Composable Controllers for Physics-Based Character Animation

Petros Faloutsos[1]   Michiel van de Panne[2,1]   Demetri Terzopoulos[3,1]

[1]University of Toronto, Department of Computer Science
[2]Motion Playground, Inc.
[3]New York University, Courant Institute, Computer Science Department

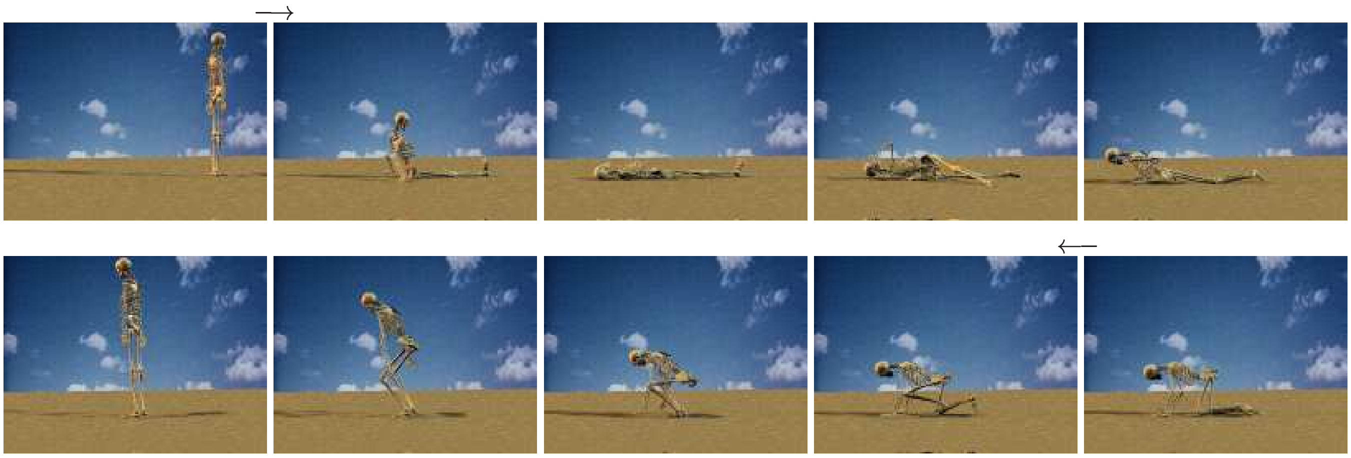


Figure 1: A dynamic "virtual stuntman" falls to the ground, rolls over, and rises to an erect position, balancing in gravity.

Abstract

An ambitious goal in the area of physics-based computer animation is the creation of virtual actors that autonomously synthesize realistic human motions and possess a broad repertoire of lifelike motor skills. To this end, the control of dynamic, anthropomorphic figures subject to gravity and contact forces remains a difficult open problem. We propose a framework for composing controllers in order to enhance the motor abilities of such figures. A key contribution of our composition framework is an explicit model of the "pre-conditions" under which motor controllers are expected to function properly. We demonstrate controller composition with pre-conditions determined not only manually, but also automatically based on Support Vector Machine (SVM) learning theory. We evaluate our composition framework using a family of controllers capable of synthesizing basic actions such as balance, protective stepping when balance is disturbed, protective arm reactions when falling, and multiple ways of standing up after a fall. We furthermore demonstrate these basic controllers working in conjunction with more dynamic motor skills within a prototype virtual stuntperson. Our composition framework promises to enable the community of physics-based animation practitioners to easily exchange motor controllers and integrate them into dynamic characters.

Keywords: Computer Animation, Character Animation, Physics-Based Animation Control, Physics-Based Modeling

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

[0]To appear in the Proceedings of SIGGRAPH 2001 (Los Angeles, CA, August 12–17, 2001). In *Computer Graphics* Proceedings, Annual Conference Series, 2001, ACM SIGGRAPH, in press.

1 Introduction

Despite the considerable history of progress in animating virtual humans [3, 7], physics-based animated characters with a large repertoire of motor skills have so far been elusive. This may seem surprising in view of the recent successes in implementing a slew of specialist controllers capable of realistically synthesizing the complex dynamics of running, diving, and various gymnastic maneuvers [16].

While a divide-and-conquer strategy is clearly prudent in coping with the enormous variety of controlled motions that humans and other animals may perform, little effort has been directed at how the resulting control solutions may be integrated to yield composite controllers with significantly broader functionalities. For example, if researcher A creates a walking controller for a dynamic character while researcher B creates a running controller for the same articulated model, it would be beneficial if they could share their controllers (perhaps through an e-mail exchange) and easily create a composite controller enabling the character to both walk and run. This is a difficult problem, but its resolution would help pave the way towards controller libraries for dynamic animation which communities of practitioners could utilize and to which they could contribute.

In this paper, we propose a simple yet effective framework for composing specialist controllers into more general and capable control systems for dynamic characters. In our framework, *individual controllers* are black boxes encapsulating control knowledge that is possibly gleaned from the biomechanics literature, derived from the robotics control literature, or developed specifically for animation control. Individual controllers must be able to determine two things: (1) a controller should be able to determine whether or not it can take the dynamic character from its current state to some desired goal state, and (2) an active controller should be able to determine whether it is operating nominally, whether it has succeeded, or whether it has failed. Any controller that can answer these queries may be added to a pool of controllers managed by a *supervisor controller* whose goal is to resolve more complex control tasks.

An important technical contribution within our controller composition framework is an explicit model of *pre-conditions*. Pre-conditions characterize those regions of the dynamic figure's state space within which an individual controller is able to successfully carry out its mission. Initially, we demonstrate the successful composition of controllers based on manually determined pre-conditions. We then proceed to investigate the question of whether pre-conditions can be determined automatically. We devise a promising solution which employs Support Vector Machine (SVM) learning theory. Our novel application of this technique learns appropriate pre-conditions through the repeated sampling of individual controller behavior in operation.

As a testbed of our techniques, we are developing a physically-simulated animated character capable of a large repertoire of motor skills. An obvious application of such a character is the creation of a *virtual stuntperson*: the dynamic nature of typical stunts makes them dangerous to perform, but also makes them an attractive candidate for the use of physics-based animation. The open challenge here lies in developing appropriate control strategies for specific actions and ways of integrating them into a coherent whole.

In this paper, we demonstrate families of composable controllers for articulated skeletons whose physical parameters reflect anthropometric data consistent with a fully-fleshed adult male. One family of controllers is for a 37 degree-of-freedom (DOF) 3D articulated skeleton, while a second family of controllers has been developed for a comparable 16 DOF 2D articulated skeleton. While the 3D skeleton illustrates the ultimate promise of the technique, the easier control associated with the 2D skeleton allows for more rapid prototyping of larger families of controllers and more careful analysis of their operation. As has been recognized in the robotics literature, the control of broad skilled repertoires of motion remains very much an open problem even for 2D articulated figures.

