The Virtual Stuntman: Dynamic Characters with a Repertoire of Autonomous Motor Skills

Petros Faloutsos [a], Michiel van de Panne [b,e]
Demetri Terzopoulos [c,d]

[a] *University of California at Los Angeles, Department of Computer Science*
[b] *University of British Columbia, Department of Computer Science*
[c] *New York University, Courant Institute, Computer Science Department*
[d] *University of Toronto, Department of Computer Science*
[e] *Motion Playground, Inc.*

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. In this paper, we report on our ongoing development of a *virtual stuntman*, a dynamic graphical character that possesses a nontrivial repertoire of lifelike motor skills. The repertoire includes basic actions such as balance, protective stepping when balance is disturbed, protective arm reactions when falling, multiple ways of rising upright after a fall, and several more vigorously dynamic motor skills. Our virtual stuntman is the product of a recently proposed framework for integrating motor controllers, which includes among other ingredients an explicit model of pre-conditions; i.e., those regions of a dynamic figure's state space within which a given motor controller is applicable and expected to work properly.

Key words: Artificial Life, Virtual Humans, Computer Animation, Character Animation, Physics-Based Animation Control, Physics-Based Modeling

1 Introduction

Despite considerable progress in animating virtual humans [1,5], physics-based animated characters with a large repertoire of motor skills have so far been elusive. This may seem surprising in view of recent successes in implementing a slew of

specialist motor controllers capable of realistically synthesizing the complex dynamics of running, diving, and various gymnastic maneuvers [14]. The present paper develops further a new framework [10] that enables the systematic integration of multiple specialist motor controllers in order to create physically-simulated animated characters with broad repertoires of motor skills.

A domain that requires a broad variety of motor skills is stuntwork for action-intensive film genres, such as westerns or martial arts films. The dynamic nature of typical movie stunts makes them dangerous to perform, but it also makes them attractive candidates for the application of physics-based animation. With this in mind, we are pursuing the enticing long-term goal of creating an autonomous *virtual stuntman*. We have made significant progress towards this goal by focusing our attention on dynamic falling due to various disturbances, getting up after a fall, and several other complex combinations of motor control skills often associated with movie stuntmanship. The technical challenge that we face lies in developing appropriate motor control strategies for specific actions and in integrating these controllers into a coherent whole.

In prior work [10], we have demonstrated families of composable motor controllers for a dynamic articulated figure whose physical parameters are consistent with a fully-fleshed adult male. As an example of the results of our efforts, Fig. 1 illustrates this 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 absorb 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 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.

1.1 Related work

The simulation and animation of human characters is a challenging problem in many respects. Models for human motion must meet a particularly high standard, given our familiarity with what the results should look like. Comprehensive solutions must aspire to distill and integrate knowledge from biomechanics, robotics, control, and animation. The biomechanics literature is a useful source of predictive models for specific motions, and computer simulation is becoming an increasingly useful tool in this domain [21,22,25]. Robotics research has made remarkable progress in the successful design of a variety of legged robots [24] and, more recently, bipedal anthropomorphic robots [18]. Computer animation, typically unencumbered by the exacting fidelity requirements of biomechanical models and the

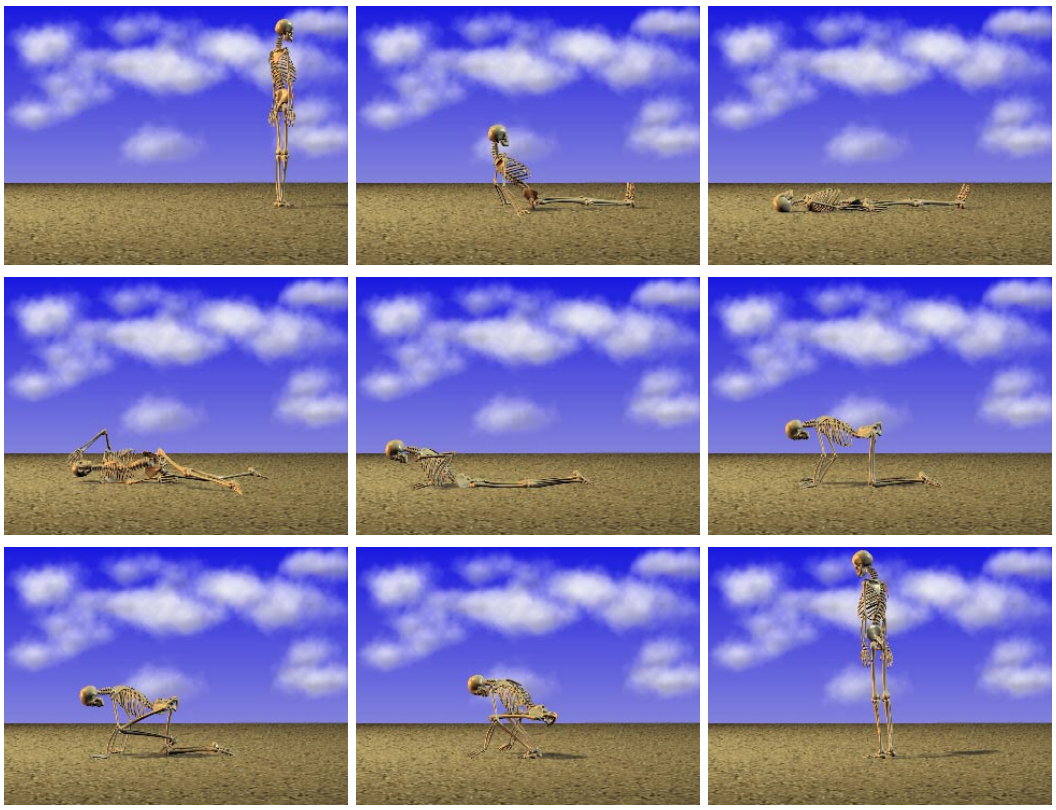


Fig. 1. A dynamic “virtual stuntman” falls to the ground, rolls over, and rises to an erect position, balancing in gravity (images in raster order).

mechanical limitations of robotic systems, has spawned a variety of kinematic and dynamic models for character motion [2]. Motor controllers for dynamically simulated characters have been successfully designed for specific human motions such as walking, running, vaulting, cycling, etc. [14,17,31].

Encouraging milestones on the road toward fully functional, dynamically simulated, articulated characters include an integrated set of motor controllers for biomechanically animated fish [26], a methodology for controller design and integration applicable to simple figures [28], and a demonstration of successful integration for selected diving and gymnastic motions such as leaping, tumbling, landing, and balancing [31]. Unlike previous work focusing on specific athletic movements, we progress toward dynamic human characters equipped with an integrated and wide-ranging repertoire of autonomous motor skills, by beginning with a core set of simple autonomous actions, including balancing, small steps, falling reactions, recovery from falls, sitting on and rising from chairs, and other actions. We then work towards more complex motor tasks, such as stunts.

