 215–224.

KAWATO, M. 1999. Internal models for motor control and trajectory planning. In *Current Opinions in Neurobiology*, Vol. 9.

KUDOH, S., KOMURA, T., AND IKEUCHI, K. 2006. Stepping motion for a human-like character to maintain balance against large perturbations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2661– 2666.

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

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

LIU, C. K. 2008. Synthesis of interactive hand manipulation. In *Proceedings of the Eurographics/SIGGRAPH Symposium on Computer Animation*.

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

LIU, C. K. AND POPOVIĆ, Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Trans. Graphics 21,* 3, 408–416.

LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical spacetime control. *SIGGRAPH*, 35–42.

LOCKHART, D. B. AND TING, L. H. 2007. Optimal sensorimotor transformations for balance. *Nat Neurosci 10*, 1329–1336.

MACIEJEWSKI, A. A. AND KLEIN, C. A. 1985. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *Int. J. Robotics Res. 4,* 3, 109–117.

METOYER, R., ZORDAN, V., HERMENS, B., WU, C.-C., AND SORIANO, M. 2008. Psychologically inspired anticipation and dynamic response for impacts to the head and upper body. *IEEE Trans. Visualization Comput. Graphics 14,* 1, 173–185.

NATURALMOTION. 2006. Endorphin. www.naturalmotion.com.

POPOVIĆ, Z. AND WITKIN, A. 1999. Physically based motion transformation. *SIGGRAPH*, 11–20.

RAIBERT, M. H. 1986. *Legged Robots That Balance*. Massachusetts Institute of Technology, Cambridge, Massachusetts.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensinal, behavior-specific spaces. *ACM Trans. Graphics 23,* 3, 514–521.

SENTIS, L. AND KHATIB, O. 2005. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *Int. J. Humanoid Robotics 2,* 4, 505–518.

SENTIS, L. AND KHATIB, O. 2006. A whole-body control framework for humanoids operating in human environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2641–2648.

SHARON, D. AND VAN DE PANNE, M. 2005. Synthesis of controllers for stylized planar bipedal walking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

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

STEWART, A. J. AND CREMER, J. F. 1992b. Beyond keyframing: An algorithmic approach to animation. In *Graphics Interface*, 273–281.

SULEJMANPAŠIĆ, A. AND POPOVIĆ, J. 2004. Adaptation of performed ballistic motion. *ACM Trans. Graphics 24,* 1.

UNO, Y., KAWATO, M., AND SUZUKI, R. 1989. Minimum muscle-tension-change model which reproduces human arm movement. In *Proceedings of the Symposium on Biological and Physiological Engineering*, 299–302.

VAN DE PANNE, M. AND LAMOURET, A. 1995. Guided optimization for balanced locomotion. In *Computer Animation and Simulation*, 165–177.

WITKIN, A. AND KASS, M. 1988. Spacetime constraints. *SIGGRAPH*. *22*, 159–168.

WOOTEN, W. L. 1998. Simulation of leaping, tumbling, landing, and balancing humans. Ph.D. thesis, Georgia Institute of Technology.

YAMANE, K. AND NAKAMURA, Y. 2000. Dynamics filter—Concept and implementation of on-line motion generator for human figures. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 688–695.

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

ZORDAN, V., MACCHIETTO, A., MEDIN, J., SORIANO, M., WU, C.-C., METOYER, R., AND ROSE, R. 2007. Anticipation from example. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'07)*. 81–84.

ZORDAN, V. B. AND HODGINS, J. K. 1999. Tracking and modifying upper-body human motion data with dynamic simulation. In *Conference on Computer Animation and Simulation*.

Received April 2008; revised October 2008; accepted December 2008