Fig. 1 illustrates the 3D dynamic character autonomously performing a complex control sequence composed of individual controllers responsible for falling reactions, rolling-over, getting up, and balancing in gravity. The upright balancing dynamic figure is pushed backwards by an external force; its arms react protectively to cushion the impact with the ground; the figure comes to rest in a supine position; it rolls over to a prone position, pushes itself up on all fours, and rises to its feet; finally it balances upright once again. A subsequent disturbance will elicit similar though by no means identical autonomous behavior, because the initial conditions and external forces will usually not be exactly the same. Control sequences of such intricacy for fully dynamic articulated figures are unprecedented in the physics-based animation literature.

After reviewing related prior work in the next section, we present the details of our control framework in Section 3. We then investigate the question of determining pre-conditions in Section 4. Section 5 describes the articulated figure models and the software system we use to implement the control framework. Section 6 presents the details of the example in Fig. 1 along with several other examples that demonstrate the effectiveness of our framework. Section 7 concludes the paper and discusses avenues for future research opened up by our work.

2 Previous Work

The simulation and animation of human characters is a challenging problem in many respects. Comprehensive solutions must aspire to distill and integrate knowledge from biomechanics, robotics, control, and animation. Models for human motion must also meet a particularly high standard, given our familiarity with what the results should look like. Not surprisingly, a divide-and-conquer strategy is evident in most approaches, focusing efforts on reproducing particular motions in order to yield a tractable problem and to allow for comparative analysis.

The biomechanics literature is a useful source of predictive models for specific motions, typically based on experimental data supplemented by careful analysis. These models target applications such as medical diagnosis, the understanding and treatment of motor control problems, the analysis of accidents and disabilities, and high-performance athletics. Computer simulation is becoming an increasingly useful tool in this domain as the motion models evolve to become more complex and comprehensive [26, 27, 29]. Given the challenge of achieving high-fidelity motion models for individual motions, there have been fewer efforts towards integrated solutions applicable to multiple motions. Reference [26] is one such example.

Robotics research has made remarkable progress in the successful design of a variety of legged robots [28] and, more recently, bipedal robots with anthropomorphic aspirations [23]. Despite their limited motion repertoires and rather deliberate movements, these robotic systems are truly engineering marvels. The work in [1] provides a good summary of behavioral architectures explored in the context of robotics. A 3 DOF ball-juggling robot is described in [6] which uses a theory of behavior composition, although the practicality of extending the method to high-DOF dynamic models of human motions is unclear.

Computer animation is to a large extent unencumbered by the exacting fidelity requirements of biomechanical models and the mechanical limitations of robotic systems. This has spawned a great variety of kinematic and dynamic models for character motion [3, 4, 7]. While motion capture solutions based on blending and warping techniques may give satisfactory results for such tasks in the short term, controller based approaches reveal more about the physics, planning, and control of such motions and they therefore serve as a basis for more general solutions. Dynamically simulated characters were first proposed over 15 years ago [2, 34] and since then have progressed in sophistication in a variety of directions. Controllers have been successfully designed for specific human motions such as walking, running, vaulting, cycling, etc. [16, 22, 35].

Dynamically simulated articulated characters equipped with an integrated, wide-ranging repertoire of motor skills currently remain an unachieved goal. Some positive steps in this direction are evident, however. Examples include an integrated repertoire of motor controllers for biomechanically animated fish [30], a methodology for controller design and integration applicable to simple figures [32], a demonstration of successful integration for selected diving and gymnastic motions [35], and adapting a controller designed for one character to work on another character [17]. The work of Wooten [35] is the most relevant as an example of a sequence of successive transitions between several controllers for human motions such as leaping, tumbling, landing, and balancing. Transitions are realized by including the end state of some controllers in the starting states of other controllers. A *digital biomechanics laboratory* is proposed by Boston Dynamics, Inc. [20] as a tool for simulating a wide range of human motion. This currently remains ambitious work in progress.

Our work is aimed at creating dynamic human characters with broadly integrated action repertoires. Unlike previous work focusing on specific athletic movements, our methodology is to begin with a core set of simple actions, including balancing, small steps, falling reactions, recovery from falls, standing up from a chair, and others. In the present paper, we do not cover in any appreciable detail the design of individual controllers to effect such basic actions.[1] Rather, our contribution here is a framework for composing individual controllers, however they may be designed, into more capable control systems for dynamic characters. An interesting tech-

[1] Full details about the individual controllers that we have designed are presented elsewhere [10].

nical contribution within our controller composition framework is the introduction of a learning approach for automatically determining controller pre-conditions. Our pre-condition learning algorithm adds to the growing body of learning algorithms that have been successfully applied in the context of computer animation in recent years [14, 15].

3 Controller Composition Framework

In our controller composition framework, we consider individual controllers as black boxes which are managed by a simple supervisor controller. When no controller is active, the supervisor polls the pool of controllers, querying each whether it can handle the transition of the dynamic character from its current state to the desired goal state. Individual controllers return an integer confidence/suitability score when queried in order to bid on becoming the active controller. In our implementation, controllers that can perform a sensible action given the current state of the character return an integer in the range $[1, 10]$, while those that can handle the current state as well as guarantee a transition to the desired state, return an integer in the range $[10, 20]$. Lastly, a value of 0 means that a controller is unsuited for the current state. The controller that returns the highest score becomes active. While this scoring scheme potentially allows for a nuanced evaluation of the controller suitability in terms of criteria such as probability of success or energy used, our current controllers resort to a simpler scheme. This consists of a binary success/failure evaluation multiplied by a weighting factor assigned to each controller that serves to establish a relative preference ordering.

The power of this scheme stems from the following attributes:

- *Simplicity*: The composition method is straightforward and easy to implement. It does not appreciably burden the controller design task.
- *Generality*: The composition method does not restrict the design of individual controllers. Each controller can be as primitive or as sophisticated as its designer wishes.

3.1 Controller Abstraction

A controller within the pool of available controllers can be as simple as a constant force, or as complex as a structured hierarchy of multiple levels of control abstraction. For example, as more controllers are added to the system, we may wish to group all the walking and running controllers together into a cluster that can be treated as one encapsulated controller.

Regardless of the encapsulation, our composition method requires controllers to define *pre-conditions*, *post-conditions* and *expected performance*. Pre-conditions are a set of conditions over the state of the character and the environment. If these conditions are met then the controller can operate and possibly enable the character to satisfy the post-conditions. Assuming that the pre-conditions were met, the post-conditions define a range of states for the final state of the character after the execution of the controller. In other words the controller realizes a mapping between a domain of input states to a range of output states for the character. Because of unexpected changes in the environment, this mapping may not always succeed, which motivates the notion of expected performance. The controller should be able to evaluate its performance in order to detect failure at any point during its operation. To do this, the controller must at all times have knowledge of the current and expected state of the character or the environment.