1.2 Overview

The remainder of the paper is organized as follows: Section 2 reviews our motor controller composition framework. Section 3 presents our prototype dynamic virtual stuntman models. Section 4 describes the motor controllers that contribute to the prototype stuntman's repertoire of motor skills. Section 5 presents several animation results. Finally, Section 6 concludes the paper and proposes promising avenues for future work.

2 Controller Composition Framework

We have proposed a simple but effective framework for composing specialist controllers into more capable control systems for dynamic characters [10]. In our framework, *individual controllers* are black boxes encapsulating specialized control knowledge. First, an individual controller must be able to determine whether or not it can take the dynamic character from its current state to some desired goal state. Second, once an individual controller is active, it 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. Hence, we have a two-level, hierarchical controller composition scheme, as illustrated in Fig. 2.

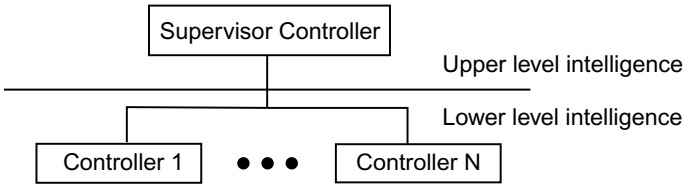


Fig. 2. Two level composition scheme.

Our composition method requires individual controllers to define *pre-conditions*, *post-conditions*, and *expected performance*. Pre-conditions, denoted $\mathcal{P}$, are a set of conditions over the state of the character and the environment. We have demonstrated the successful composition of controllers based on manually determined pre-conditions and on those learned automatically by a Support Vector Machine (SVM) technique which learns appropriate pre-conditions through the repeated sampling of the behavior of individual controllers in operation. If the pre-conditions are met, then the controller can operate and possibly enable the character to satisfy the post-conditions, denoted $\mathcal{O}$, the range of states that the character may be in after the execution of the controller. Thus, the controller realizes a transition between a domain of input states to a range of output states for the character. Because of unexpected changes in the environment, however, this transition may not always

succeed, which motivates the notion of expected performance, denoted $\mathcal{E}$; 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 continually be aware of the current and expected state of the character or the environment.

At each time step of the physics-based simulation, the supervisor controller first checks whether it needs to initiate a bid process, and proceeds to do so if the user-specified target state has changed or if there is no active controller (other than a default controller). During the bidding process, all available individual controllers determine whether their pre-conditions are satisfied and, if so, they bid for control over the dynamic character. The supervisor controller selects from among the collection of bidding controllers the one that returns the highest priority, registers it as the *active controller*, and invokes a method associated with the controller which implements its control strategy. The method returns to the supervisor controller a status parameter. If the status parameter indicates that the controller has failed, then a new bidding process is initiated. [1] Along with the status parameter, the method returns target values for some or all of the dynamic character's degrees of freedom along with associated stiffness and damping parameters, which are used by a set of proportional-derivative controllers to calculate the actual control torques. Alternatively, the active controller can choose to apply torques directly to the character and return no values for the supervisor's proportional-derivative controllers. In the case where no available controller bids for control, the supervisor controller activates the *default* controller, a generic controller, which we will describe in more detail later, that tries to do something sensible with the character when no specialist controller is able to assume control.

Some controllers automatically bid for control over the character when their pre-conditions are met; hence, many controller transitions occur autonomously, such as taking a protective step in response to a loss of balance. However, other actions are initiated voluntarily, and the associated controllers become active only at the request of the user. For example, a character balancing upright can be instructed to remain standing, to sit-down, to walk, or to take a dive. Currently, the user directs voluntary actions by interactively entering command strings to the supervisor controller. These commands increase the suitability score of the designated controller and forces invocation of the arbitration process which selects and activates the designated controller. The control of voluntary motions could be delegated to a high-level planner, but motion planning is beyond the scope of our current work.

Our control composition framework is implemented within DANCE, a portable, extensible object-oriented modeling and animation system [19]. [2] DANCE provides a

[1] An additional check avoids an infinite loop when a badly designed controller bids for control and immediately fails.

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

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 application program interfaces (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.

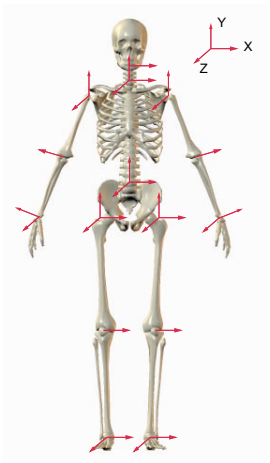
An actuator is a generic concept that includes anything that can exert forces or, in general, interact in any way with systems or other actuators. For example, gravity, the ground, the collision mechanism, the supervisor controller and individual controllers are implemented as actuators. DANCE places 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 for articulated bodies [15]. However, any simulator that follows a simple API can be dynamically loaded into the system. Our simulators are automatically produced by SD/FAST from model specification files. They use Kane's method for computing articulated dynamics and an explicit, fourth order Runge-Kutta time integration method.

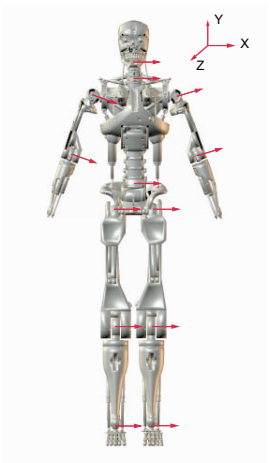
Actuators and simulators are implemented as DANCE plug-ins. This allows the user to dynamically load controllers and simulators at runtime. In addition, researchers can exchange, simulators, and controllers in the form of dynamically linked pieces of code.

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.

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.



(a)



(b)

Joint	*Skeleton model DOFs*	*Robot model DOFs*
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

(c)

Joint	Axis	Lower Limit	Upper Limit
Head	x	-45	45
Neck	x	-50	90
	z	-60	60
	y	-80	80
Shoulder	z	-90	90
	y	-80	160
Elbow	y	0	120
	x	-90	40
Wrist	z	-90	90
	y	-45	45
Waist	x	-45	90
	z	-55	55
	y	-50	50
Hip	x	-165	45
	y	-120	20
	z	-20	20
Knee	x	0	165
Ankle	x	-45	50
	z	-2	35

(d)

Fig. 3. Anthropomorphic models. (a) 3D-motion skeleton model and (b) 2D-motion "Terminator" robot model, (c) their rotational degrees of freedom (DOFs), and (d) lower/upper joint limits for the skeleton model.