Defining the pre-conditions, post-conditions, and expected performance for complex characters, motions, and environments is not a straightforward task. However, we believe that the effort required to generate these specifications is a fair and necessary price to pay to achieve the benefits of composability. Controllers that adhere to these specifications can form a pool of available controllers managed by the supervising controller. Fig. 2 presents an overview of the supervising controller's function and its interaction with the individual controllers at every time step of the simulation.

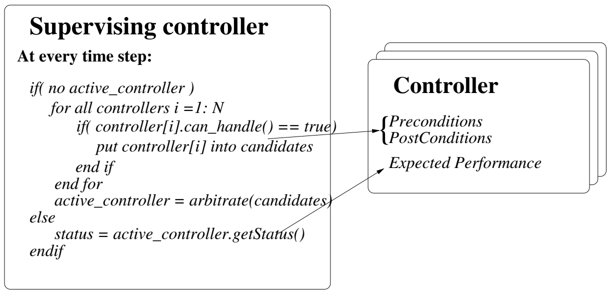


Figure 2: Controller selection and arbitration during simulation.

Before we elaborate on pre-conditions, post-conditions, and expected performance in subsequent sections, let us define the following quantities and symbols: The *state* $\mathbf{q} = [\mathbf{x}\ \dot{\mathbf{x}}]'$ of a figure is the vector of generalized positions $\mathbf{x}$ and velocities $\dot{\mathbf{x}}$, where the dot indicates a time derivative. The position and velocity of the center of mass are denoted as $\mathbf{c}$ and $\dot{\mathbf{c}}$ respectively. The *base of support* of a figure (often called the *support polygon*) is denoted as $\mathcal{S}$. It is represented by a polygon that surrounds the foot or feet that are in contact with the ground at any given time.

3.2 Pre-Conditions

In general, pre-conditions are relationships and constraints involving several different parameters. We have used the following parameters in our work:

- The initial state $\mathbf{q_i}$ of the figure. Most of our controllers can operate within a small region of the state space which we denote $\mathcal{R}(\mathbf{q}_i)$.
- Environmental parameters. These include the contact points between the character and the ground, as well as the normal of the ground and the amount of friction at the contact points. In the following we denote conditions (generally indicated by the letter C) on the environment parameters as C_e.
- The balance of the figure. Usually, this is indicated by the relative position and velocity between the figure's center of mass $\mathbf{c}$ and the base of support. Typically, if the projection of $\mathbf{c}$ along the gravity vector $\mathbf{g}$ does not intersect the base of support $\mathcal{S}$, the figure is considered to be unbalanced. We denote the balance conditions as $C_b(\mathcal{S}, \mathbf{g}, \mathbf{c}, \dot{\mathbf{c}})$.
- A target state $\mathbf{q}_t$, or in general a target region of the state space $\mathcal{R}(\mathbf{q}_t)$, which can be provided by the user.

Pre-conditions consist of unions of instances of the above conditions and are denoted

$$\mathcal{P} = C(\mathcal{R}(\mathbf{q}_i), \mathcal{R}(\mathbf{q}_t), C_b, C_e). \quad (1)$$

The determination of pre-conditions is crucial to the success of our composition framework and will be examined in detail in Section 4.

3.3 Post-Conditions

Successful operation of a controller brings the character from an initial state, as defined by the pre-conditions, to a desired state or a desired region $\mathcal{R}(\mathbf{q}_o)$ in the state space. This region along with balance C_b and possibly environmental constraints C_e form the post-conditions of a controller:

$$\mathcal{O} = C(\mathcal{R}(\mathbf{q}_o), C_b, C_e). \quad (2)$$

Note that the pre-conditions may reference a subset of the post-conditions that is sufficient to characterize what the controller can achieve. In general, however, the post-conditions are different from the pre-conditions. For example, while a pre-condition for a falling controller requires that the center of mass be moving, the post-conditions require that the center of mass be at rest.

3.4 Expected Performance

Our framework permits the automatic selection of the appropriate controller based on the information provided by the controllers themselves. Only the individual controllers can detect whether they are operating normally or whether failure is imminent. Failure in our case means that the controller cannot meet its post-conditions $\mathcal{O}$. The controller may fail because of a sudden change in the environment or because of badly designed pre-conditions. The sooner a controller can detect failure the sooner another more appropriate controller can take over. This is important for making a character behave naturally. For example, the character should not attempt to continue a walking gait if it has lost its balance and it is falling. In our implementation, the expected performance $\mathcal{E}$ consists of expressions similar to those of the pre-conditions $\mathcal{P}$. In particular if the controller successfully completes its task in the time interval $[t_1, t_2]$, then $\mathcal{E}(t_1) \in \mathcal{P}$ and $\mathcal{E}(t_2) \in \mathcal{O}$.

3.5 Transitions

Transitions between controllers are not explicitly modeled as they would be in a finite state machine. They occur implicitly in response to the evolution of the motion over time, as the system state traverses the "regions-of-competency" of the various controllers. Nevertheless, given that most controllers are designed for specific situations, typical patterns of controller activation occur. Fig. 3 shows the family of controllers designed for the 3D dynamic character and their typical transition patterns. For example, the controllers and transitions used in achieving the motion shown in Fig. 1 is given by balance → fall → default → rollover → prone-to-standing → balance. Fig. 4 similarly shows the family of controllers designed for the 2D dynamic character and their typical transition patterns. Note that not all possible transitions are shown in either of Figs. 3 and 4. For example, the prone-to-standing → fall transition can occur if the figure is given a sufficiently strong push while rising. Most of the transitions which are not shown but are still practically feasible are of this nature, dealing with falling behaviors. Note that the fall controller always responds to the specific direction of the current fall.

Any transition involves one controller being deactivated and another being activated. A controller can become deactivated (and thereby elicit a transition) for one of three reasons. First, it may relinquish control by declaring success upon reaching its post-condition, as is the case for a standup controller which has successfully returned the character to a standing position. Second, user intervention may elicit a transition. The controllers designed for sitting or balanced standing will retain control until intervention by a user (or by a higher level planner) forces a desired transition. Thus, when the 2D character is balanced a user-driven process must choose among the next plausible actions, namely one of *sit*, *walk*, or *dive* (see Fig. 4). Third, a controller may detect failure, as will be the case for unpredictable events such as a push or an unforeseen obstacle causing the character to trip. The transitions in Figs. 3 and 4 are labelled according to the type of controller deactivations which typically elicit the given transition patterns.

We note that our framework is designed to work in interactive settings. As such, controllers typically start with slightly different initial conditions each time they are invoked, the user can interact with the character at any time, and generally there are no guarantees that the controller will reach the same end state each time it operates. As a result, the transition graph is dynamic in structure.

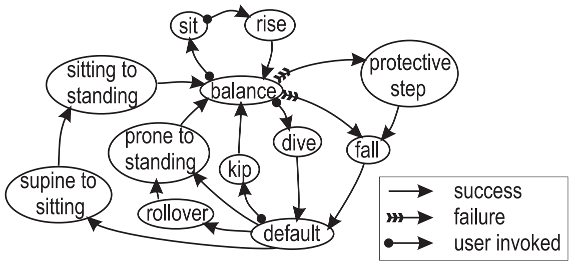


Figure 3: Controllers and typical transitions for 3D figure

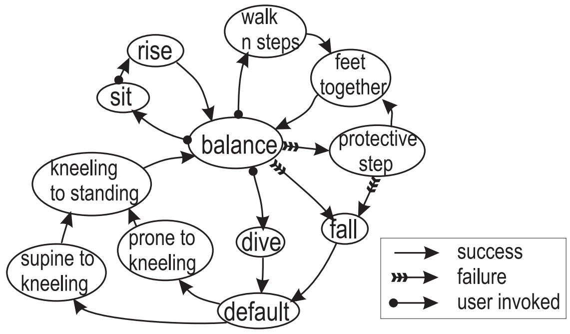


Figure 4: Controllers and typical transitions for 2D figure

4 Determining Pre-Conditions

For controllers associated with complex dynamic characters, determining the exact region of the state space and the general conditions that determine success or failure of the controller is in general a non-trivial matter. In this section, we address this problem via manual and automatic approaches. The manual approach allows designers to incorporate their knowledge within controllers, whereas the automatic approach is based on machine learning techniques.

4.1 Manual Approach

For certain cases, suitable pre-conditions for specific controllers may be found in the biomechanics literature [8, 25]. For example Pai and Patton [25] present a comprehensive study of balance in the sagittal plane and identify the conditions under which a human can compensate for postural disturbances and maintain balance without stepping. For certain other cases, the pre-conditions are trivially defined by the desired motion itself. Certain controllers function as intermediate stages between other controllers. If controller B is the intermediate step between A and C then the post-conditions of A dictate the pre-conditions of B and similarly the pre-conditions of C define the post-conditions of B. Finally, in some cases the pre-conditions are computed by manual experimentation. For example a simple balance controller based on an inverted pendulum model [12] has intrinsic stability that can tolerate small disturbances. After the controller has been designed, repeated testing under disturbances of increasing magnitude can yield an approximation of the pre-conditions and the post-conditions.

In any case, the designer of a controller presumably understands the way the controller operates, and thus is able to provide high level conditions on its success or failure. For example, the designer of a walking controller knows if the controller can operate when the walking surface has minimal friction properties. Also, human motion is shaped by notions such as comfort, and only the designer can take this into account. For example, if a person is pushed while standing he/she might take a protective step because it may be more comfortable to do so instead of maintaining an inverted pendulum balancing strategy. Similarly, the way people react to slipping and imbalance and the protective behaviors they employ are largely age dependent.

4.2 Automatic, Learning Approach

In this section, we introduce an automatic, machine learning approach to determining pre-conditions, which is based on systematically sampling the performance of controllers. Our method uses a machine learning algorithm attributed to Vapnik [33] known as *Support Vector Machines* (SVMs), which has recently attracted much attention, since in most cases the performance of SVMs matches or exceeds that of competing methods.

4.2.1 Support vector machines (SVMs)

SVMs are a method for fitting functions to sets of labeled training data. The functions can be general regression functions or they can be classification functions. In our application, we use simple classification functions with binary outputs which encode the success or failure of a controller.

Burges [5] provides an excellent tutorial on SVMs. Mathematically, we are given l observations, each consisting of an d-dimensional vector $\mathbf{x}_i \in \Re^d, i = 1, \ldots, l$ and the associated "truth" $y_i \in \{-1, 1\}$ provided by a trusted source. Here, $y_i = 1$ labels a positive example—in our application, the observed success of a controller applied when the dynamic figure is in state $\mathbf{x}_i$—while $y_i = -1$ labels a negative example—the failure of the controller applied to state $\mathbf{x}_i$. The set of observations $\{\mathbf{x}_i, y_i\}$ is called the *training set*. The SVM is a machine whose task is to learn the mapping $\mathbf{x}_i \mapsto y_i$ from a training set. The SVM is defined by functional mappings of the form $\mathbf{x} \mapsto f(\mathbf{x}, \alpha)$, where α are parameters. A particular choice of α generates a "trained" SVM. In a trained SVM, the sign of the *decision function* $f(\mathbf{x})$ represents the class assigned to a test data point $\mathbf{x}$. In our application, a properly trained SVM predicts if a controller will succeed ($f(\mathbf{x}) > 0$) or fail ($f(\mathbf{x}) < 0$) on a given state $\mathbf{x}$ of the dynamic character.

How does one train an SVM? In the simplest case of a linear SVM with separable training data, there exists a *decision boundary* separating positive from negative examples which takes the form of a "separating hyperplane" in $\Re^d$. The SVM training algorithm computes the separating hyperplane with the largest *margin* $d_+ + d_-$, where d_+ (d_-) is the shortest distance from the separating hyperplane to the closest positive (negative) example. SVM training requires the solution of a quadratic programming optimization problem involving a Lagrange multiplier α_i for every datapoint in the training set. Those datapoints in the solution with corresponding $\alpha_i > 0$ are called *support vectors*.

The support vectors are critical elements of the training set. They lie closest to the separating hyperplane. If other observations in the training set are moved (subject to certain restrictions) or removed and SVM training is repeated, the same separating hyperplane will result. To use a trained SVM, we simply determine on which side of the decision boundary a given test data point $\mathbf{x}$ lies and assign the corresponding class label to that point. The linear SVM is easily generalized to nonseparable training data.

Furthermore, it is straightforward to generalize the theory to encompass nonlinear SVMs for which the decision boundaries are no longer hyperplanes (i.e., the decision function are no longer linear functions of the data). The trick, in principle, is to map the data to some higher (possibly infinite) dimensional space in which the linear theory can be applied. This is easily done by introducing *kernel functions* $K(\mathbf{x_i}, \mathbf{x_j})$, such as the polynomial kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$, or the Gaussian or radial basis function (RBF) kernel $K(\mathbf{x}, \mathbf{y}) = \exp(-|\mathbf{x} - \mathbf{y}|^2 / 2\sigma^2)$. For the mathematical details, we refer the reader to [5].

4.2.2 Applying SVMs

To apply the SVM technique to the problem of determining controller pre-conditions, we train a nonlinear SVM classifier to predict the success or failure of a controller for an arbitrary starting state. Thus, the trained SVM demarcates the boundary of regions in the figure's state space wherein the controller can successfully do its job. Training sets comprising examples $\{\mathbf{x}_i, y_i\}$ are generated by repeatedly starting the dynamic figure at a stochastically-generated initial state $\mathbf{x}_i$, numerically simulating the dynamics of the figure under the influence of the controller in question, and setting $y_i = +1$ if the controller succeeds or $y_i = -1$ if it fails.