3 Prototype Virtual Stuntman Models

Fig. 3 illustrates two dynamic articulated characters that will serve as prototype virtual stuntman models. The red arrows indicate the positions of the joints and their rotational degrees of freedom (DOFs), which are also enumerated in the table. The skeleton model, which is capable of full 3D motion, has 37 DOFs, six of which correspond to the global translation and rotation parameters. The 16 DOF "Terminator" robot model is limited to producing 2D (planar) motion. The leftmost table in the figure lists the DOFs of the models. The physical properties, such as mass and moments of inertia, of both models are consistent with anthropometric data for a fully-fleshed adult male, as found in the biomechanics literature (see Winter [30]). In particular, the overall mass of each model is 89.57 kilograms. The movement of the rotational degrees of freedom of the models is restricted by the physical limits of the human body. After researching the literature, we have decided to use the joint limits indicated (for the skeleton model) in the rightmost table in the figure.

The supervisor controller of the character is responsible for maintaining the articulated figure's joint limits. We use a method based on *exponential springs* to ensure

that rotations of the character's body parts do not exceed the user specified limits. If any rotational degree of freedom q_i, exceeds its allowable range of (q_i^l, q_i^u), where the superscripts designate "lower" and "upper" limits, respectively, the exponential springs produce the forces:

$$\texttt{if} \quad (q_i^l - q_i) > \epsilon \quad \texttt{then} \quad f_i^l = k_s^l(e^{k_s^e(q_i^l - q_i)} - 1) - k_d\dot{q}_i,$$
$$\texttt{if} \quad (q_i - q_i^u) > \epsilon \quad \texttt{then} \quad f_i^u = k_s^l(e^{k_s^e(q_i - q_i^u)} - 1) - k_d\dot{q}_i,$$

depending on the limit that has been violated. Exponential springs are widely used in a variety of control problems. We have determined that the spring constants $k_s^l = 10.0$, $k_s^e = 1.0$, and $k_d = 10.0$ produce satisfactory behavior.

Feedback is crucial to the motor control of complex dynamic characters, such as virtual humans. Motor controllers need information about the state of the character, where it is facing, whether it is balanced, etc. Controllers also need to have information about the environment, such as body/ground contact points, the slope of the terrain at contact points, the position of obstacles, etc. Most of the information on the character can be computed from the state parameters; however, it is often more convenient to use higher-level sensors that are more intuitive, can be computed once per time step, and can be shared among controllers. In our current implementation, 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 the following common sensor values which are made available to all specialist controllers:

- *Support polygon*. The support polygon $\mathcal{S}$ is defined by the convex hull of the feet that are in contact with the ground, and it is crucial for the balance of the character.
- *Center of mass information*. The position $\mathbf{c}$, velocity $\dot{\mathbf{c}}$, acceleration $\ddot{\mathbf{c}}$, and relative position of the center of mass with respect to the support polygon.
- *Hip center of mass information*. The position $\mathbf{c}^h$, velocity $\dot{\mathbf{c}}^h$, acceleration $\ddot{\mathbf{c}}^h$, and relative position of the hip's center of mass with respect to the support polygon.
- *Contact information*. An indication of whether the feet, head, hip and thighs are in contact with the ground.
- *Orientation*. The *facing* vector $\mathbf{v}^f$ and *up* vector $\mathbf{v}^u$ of the hip, indicating the direction that the hip faces and how far it leans, respectively.

Fig. 4 shows the support polygon, the facing vector and the up vector relative to the skeleton model.

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 reduced-

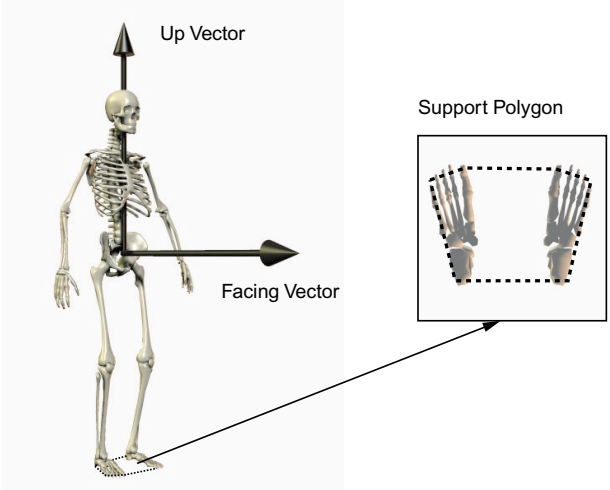


Fig. 4. A few sensors associated with the 3D-motion model.

DOF, 2D-motion robot model simulates in real time on a 733 MHz Pentium III computer system, whereas the 3D-motion skeleton model runs between 5 and 9 times slower than real time.

4 Implementing Motor Skills

In this section, we present the individual, specialist controllers that we have implemented for the prototype virtual stuntman and we describe in detail their analytical, composable APIs. Most of the controllers for our models are based on pose control, which has often been used both for articulated objects [27] and soft objects [9]. 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 can 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 [20] and analysis of protective and falling behaviors [7]. The basic controllers are augmented with the aforementioned representation of pre-conditions, post-conditions, and expected performance.

Fig. 5 lists the number of poses (states) of each controller. Controllers that do not follow the *pose controller* (finite state machine) paradigm are marked with “C”. The table shows also whether the controllers implement sensor-based (“S”) or time-based (“T”) transitions between poses.

Let us first define the following quantities and symbols: The *state* $\mathbf{q} = [\mathbf{x}\ \dot{\mathbf{x}}]'$ of an articulated figure is the vector of generalized joint angles $\mathbf{x}$ and angular velocities $\dot{\mathbf{x}}$, where the dot indicates a time derivative. The position and velocity of the center of

Controller	# poses (3D)	Transitions	# poses (2D)	Transitions
Balance	C	N/A	C	N/A
Fall	3	T & S	3	T & S
ProtectiveStep	4	T & S	5	T & S
Step	9	T & S	-	-
Plunge	3	T & S	6	T & S
SupineToCrouch	10	T & S	-	-
SupineToKneel	-	-	10	T & S
ProneToCrouch	12	T & S	-	-
ProneToKneel	-	-	6	T & S
RollOver	11	T & S	-	-
CrouchToStand	C	N/A	C	N/A
Kip	9	T & S	N/A	N/A
StandToSit	3	T	3	T & S
SitToCrouch	1	T & S	1	T & S
StandToAllFour	C	N/A	-	-
DStanceToCrouch	-	-	6	T & S
Slow Steps (Walk)	-	-	6	T & S

Fig. 5. Number of poses and pose transition types.

mass are denoted as $\mathbf{c}$ and $\dot{\mathbf{c}}$ respectively. As mentioned earlier, the support polygon of a figure is denoted as $\mathcal{S}$.

4.1 Default controller

The default controller is activated when no other controller requests control of the character. Its goal is to perform a sensible action in any given situation. In the absence of a better understanding of the situation, the most sensible thing to do is to keep the character in a comfortable position. We currently distinguish between two different situations, standing in place and lying on the ground. In the first case, the controller attempts to maintain the character's upright stance using moderate force while keeping the arms loose. If the character is leaning by more than a given threshold slant, then it is considered to be in a lying position, in which case the controller makes the character assume a relaxed pose. Thus far, these two strategies have worked well, in the sense that they bring the character smoothly into a perceived comfortable position. The default controller faces the difficult task of encompassing all situations for which we have not yet designed appropriate controllers. It therefore represents only a starting point for future improvements.