The distribution of the stochastically-generated initial states is of some importance. The sample points should ideally be located close to the boundaries which demarcate the acceptable pre-condition region of state-space. However, these boundaries are in fact the unknowns we wish to determine and thus we must resort to a more uniform sampling strategy. Unfortunately, the high dimensionality of the state-space precludes regular sampling. We thus adopt the following stochastic process to generate a suitable distribution of initial states: First, a nominal initial state is chosen, based upon the designer's knowledge of the controller. A short-duration simulation (typically 0.3s) is then carried out from this initial state while a randomized perturbation process is executed. This currently consists of applying an external force of random (but bounded) magnitude and random direction to the center-of-mass of the pelvis. Simultaneously, the character's joints are perturbed in a stochastic fashion by setting randomized offset target angles for the joints and using the character's PD joint controllers to drive the joints towards these perturbed positions. While the perturbation strategy is admittedly ad-hoc, we have found it to be effective in sampling the pre-condition space, as is validated by the online use of the learned pre-condition models.

We employ T. Joachims' SVM^{light} software which is available on the WWW [21]. The software can accommodate large training sets comprising tens of thousands of observations and it efficiently handles many thousands of support vectors. It includes standard kernel functions and permits the definition of new ones. It incorporates a fast training algorithm which proceeds by solving a sequence of optimization problems lower-bounding the solution using a form of local search. It includes two efficient estimation methods for error rate and precision/recall.

The SVM training phase can take hours in our application, but this is done off-line. For example, on a 733 MHz PIII computer, the SVM training time for a training set of 8,013 observations is 2,789 seconds using the polynomial kernel, 2,109 seconds using the linear kernel, and 211 seconds using the radial kernel. For a training set of 11,020 observations, the training time is 8,676 seconds using the polynomial kernel, 3,593 seconds using the linear kernel, and 486 seconds using the radial kernel. Once trained, the SVM classifier can provide answers on-line in milliseconds.

4.2.3 Pre-condition learning results

Through systematic experimentation, we have evaluated the performance of our automatic, SVM-based algorithm for learning con-

Controller	Training set size	Test set size	NN	SVM
StepToStand	8,999	9,110	80.97%	87.29%
LyingOnBellyToKneel	4,200	4,223	93.27%	94.46%
LyingOnBackToKneel	2,234	1,879	100.0%	100.0%
BendToStand	6,926	14,272	98.05%	99.77%
StandInPlace	17,317	20,393	83.63%	87.67%
Walk	11,020	8,658	92.78%	97.73%
StandToSit	1,100	1,286	64.15%	69.60%
StandToStep	16,999	17,870	72.12%	79.18%
KneelToStand	6,000	11,998	79.45%	85.06%

Table 1: Comparison between learned SVM and NN pre-conditions.

troller pre-conditions. We compared the performance of the SVM algorithm to that of a *nearest neighbor* (NN) classifier [9]. Given a training set, the nearest neighbor classifier returns for an arbitrary state $\mathbf{x}$ the same succeed/fail label as the label for that observation in the training set that is closest to $\mathbf{x}$. NN classifiers should perform particularly well in cases where the feasible area in the state space is highly fragmented and localized. Note that the NN method requires zero training time, but that it provides an answer in $O(n)$ time where n is size of the training set.

Table 1 summarizes the percentage success rates (rightmost columns) of learned pre-conditions for a variety of controllers that we use later in our demonstrations. To compute accuracy rates, we trained the SVM and NN pre-condition learning algorithms using randomly sampled observations collected from each of the controllers. Then we generated test sets of novel observations and compared their true success/fail status against that predicted by the trained NN and SVM pre-conditions to obtain the accuracy percentages listed in the rightmost two columns of the table. The results show that the SVM algorithm consistently outperforms the NN classifier. For the results shown in the table, the SVM algorithm employed polynomial kernel functions. We ran a similar set of experiments using Gaussian RBF kernel functions, but the accuracies were consistently lower than those obtained with polynomial kernel functions.

5 Implementation

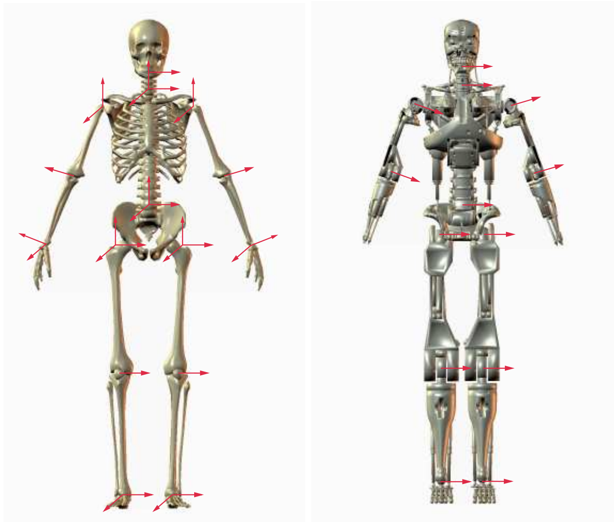
Our control composition framework is implemented within DANCE, a portable, extensible object-oriented modeling and animation system [24].[2] DANCE provides a platform that researchers can use to implement animation and control techniques with minimal design and implementation overhead. The core of the system supports four base classes, *Systems*, *Simulators*, *Actuators* and *Geometries* which are loadable as plug-ins in accordance with simple APIs.

Articulated objects are a *System* subclass that support skeleton hierarchies. They have kinematic properties and, usually, fully dynamic physical properties as well. Our virtual actors, which will be described shortly, are dynamic articulated objects implemented as *Systems* within DANCE.

An actuator is a generic concept that includes anything that can exert forces or, in general, interact in any way with systems or other actuators. For example, gravity, the ground, the collision mechanism, the supervisor controller and individual controllers are implemented as actuators. DANCE places no restrictions on the complexity of the controllers.

Simulators compute the equations of motion of all the dynamic characters and other systems in DANCE. DANCE offers built in support for SD/FAST, a commercial system which produces optimized simulation code [18]. However, any simulator that follows a simple

[2] DANCE is freely available for non-commercial use via the URL: www.dgp.toronto.edu/software/dance.htm



Joint	*Rotational DOFs 3D skeleton model*	*Rotational DOFs 2D terminator model*
Head	1	1
Neck	3	1
Shoulder	2	1
Elbow	2	1
Wrist	2	-
Waist	3	1
Hip	3	1
Knee	1	1
Ankle	2	1

Figure 5: Dynamic models and their degrees of freedom (DOFs).

API can be dynamically loaded into the system. Our simulators are automatically produced by SD/FAST from description files. They use Kane's method for computing articulated dynamics and a fourth order explicit Runge-Kutta time integrator for numerically simulating the motions.

Actuators and simulators are implemented as DANCE plug-ins. This allows the user to dynamically load controllers and simulators at runtime. In addition, researchers can exchange, simulators, and controllers in the form of dynamically linked pieces of code.