4.2 Everyday actions

A skillful, anthropomorphic character should be able to perform autonomously all the motor tasks that humans are able to do. However, even very common tasks such as walking require the sophisticated control of body dynamics. As stated earlier, we focus on a subset of everyday motions, starting with the most simple one—standing

in place. In the event of a loss of balance, the character should react naturally, either with a restorative leg motion or with a protective falling behavior, as is appropriate in the specific circumstance. Affording a dynamic articulated figure natural reactions to a loss of balance or an impending fall, plus the ability rise up subsequent to a fall, is an essential step towards believable, autonomous characters.

4.2.1 Balancing

Balancing in a quiescent, upright stance is a complex biomechanical control phenomenon that depends on different factors, such as the distance between the feet, and the presence of (or lack of) visual feedback, Day [6]. A considerable body of research aims to understand the sensory information (van der Kooij [29]), and reflex responses that humans use to maintain quiet stance (Fitzpatrick [11]). The strategies that people employ as a response to disturbances during quiet stance are generally divided into *hip strategies* and *ankle strategies* depending on whether the hips or the ankles are the dominant regulators of the postural stability. Gatev [12] provides a comprehensive analysis of balance strategies during quiet stance focusing on ankle control. Most researchers in biomechanics seem to agree that ankle strategies are more likely to occur in response to small disturbances, while hip strategies occur in response to larger disturbances.

Our *balance* controller is responsible for maintaining a natural standing posture. It is based on an inverted pendulum model that uses the ankles to regulate the body sway [11]. 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. Our balance controller uses an ankle angle of 0.06 radians as the equilibrium position.

For this 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 by Pai [20], 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:

$\mathcal{P}$:
Velocity: $|\dot{\mathbf{c}}| < 0.3$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Posture: (upright) $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 0.1$ rad,
where $i = (\text{thigh}, \text{knee}, \text{waist})$, $\mathbf{q}_0 = \mathbf{0}$,
and n is a normalization parameter.
Contact: feet on ground.
$\mathcal{O}$:

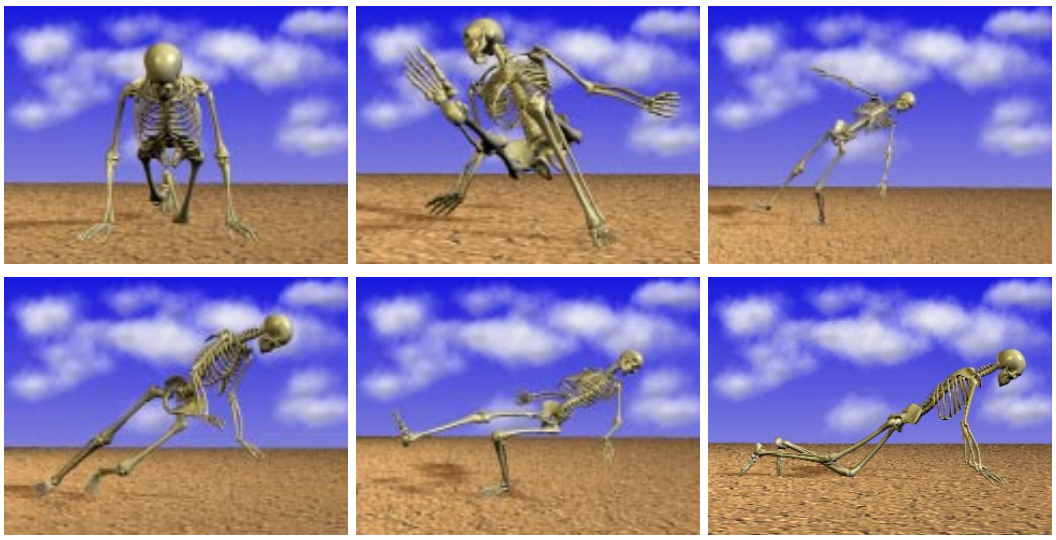


Fig. 6. Falling in different directions

Velocity: $|\dot{\mathbf{c}}| < 0.05$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Posture: (upright) $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 0.1$ rad,
where $i = (\text{thigh}, \text{knee}, \text{waist})$, $\mathbf{q}_0 = \mathbf{0}$,
and n is a normalization parameter.
Contact: feet on ground.

The expected performance $\mathcal{E}$ is identical to the the pre-conditions. We enhance the behavior of our balance controller in a simple fashion by kinematically simulating the character's visual attention. In particular, we apply Perlin noise, Perlin [23], to the degrees of freedom of the neck that makes the character look around in its environment.

Because of the relatively simple task that this controller has to accomplish and the inherent stability of the simple ankle strategy that we employ, the balance controller can be used successfully on slightly different terrains and characters. Nevertheless, the controller could be enhanced to employ more complex strategies, especially as responses to larger external disturbances. For example, an animated character should attempt to maintain balance by shifting its weight, or bending at the waist. If the character cannot maintain balance, it must then resort to taking a step or even initiating a fall behavior.

4.2.2 Falling

The manner in which people fall depends upon a number of factors, such as their physique, their age, and their training. Involuntary falling reactions are very common in everyday life, especially among young children and the elderly. They are probably the most common reason behind fracture injuries among the elderly. Hsiao

and Robinovitch [16] show 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 to absorb the shock of the impact using mostly the hands.

Wu [32] provides a way to distinguish falls from normal activities based solely on velocity characteristics. The pre-conditions of our fall controller define a larger acceptable region in velocity space than the one specified by Wu because they are defined in accordance with those of the balance controller. All situations that are beyond the capabilities of the latter should be handled by the fall controller:

$\mathcal{P}$:
  Vertical Velocity: $\dot{c}_v < 0.3$ m/sec.
  Balance: projection$(\mathbf{c}) \notin \mathcal{S}$.
  Contact: hip not on ground, hands not on ground.
$\mathcal{E}$:
  If falling forward, face down $v_y^f < 0.1$.
  If falling backward, face up $v_y^f > -0.1$.
  Contact with the ground in 3 seconds.
$\mathcal{O}$:
  Either
    Velocity: $|\dot{\mathbf{c}}| < 0.3$ m/sec.
  or
    head on ground.

The pre-conditions ensure that if the character is not balanced, then the fall controller bids to take over. The fall controller succeeds when the velocity and acceleration of the character are brought close to zero or when the head touches the ground. The expected performance ensures that the character keeps on falling in the same direction. In addition, it requires (a) that the character's facing direction does not reverse, something which might happen when falling from a great height, and (b) that the character touches the ground within 3 seconds in order to ensure that the fall was from a short height.

Our implementation of the fall controller computes the direction of the fall and responds accordingly. It can therefore handle a variety of pushes. Fig. 6 shows snapshots of falls in different directions. The second frame in Fig. 1 also demonstrates the action of the fall controller within a fall-and-recover sequence. The controller is relatively robust and it can be used on different characters and ground models.

4.2.3 Stand-to-sit and sit-to-crouch

Sitting down on a chair and rising up from a chair are common actions. We have implemented one controller that makes the character sit, starting from an upright stance, and another controller that prepares the character for the reverse action by

(a) Stand to sit controller.

(b) Sit to crouch controller.

Fig. 7. Sitting and rising from a chair.

making it lean forward until he is in a crouch position. [3] The resulting actions are illustrated in Fig. 7. The pre-conditions, post-conditions and expected performance of each controller are relatively simple:

Stand to sit controller:
$\mathcal{P}$:
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
Posture: (upright) $(1/n)\sum_i \sqrt{(q_i - q_{0,i})^2} < 0.1$ rad,
where $i = (\text{thigh}, \text{knee}, \text{waist})$, $\mathbf{q}_0 = \mathbf{0}$,
and n is a normalization parameter.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Contact: hip not on ground, hands not on ground.
$\mathcal{E}$:
Up vector: $v_y^u > 0.7$.
Requires that the character does not lean sideways, $|v_z^u| < 0.05$.
$\mathcal{O}$:
Up vector$v_y^u > 0.7$.
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
Requires that the character does not lean sideways, $|v_z^u| < 0.05$.

Sit to crouch controller:

[3] We use the term *crouch* to refer to any balanced posture of the character for which the legs of the character are symmetrically positioned.

$\mathcal{P}$:
Up vector: $v_y^u > 0.7$.
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
Posture: sitting: $(1/n) \sum_i |\mathbf{q}[i] - \mathbf{q}_0| < 0.5$ rad,
where $i = (\text{thigh}, \text{knee}, \text{waist})$,
$\mathbf{q}_0 = [-1.5 \;\; 1.5 \;\; 0.0]$,
and n is a normalization parameter.
Balance: projection$(\mathbf{c}) \notin \mathcal{S}$.
Contact: hip not on ground, hands not on ground.
$\mathcal{E}$:
Up vector: $v_y^u > 0.7$.
The character does not lean sideways, $|v_z^u| < 0.05$.
Time of completion less than 4 seconds.
$\mathcal{O}$:
Up vector: $v_y^u > 0.7$.
The character does not lean sideways, $|v_z^u| < 0.05$.
Balance: projection$(\mathbf{c}) \in \mathcal{S}_s$.

Here, $\mathcal{S}_s$ is subset of $\mathcal{S}$, shorter along the front-to-back axis, so as to ensure a more balanced final posture. If the controller does not succeed in four seconds, then something is assumed to be wrong and the controller aborts. We have tested these controllers with chairs of height 40 cm.

4.2.4 Rising from a supine position

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

For the three dimensional model the pre-conditions require that the character be lying with his back flat on the ground, within some tolerance. The post-conditions require that the character be balanced on its feet, with the feet side by side, but not necessarily straightened up. The expected performance makes sure that the character does not fall sideways and that it completes its task within 20 seconds.

$\mathcal{P}$:
Facing vector: $v_y^f > 0.97$
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
Contact: hip on ground.
The character does not lean sideways, $|v_z^u| < 0.05$.
$\mathcal{E}$:
The character does not lean sideways, $|v_z^u| < 0.05$.
Facing vector: $v_y^f > -0.2$.
Up vector: $v_y^u < 0.99$ if the hip is not on the ground.
$\mathcal{O}$:
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.

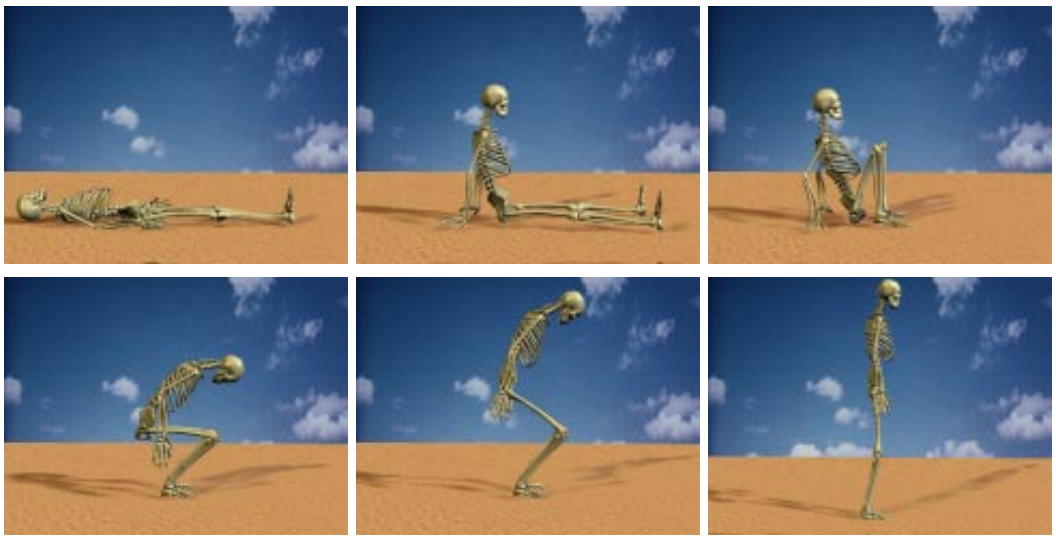


Fig. 8. Rising from a supine position on the ground and balancing erect in gravity.

Contact: hip not on ground.

A snapshot of a resulting motion is shown in Fig. 8. This controller is very sensitive to the character and the ground's friction model, which has a cofficient of friction of 0.6.

For the two dimensional model, the pre-conditions are the same as for the three dimensional case. However, the expected performance and the post-conditions require that the character ends up in a kneeling position within an appropriate time period:

$\mathcal{P}$:
Facing vector: $v_y^f > 0.97$.
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
Contact: hip on ground.
$\mathcal{E}$:
First phase:
Facing vector: $v_y^f > -0.2$.
Up vector: $v_y^u < 0.99$.
Second phase (kneel):
Up vector: $v_y^u > 0.7$.
Hip not on ground.
Posture: kneeling: $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 0.8$ rad,
where $i = (\text{thigh}, \text{knee})$, $\mathbf{q}_0 = [-1.43 \;\; 2.92\}$,
and n is a normalization parameter.
$\mathcal{O}$:
Up vector: $v_y^u > 0.7$.
Contact: hip not on ground.
Posture: kneeling: $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 0.8$ rad,
where $i = (\text{thigh}, \text{knee})$, $\mathbf{q}_0 = [-1.43 \;\; 2.92\}$,

and n is a normalization parameter.
Velocity: $\dot{\mathbf{c}} < 0.1$ m/sec.