Object collisions (including self collisions) are handled by the *Collision* actuator. This actuator works on pairs of objects. The DANCE API allows it to work with objects that have different simulators. Collision detection is based on a library that uses oriented bounding boxes [13]. Collision resolution uses a penalty method that corrects geometry interpenetration using spring-and-damper forces. As with all penalty methods, it can make the system stiff, but it has performed well in our experiments to date.

5.1 Virtual Stuntman

5.1.1 Dynamic model

Fig. 5 depicts our 2D and 3D articulated character models. The red arrows indicate the joint positions and axes of rotational degrees of freedom (DOFs) which are also presented in the table. The 3D skeleton model has 37 DOFs, six of which correspond to the global translation and rotation parameters. The table in Fig. 5 lists the DOFs for the skeleton and a 2D "terminator" model. The dynamic properties of both models, such as mass and moments of inertia, are taken from the biomechanics literature and correspond to a fully-fleshed adult male.

The models are equipped with natural limits both on the motion of the joints and the strength of their muscles. However, DANCE has no built in muscle model and does not enforce the limits automatically. Users can implement the model they prefer and include code to enforce the limits of the model. Our plug-in control scheme uses rotational spring-and-damper forces for control and enforces the limits on the joints with exponential springs.

5.1.2 Pose and continuous control

Most of the controllers for our virtual stuntperson are based on pose control, which has often been used both for articulated objects [31] and soft objects [11]. Pose control is based on cyclic or acyclic finite state machines with time transitions between the states. Each state of the controller can be static or depend on feedback parameters. For some of our controllers, we use continuous control, in the sense that the control parameters are tightly coupled with some of the feedback sensors. The balance controllers are an example of this.

We designed several controllers based in part on experimental studies of how humans detect loss of balance [25] and analysis of protective and falling behaviors [8]. The resulting parameterized controllers have been enhanced with appropriate pre-conditions, post-conditions, and expected performance and have been integrated using an arbitration-based supervising controller.

5.1.3 Sensors

Each controller has full access to the internal data structures of DANCE including all the information associated with any character or object in the system. This allows the controllers to define arbitrary sensors that keep track of necessary information such as state parameters for feedback loops and the state of the environment. For efficiency, the supervisor controller calculates a number of common sensor values that are available to all the controllers.

5.1.4 Command interface

Many controller transitions in the control framework happen autonomously, such as taking a protective step in response to losing balance. However, other actions are initiated in a voluntary fashion. For example, a standing character can do any of (1) remain standing using the balance controller, (2) sit-down, (3) walk, and (4) dive. Currently, the user directs these voluntary motions by interactively entering command strings to the supervisor controller which, in turn, directly increases the suitability score of the designated controller and forces the arbitration process to be invoked to select a new active controller. The control of voluntary motions could equivalently be delegated to a high-level planner, although this kind of planning is beyond the scope of our work at present.

6 Results

At the heart of our prototype system is a composite controller that is capable of handling a large number of everyday tasks, such as walking, balancing, bending, falling, and sitting. In addition, we present brief descriptions of the controllers involved in producing several stunt actions. While the given controller descriptions are for the 3D character, the equivalent 2D controllers are very similar. Finally, we discuss motion sequences generated using these families of controllers[3].

[3] See www.dgp.toronto.edu/∼pfal/animations.html for the associated animations.

6.1 Everyday Actions

We began our implementation with the simple tasks of standing, recovering balance when pushed, and falling. An autonomous human agent should be able to balance, standing naturally in place. Should loss of balance occur, the character ought to react naturally either with a restoring motion or with a protective falling behavior depending on which action is appropriate in each case. Affording a dynamic articulated figure with natural reactions to loss of balance or impending falls is an essential step towards believable autonomous characters.

6.1.1 Balancing

A *balance* controller is responsible for maintaining a natural standing posture. This controller is based on an inverted pendulum model [12], using the ankles to control the body sway. Despite the fact that the body of the character is not as rigid as the inverted pendulum hypothesis suggests, the approximation works well in practice. As an example of the type of manually defined pre-conditions and post-conditions used for this controller and others, these details are given in Appendix A for the balance controller.

An animated character should attempt to maintain balance in response to external disturbances by shifting its weight, taking a step or bending at the waist. If the character cannot maintain balance, it must then resort to a falling behavior.

6.1.2 Falling

The manner in which people fall depends on a number of factors such as their physique, their age and their training. For example, the work in [19] shows that, during a fall, the elderly are more likely to impact their hip first as compared to younger adults falling under the same conditions. Our *fall* controller is designed with the average adult in mind. Its main action is thus to absorb the shock of the impact using mostly the hands.

The pre-conditions of the fall controller are defined in accordance with those of the balance controller. Situations that are beyond the capabilities of the latter should be handled by the fall controller. Our implementation of the fall controller can handle falls in any direction, responding in different ways to falls in different directions. Fig. 6 depicts frames from falls in a variety of directions. The second frame in Fig. 1 also demonstrates the action of the fall controller within a fall-and-recover sequence.

6.1.3 Sitting

Sitting down in a chair and rising from a chair are common everyday tasks. We have implemented a controller that can do both depending on the instructions of the animator. Apart from the command string supplied by the user, the pre-conditions are either a balanced upright posture or a balanced sitting posture. The post-conditions are similarly defined. The resulting action is illustrated in Fig. 7.

6.1.4 Rising from a supine position

Getting up off the ground is a surprisingly difficult motion to simulate. It involves rapid changes of the contact points and significant shifting of the figure's weight. In addition, the frictional properties of the ground model can influence the motion.

The pre-conditions for this controller are straightforward. The character must be lying with its back flat on the ground, within some tolerance. The post-conditions are that the character should be on its feet with its center of mass within the support polygon. Then it would be up to another controller to take over and bring the character from a crouching position to a standing one. A snapshot of a resulting motion is shown in Fig. 8.

Figure 6: Falling in different directions

6.1.5 Rolling over

When lying on their back, some people may choose to roll-over to a prone position before attempting to stand. We have implemented a *roll-over* controller that can emulate this action. The fourth frame in Fig. 1 demonstrates the action of the roll-over controller.

The pre-conditions of the roll-over controller require a supine posture, and no movement of the center of mass. The post-conditions of the roll controller are fairly simple and they include any prone position for which the character is extended and fairly straight; i.e., no crossing of legs or arms, etc.

6.1.6 Rising from a prone position

Frames 5–9 in Fig. 1 demonstrate the action of a controller that enables the virtual stuntperson to rise from the prone position. When lying face-down, the pre-conditions can be fairly relaxed. Our controller assumes that is has the time to change the state of the character to one from which it knows how to rise. As long as the figure is not lying on its arms and the ground is relatively flat it will attempt to get up. The post-conditions are chosen such that they satisfy the pre-conditions of the *balance* controller.

6.2 Stunts

Apart from everyday actions, we want our dynamic character to be able to do a variety of other voluntary actions dictated by the animator. Such actions can potentially include vigorous and/or physically dangerous actions. It is our hope that if a large number of researchers contribute controllers the character can eventually be used as a virtual stuntperson.

6.2.1 Kip move

The kip is an athletic motion often seen in martial arts films and is depicted in Fig. 9. The controller is based on a pose controller whose pre-conditions include a variation of supine positions. As before, the first part of the controller makes sure that the character assumes a position suitable for performing the kip. The larger part of the motion is ballistic, which focuses the control mainly at the kick off and landing phases. The last part of the controller applies continuous control to bring the stuntman to an erect position from which the balance controller can take over.

6.2.2 Plunge and roll

Fig. 10 shows the stuntman performing a suicidal dive down stairs. The character can be instructed to lunge forward and upward at a takeoff angle controlled by the user. When the hands contact the ground a front-roll is attempted. The pre-conditions of this controller are defined be an upright position and little movement of the center of mass.

We have also experimented with a multiple character scenario, with one character tackling another, Fig. 11. While the timing of the tackle is scripted, it illustrates the capability of the system to cope with a pair of interacting characters, each equipped with its own supervisory controller.

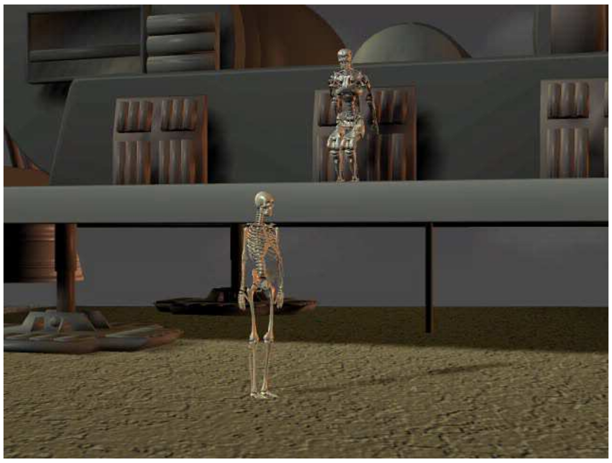


Figure 11: Two interacting virtual characters.

6.3 Animation Sequences

We have produced two relatively long animation sequences that demonstrate the potential of the our framework. The sequence for the 3D skeleton model presented in Fig. 1 involves controllers whose pre-conditions are provided analytically by the designer. Such conditions tend to define square regions within the space defined by the parameters involved. Despite their simple form, such pre-conditions can generally work well as is demonstrated by the intricacy of the animation produced. We expect to investigate the application of SVM-learned pre-conditions to the 3D model in the future.

A second animation sequence with the 2D terminator model (see Fig. 12) makes use of a set of controllers having a mix of analytic and learned pre-conditions. The sequence of controllers that generated the animation was: balance → sit → lean-forward → rise → balance → walk → step-to-stand → balance → dive → default → kneel → kneel to stand → balance → step-forward → step-to-stand → balance → step-back → step-to-stand → balance → fall → default. The analytical pre-conditions prune large parts of the state space and the svm-classifier provides a more accurate success/failure prediction within the remaining region. During the animation sequence, the svm-classifier correctly refined the analytical answer in several cases.

6.4 Performance Issues

Most of the computational burden in our approach lies in the numerical simulation of the equations of motion. The computations associated with the controllers and our composition framework are negligible in comparison. In general, the 2D model simulates in real time, while the 3D model runs between 5 and 9 times slower than real time on a 733 MHz Pentium III system.

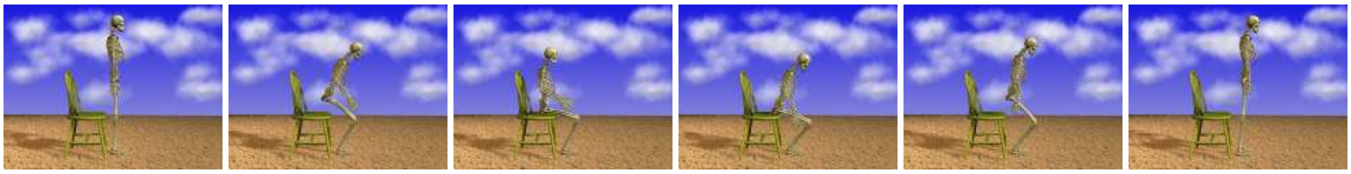


Figure 7: Sitting and rising from a chair

Figure 8: Rising from a supine position on the ground and balancing erect in gravity.

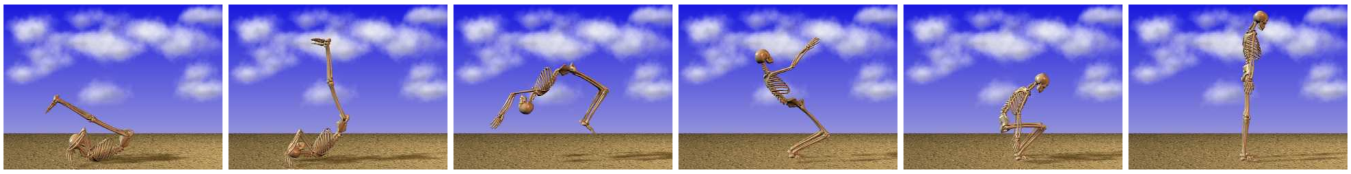


Figure 9: Kip move: A more vigorous way of getting up from the supine position as in the first frame of Fig. 8.

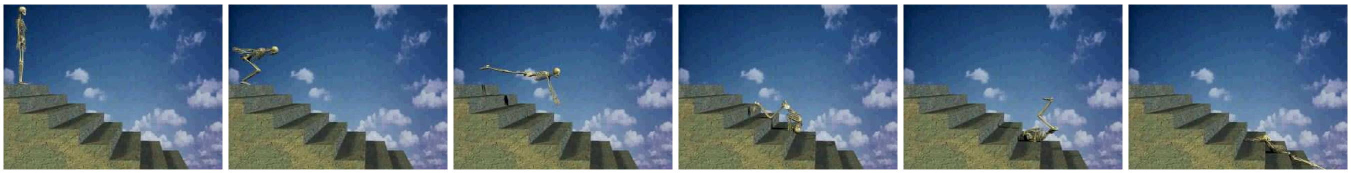


Figure 10: Ouch!

7 Conclusion

The challenges of physics-based controller design plus the technical obstacles that researchers face when attempting to share their algorithms has hindered progress in the important area of physics-based character animation. This paper has presented a methodology for ameliorating the problem with a framework which facilitates the exchange and composition of controllers. Our framework has been implemented within a freely available system for modeling and animating articulated characters. To our knowledge, our system is the first to demonstrate a dynamic anthoropomorphic character with controlled reactions to disturbances or falls in any direction, as well as the ability to pick itself up off the ground in several ways, among other controlled motions. We hope that our system will foster collective efforts among numerous practitioners that will eventually result in complex composite controllers capable of synthesizing a full spectrum of human-like motor behaviors.