The resulting motion is depicted in frames 12–16 in Fig. 15. The controller works for grounds with coefficient of friction equal or greater than 0.55.

4.2.5 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 that the character be in a supine position and that the center of mass not have noticable motion. The post-conditions of the roll controller are fairly simple; they include any prone position for which the character is extended and fairly straight; i.e., no crossing of legs or arms, etc. The expected performance is simple and makes sure that the character is facing down during the second part of the motion, where it is expected to do so. The controller has a limited time period to complete its task.

$\mathcal{P}$:
Facing vector: $v_y^f > 0.5$.
Up vector: $v_y^u < 0.3$.
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
Contact: hip on ground.
$\mathcal{E}$:
Up vector: $v_y^u < 0.5$.
Velocity: $|\dot{\mathbf{c}}| < 5.0$ m/sec.
First phase:
Facing vector: $v_y^f > 0.0$.
Second phase:
Facing vector: $v_y^f < 0.0$.
$\mathcal{O}$:
Facing vector: $v_y^f < -0.5$.
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.

The controller has been tested successfully for a few different starting states. One of its limitations is the lack of synergy between different body parts. For example, when the right arm moves away from the characters ribs, the hip is not moved in a fashion that can facilitate the arm's motion. Therefore the arm actuators have to use more energy to effect the desired motion.

4.2.6 Rising from a prone position

Frames 5–9 in Fig. 1 demonstrate the action of a controller that enables our three dimensional model to rise from the prone position. The pre-conditions require that

the character be lying face down but not on its arms. The post-conditions require that the character end up in a crouching position.

$\mathcal{P}$:
Facing vector: $v_y^f < -0.3$
Up vector: $v_y^u < 0.5.$
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
$\mathcal{E}$:
Facing vector: $v_y^f < 0.5.$
Up vector: $v_y^u < 0.99.$
Velocity: $|\dot{\mathbf{c}}| < 5.0$ m/sec.
Time of completion less than 15 secs.
$\mathcal{O}$:
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Up vector: $v_y^u > 0.7.$

The pre-conditions are similar for the two dimensional character. However, the expected performance and the post-conditions are different, since the goal in this case is for the character to reach a kneeling state.

$\mathcal{P}$:
Facing vector: $v_y^f < 0.$
Up vector: $v_y^u < 0.5.$
Velocity: $|\dot{\mathbf{c}}| < 0.05$ m/sec.
Contact: hip on ground.
$\mathcal{E}$:
First phase:
Facing vector: $v_y^f < 0.0.$
Contact: hands on ground or hip on ground.
Second phase:
Up vector: $v_y^u > 0.7.$
Contact: knee on ground and hip not on ground.
Third phase:
Contact: Knees on ground and hip not on ground.
Posture: kneeling: $(1/n)\sum_i \sqrt{(q_i - q_{0,i})^2} < 0.8$ rad,
where $i = (\text{thigh}, \text{knee})$, $\mathbf{q}_0 = [-1.43 \;\; 2.92\}$,
and n is a normalization parameter.
Facing vector: $v_y^f > 0.0.$
Up vector: $v_y^u > 0.0$
$\mathcal{O}$:
Facing vector: $v_y^f > 0.0.$
Up vector: $v_y^u > 0.0$
Contact: Knees on ground and hip not on ground.
Posture: kneeling: $(1/n)\sum_i \sqrt{(q_i - q_{0,i})^2} < 0.8$ rad,
where $i = (\text{thigh}, \text{knee})$, $\mathbf{q}_0 = [-1.43 \;\; 2.92\}$,
and n is a normalization parameter.
Velocity: $|\dot{\mathbf{c}}| < 0.05$ m/sec.

Fig. 9. Prone to kneel controller.

The motion produced by this 2D controller is shown in Fig. 9. The controller is not very sensitive to friction and has been tested with a friction coefficient in the range of $[0.4, 0.6]$. Both controllers must reach their post-conditions within a limited time interval, otherwise they fail.

4.2.7 Kneel-to-crouch

Currently, this controller has been implemented only for the 2D model. It is a pose controller that takes the character from kneeling position, such as the one produced by the previous controller, to a crouching position. The pre-conditions, post-conditions and expected performance are as follows:

$\mathcal{P}$:
- Facing vector: $v_y^f > -0.5$
- Velocity: $|\dot{\mathbf{c}}| < 0.5$ m/sec.
- Contact: feet and knees on ground.
- Posture: kneeling: $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 0.5$ rad, where $i = (\text{thigh}, \text{knee})$, $\mathbf{q}_0 = [-1.43 \;\; 2.92]$, and n is a normalization parameter.

$\mathcal{E}$:
- Velocity: $|\dot{\mathbf{c}}| < 5.0$ m/sec.
- First phase:
  - Facing vector: $v_y^f > -0.5$.
  - Contact: foot on ground, hip not on ground.
- Second phase:
  - Up vector: $v_y^u > 0.5$.
  - Contact: feet on ground and hip not on ground.

$\mathcal{O}$:
- Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
- Balance: projection$(\mathbf{c}) \in \mathcal{S}$.

A limitation of this controller is that any rising motion before the center of mass is placed above the front foot, relies too much on the help of the back leg. A more natural approach would be to first shift the center of mass above the front foot and then have the character rise. This limitation makes the control very sensitive to the ground model.

Fig. 10. Taking a step.

4.2.8 Step

This is a simple controller designed for the three dimensional model that makes it perform a single step. The final stage of the controller brings both feet together to achieve a standing position. The motion is depicted in Fig. 10. The pre-conditions require the character to be in an upright stance, while the expected performance makes sure that the character remains upright during the operation of the controller. The post-conditions require that the character is upright as the controller completes its operation.

$\mathcal{P}$:
Up vector: $v_y^u > 0.9$.
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
Contact: hip not on ground.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
$\mathcal{E}$:
Up vector: $v_y^u > 0.9$.
Velocity: $|\dot{\mathbf{c}}_{lat}| < 1.0$ m/sec,
$|\dot{\mathbf{c}}_{sag}| < 0.08$ m/sec.
$\mathcal{O}$:
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.

Here, $\dot{\mathbf{c}}_{lat}$ and $\dot{\mathbf{c}}_{sag}$ are the lateral and sagittal velocity of the center of mass, respectively. The controller is very sensitive to the ground model and the character model and it has been tested only for a very narrow region of initial states.

4.2.9 Protective step

Human subjects whose balance is disturbed quiet stance exhibit a variety of behaviors in their attempt to maintain balance. The sort of behavior they exhibit depends on their physical conditioning, personal preferences and the magnitude and duration of the disturbance. Pai and Patton [20] have studied under what circumstances a subject can maintain balance without stepping. Do *et al.* [7] have studied the biomechanical responses of human subjects to induced forward falls. They conclude that an induced forward fall starts with an invariable preparation process which is followed by an adaptable recovery one.