Given the enormous challenge of building controllers capable of large repertoires of dynamic human-like motion, it is inevitable that the work presented in this paper is incomplete in many ways. Published control methods for 3D walking, running, and stair climbing make obvious candidates for integration into our system. Coping with variable terrain and dynamic environments are dimensions of added complexity that should provide work for years to come. Automatic parameterization of controllers to variations in character dimensions and mass is a necessary step for having solutions adaptable to a variety of characters. Deriving controllers from motion-capture data is an exciting but difficult prospect, although some progress is already being made in this area. Other methods of "teaching" skills to a dynamic character also warrant investigation. Finally, intelligently integrating controllers which affect only subsets of DOFs needs to be addressed in order to allow for the parallel execution of controllers.

Acknowledgements

We wish to thank Joe Laszlo for his help with the video editing equipment and for useful discussions. We would also like to thank Symbolic Dynamics Inc. for allowings us to distribute the equations of motion of the 3D human model. This work was supported by grants from NSERC and CITO.

A Balance controller

The articulated body must be in a balanced upright position, the velocity and acceleration of the center of mass should not exceed certain threshold values as explained in [25], and both feet must maintain contact with the ground at all times. The controller can tolerate small perturbations of the posture and the velocity/acceleration of the center of mass by stiffening the ankle joints. For larger accelerations of the center of mass, the controller actively actuates the ankle joint to reduce the acceleration of the center of mass. The post-conditions are similar to the pre-conditions. In mathematical form using the notation defined in Section 3:

Figure 12: A still image from the terminator sequence. The dynamic terminator model has been knocked backward by the force of a collision to the head by the red ball. The terminator maintains balance by taking a protective step.

$\mathcal{P}$:

Acceleration: $\ddot{\mathbf{c}} < 0.1$ m/sec^2.
Velocity: $\dot{\mathbf{c}} < 0.3$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Posture: $(1/n)\sum_i |\mathbf{q}[i] - \mathbf{q}_0| < 0.1$ rad, where $i =$ (thigh, knee, waist) and n is a normalization parameter.

$\mathcal{O}$:

Acceleration: $\ddot{\mathbf{c}} < 0.01$ m/sec^2.
Velocity: $\dot{\mathbf{c}} < 0.05$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Posture: $(1/n)\sum_i |\mathbf{q}[i] - \mathbf{q}_0| < 0.1$ rad, where $i =$ (thigh, knee, waist) and n is a normalization parameter.

References

[1] Ronald C. Arkin. *Behavioral Robotics*. MIT Press, 1998.

[2] W. W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. *Proceedings of Graphics Interface '85*, pages 407–415, 1985.

[3] N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press, 1993.

[4] N. I. Badler, B. Barsky, and D. Zeltzer. *Making Them Move*. Morgan Kaufmann Publishers Inc., 1991.

[5] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[6] R. R Burridge, A. A. Rizzi, and D. E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, June 1999.

[7] Tolga Capin, Igor Pandzic, Nadia Magnenat Thalmann, and Daniel Thalmann. *Avatars in Networked Virtual Environments*. John Wiley & Sons, 1999.

[8] M. C. Do, Y. Breniere, and P. Brenguier. A biomechanical study of balance recovery during the fall forward. *Journal of Biomechanics*, 15(12):933–939, 1982.

[9] R. O. Duda and P. E Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[10] Petros Faloutsos. *Composable Controller for Physics-based Character Animation*. PhD thesis, Univeristy of Toronto, DCS, Toronto,Canada, 2001. To be awarded.

[11] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, 1997.

[12] R. C Fitzpatrick, J. L. Taylor, and D. I. McCloskey. Ankle stiffness of standing humans in response to imperceptible perturbation: reflex and task-dependent components. *Journal of Physiology*, 454:533–547, 1992.

[13] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Computer Graphics (SIGGRAPH 96 Proceedings)*, pages 171–180, 1996.

[14] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-based locomotion through control abstraction. *Proceedings of ACM SIGGRAPH: Computer Graphics*, pages 63–70, August 1995.

[15] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. *Proceedings of ACM SIGGRAPH: Computer Graphics*, pages 9–20, July 1998.

[16] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. *Proceedings of SIGGRAPH 95, ACM Computer Graphics*, pages 71–78, 1995.

[17] Jessica K. Hodgins and Nancy S. Pollard. Adapting simulated behaviors for new characters. *Proceedings of SIGGRAPH 97*, pages 153–162, August 1997.

[18] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. Sd/fast. Symbolic Dynamics, Inc., 1991.

[19] E. T Hsiao and S. N Robinovitch. Common protective movements govern unexpected falls from standing height. *Journal of biomechanics*, 31:1–9, 1998.

[20] Boston Dynamics Inc. The digital biomechanics laboratory. `www.bdi.com`, 1998.

[21] T. Joachims. Making large-scale svm learning practical. advances in kernel methods. In B. Schölhopf, C. Burges, and A. Smola, editors, *Support Vector Learning*. MIT-Press, 1999. http://www-ai.cs.uni-dortmund.de/DOKUMENTE/joachims_99a.pdf.

[22] Joseph F. Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH 96*, pages 155–162, August 1996.

[23] Honda Motor Co. Ltd. www.honda.co.jp/english/technology/robot/.

[24] Victor Ng and Petros Faloutsos. Dance: Dynamic animation and control environment. Software system, www.dgp.toronto.edu/DGP/DGPSoftware.html.

[25] Yi-Chung Pai and James Patton. Center of mass velocity-position predictions for balance control. *Journal of biomechanics*, 30(4):347–354, 1997.

[26] Marcus G. Pandy and Frank C. Anderson. Three-dimensional computer simulation of jumping and walking using the same model. In *Proceedings of the VIIth International Symposium on Computer Simulation in Biomechanics*, August 1999.

[27] Marcus G. Pandy, Felix E. Zajac, Eunsup Sim, and William S. Levine. An optimal control model for maximum-height human jumping. *Journal of Biomechanics*, 23(12):1185–1198, 1990.

[28] M. H. Raibert. *Legged Robots that Balance*. MIT Press, 1986.

[29] Cecile Smeesters, Wilson C. Hayes, and Thomas A. McMahon. Determining fall direction and impact location for various disturbances and gait speeds using the articulated total body model. In *Proceedings of the VIIth International Symposium on Computer Simulation in Biomechanics*, August 1999.

[30] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. *Proceedings of SIGGRAPH 94*, pages 43–50, 1994.

[31] M. van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, pages 40–49, March 1996.

[32] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics (SIGGRAPH 90 Proceedings)*, 24(4):225–234, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[33] V. Vapnik. *Estimation of Dependecies Based on Empirical Data (in Russian)*. Nauka, Moscow, 1979. English translation Springer Verlag, New York, 1982.

[34] Jane Wilhelms and Brian A. Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In *Graphics Interface '85*, pages 97–104, May 1985.

[35] Wayne Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology, March 1998.