Our controller for the three dimensional case is currently designed to produce the visual effect. It is not sophisticated enough to actually maintain balance. However, it is designed to take a step in the proper direction. Its pre-conditions are complimentary to the pre-conditions of the balance controller. At the same time they ensure the controller does not attempt to handle situations that are not appropriate. The expected performance ensures that the controller realizes its failure early so that a fall controller can take over. The post-conditions require that the character reaches a balanced upright stance.

$\mathcal{P}$:
  Facing vector: $v_y^f > -0.6$.
  Up vector: $v_y^u > 0.7$.
  Velocity: $|\dot{\mathbf{c}}| < 1.0$ m/sec.
  Contact: hip and hands not on ground.
$\mathcal{E}$:
  Contact: hip and hands not on ground.
  Up vector: $v_y^u > 0.7$.
  Facing vector: $v_y^f > -0.6$.
$\mathcal{O}$:
  Velocity: $|\dot{\mathbf{c}}| < 0.05$ m/sec.
  Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
  Contact: hip and hands not on ground.

The three dimensional version of this controller is fairly robust in the sense that, despite its failure to regain the balance of the character, it produces a satisfactory visual effect for a variety of situations, ground parameters, and human models. The motion produced can be seen in Fig. 6.

The two dimensional version of the controller is similar. However, because the character cannot fall to the side, the controller works successfully for a wide range of disturbances. The response of the controller, i.e. the length of the step that the character takes, is parameterized with respect to the velocity of the center of mass at the time that the controller takes over. When the character is pushed backwards, the final phase of the controller makes the character's torso lean forward to facilitate a potential transition to another controller. The composable interface of the two dimensional controller is as follows:

$\mathcal{P}$:
  Up vector: $v_y^u > 0.7$.
  Velocity: $|\dot{\mathbf{c}}| < 1.0$ m/sec.
  Contact: hip and hands not on ground.
  Posture: (standing up) $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 1.0$ rad,
    where $i = (\text{thigh}, \text{waist})$, $\mathbf{q}_0 = \mathbf{0}$,
    and n is a normalization parameter.
$\mathcal{E}$:
  Facing vector: if falling forward $v_y^f < 0.1$,
    if falling backward $v_y^f > -0.3$.

Contact: hip and hands not on ground.
Up vector: $v_y^u > 0.7$.

$\mathcal{O}$:
Velocity: $|\dot{\mathbf{c}}| < 0.05$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Contact: hip and hands not on ground,
feet on ground.
Posture: (trunk leaning forward) $v_y^u * v_y^f < 0$.

The motion produced by this controller can be seen in the last part of Fig. 15. The protective-step controller has a higher priority than the fall controller, which ensures that, when appropriate, the character will first attempt to maintain balance by stepping and then resort to a fall behavior.

4.2.10 Crouch-to-stand

The crouch-to-stand controller achieves an upright stance starting at a variety of crouching states for both the three dimensional and the two dimensional case. The two dimensional case is more robust than the three dimensional one. However, both can fail even for states than can be considered as being "between" states that the controllers can handle successfully. The pre-conditions of both controllers ensure that the character is not already straight and that there is little movement. The composable interface is as follows:

$\mathcal{P}$:
Facing vector: $v_y^u > -0.6$.
Up vector: $v_y^u > 0.7$.
Velocity: $|\dot{\mathbf{c}}| < 1.0$ m/sec.
Contact: hip and hands not on ground,
feet on ground.
Posture: (not too straightened) $(1/n)\sum_i \sqrt{(q_i - q_{0,i})^2} > 0.3$ rad,
where $i = (\text{waist}, \text{thigh}, \text{knee})$,
$\mathbf{q}_0 = \mathbf{0}$.
and n is a normalization parameter.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.

$\mathcal{E}$:
Contact: hip and hands not on ground,
feet on ground.
Up vector: $v_y^u > 0.7$.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Posture: (not too straightened) $(1/n)\sum_i \sqrt{(q_i - q_{0,i})^2} > 0.03$ rad,
where $i = (\text{waist}, \text{thigh}, \text{knee})$,
$\mathbf{q}_0 = \mathbf{0}$.
and n is a normalization parameter.

$\mathcal{O}$:
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.

Fig. 11. Double stance to crouch.

Balance:	projection$(\mathbf{c}) \in \mathcal{S}$.
Contact:	hip and hands not on ground, feet on ground.
Posture:	(straightened) $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} \leq 0.3$ rad, where $i = (\text{waist}, \text{thigh}, \text{knee})$, $\mathbf{q}_0 = \mathbf{0}$. and n is a normalization parameter.

The resulting motions can be seen in frames 8–12 in Fig. 12 and Fig. 15.

4.2.11 Double-stance-to-crouch

This controller has been implemented for the two dimensional robot model only. It takes the character from a double stance to a symmetric one (crouch). The resulting motion can be seen in Fig. 11. The main function of this controller is to place the center of mass above the front foot and then bring the back leg to a symmetric position. The controller is fairly robust and it has performed successfully for a variety of starting states. The composable interface is as follows:

$\mathcal{P}$:

Facing vector:	$v_y^f > -0.7$.
Up vector:	$v_y^u > 0.7$.
Velocity:	$\lvert\dot{\mathbf{c}}\rvert < 1.0$ m/sec.
Contact:	hip and hands not on ground, feet on ground.
Posture:	leg asymmetry: $\lvert q_i - q_j\rvert > 0.2$ rad, where $i =$ left thigh and $j =$ right thigh.
Balance:	projection$(\mathbf{c}) \in \mathcal{S}$.

$\mathcal{E}$:

Contact:	hip and hands not on ground, one foot at least on ground.
Up vector:	$v_y^u > 0.7$.
Balance:	projection$(\mathbf{c}) \in \mathcal{S}$.
Posture:	leg asymmetry: $\lvert q_i - q_j\rvert > 0.1$ rad, where $i =$ left thigh and $j =$ right thigh.

$\mathcal{O}$:

Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Contact: hip and hands not on ground,
both feet on ground.
Posture: leg symmetry: $|q_i - q_j| \leq 0.1$ rad,
where $i =$ left thigh and $j =$ right thigh.
Up vector: $v_y^u > 0.7$.

4.2.12 Walk

Walking is an essential motion for a skillful simulated agent. Unfortunately it is also a difficult motion to simulate in a robust fashion. Despite the large amount of research on dynamic walking, there are no solutions that are sufficiently general with respect to different models and terrains.

Our controller for two dimensional walking implements a slow walking gait over flat terrain. The idea behind this controller is to use the swing leg to bring the center of mass above the pivot leg and then swing while the center of mass is statically balanced. When the swing leg is in front of the body then the controllers throws the character on the swing leg (now the front leg) using the ankle of the back leg. Frames from an associated animation can be seen in Fig. 15. The user can specify how many steps the character must take. The controller maintains a step counter and when the required number of steps is reached, it signals success, leaving the character in a double stance position (feet spread). The composable interface is as follows:

$\mathcal{P}$:
Facing vector: $v_y^f > -0.5$.
Up vector: $v_y^u > 0.9$.
Velocity: $|\dot{\mathbf{c}}| < 0.5$ m/sec.
Contact: hip and hands not on ground,
feet on ground.
Posture: upright: $(1/n) \sum_i \sqrt{(q_i - q_{0,i})^2} < 0.1$ rad,
where $i = (\text{thigh}, \text{waist}, \text{knee})$
and n is a normalization parameter.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
$\mathcal{E}$:
Velocity: $|\dot{\mathbf{c}}| < 1.0$ m/sec.
Contact: hip and hands not on ground,
one foot at least on ground.
Up vector: $v_y^u > 0.7$.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
$\mathcal{O}$:
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Contact: hip and hands not on ground,

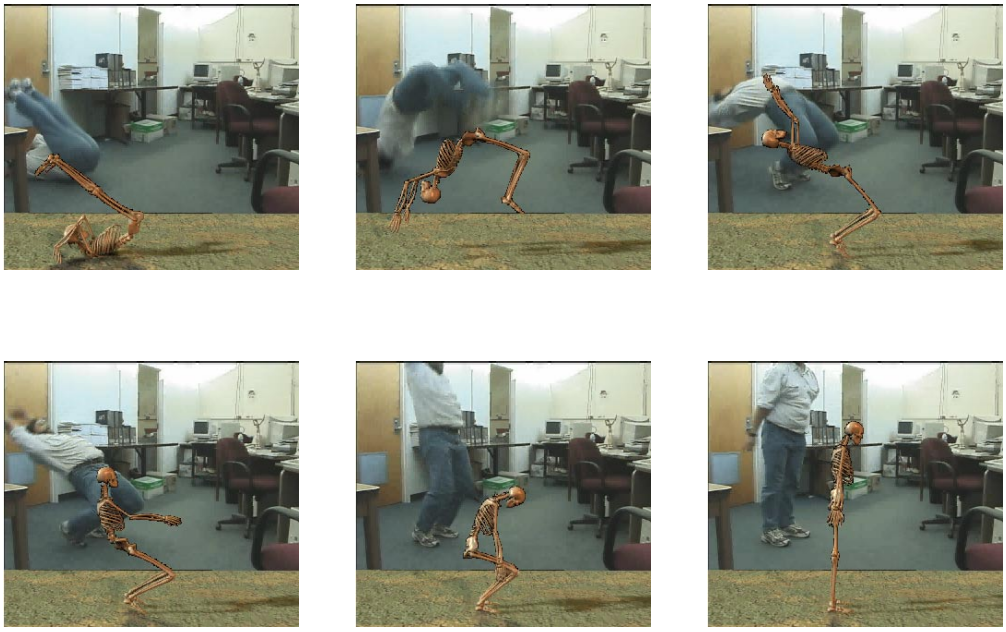


Fig. 12. The *kip* stunt performed by a real and a virtual human.

both feet on ground.
Up vector: $v_y^u > 0.7$.
Required number of steps reached.

The controller is robust with respect to a narrow region of initial upright stance configurations and it can tolerate small changes in the ground friction coefficient.

4.3 Stunts

In addition to everyday actions, our dynamic character should be able to perform a variety of interesting voluntary actions dictated by the animator. Such actions can potentially include physically dangerous stunts.

4.3.1 The kip *stunt*

The *kip* stunt, shown in Fig. 12, is an athletic motion often seen in martial arts films. It provides a very quick and energetic way to get up from a supine position. Variations of the kip are used extensively in gymnastics, Bergemann [4]. The basic mechanics of the motion are not very complex but the timing is crucial. The main idea behind the kip is to get the body airborne and at the same time provide enough rotational momentum which will allow the feet to be placed under the center of the mass. We determined the mechanics of the kip by observing human

subjects, as shown in Fig. 12. The maximum height of the center of mass during a kip varies between subjects. Subjects with a gymnastics background tend to have a smoother and more gracious motion such as the one performed by the human subject in Fig. 12. Subjects with a martial arts training tend to perform a sharper motion with less height and a landing posture closer to the crouch posture of the simulated character.

The kip controller is based on a pose controller whose pre-conditions include a variation of supine positions. The first pose of the controller makes sure that the character assumes a position suitable for performing the kip. This include straightening the legs and arms. 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. Alternatively, the kip controller could finish with the character in a crouched position and let another controller perform the crouch-to-stand action, such as the *crouch-to-stand* controller presented in Section 4.2.10. The specifications below correspond to a kip controller that leads to a crouching position. The expected performance of this controller is relatively simple because the motion is fast and largely ballistic.

$\mathcal{P}$:
Velocity: $|\dot{\mathbf{c}}| < 0.005$ m/sec.
Face up: $v_y^f > 0.97$.
Posture: lying on back: $(1/n)\sum_i \sqrt{(q_i - q_{0,i})^2} < 1.5$ rad,
where $i = (\text{thigh}, \text{knee}, \text{waist})$, $\mathbf{q}_0 = \mathbf{0}$
and n is a normalization parameter.
Contact: hip on the ground.
$\mathcal{E}$:
Horizontal Velocity: $|\dot{\mathbf{c}}_h| < 1.0$ m/sec.
Requires that the character doesn't lean sideways, $|v_z^u| < 0.05$.
$\mathcal{O}$:
Up vector: $v_y^u > 0.7$.
Velocity: $|\dot{\mathbf{c}}| < 0.1$ m/sec.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Contact: feet on the ground, hip not on the ground.

Our kip controller is very sensitive to the ground model. Implementing a more robust version of the controller would be an interesting short project. The rigid back of our articulated model, was a limitation during the development of the kip controller. The kip requires the character to roll on its back, and a flat back makes this difficult. Our controller expends more energy than one would expect while performing the initial rolling back motion.

4.3.2 Plunging and rolling

Plunging down stairs or slopes is not a task that any human subject would like to perform for the purposes of motion capture. Such dangerous stunts are better left to simulated characters. Fig. 13 shows the virtual stuntman performing a suicidal dive down stairs. The character can be instructed to lunge forward and upward at an angle specified by the user. When the hands contact the ground or 2 seconds after the last pose becomes active, the controller assumes success. This allows another controller to take over, and handle the impact with the ground, for example a gymnastic controller that can absorb the shock in a specific fashion. If such a controller does not exist then the default controller takes over and the character rolls over his head. The plunge controller bids for control of the character only when dictated by the user. For this reason and because the duration of the action is short, the pre-conditions, post-conditions and expected performance of this controller are relatively simple. The expected performance ensures that the character falls with gravitational acceleration until its hands touch the ground.

$\mathcal{P}$:
Velocity: $|\dot{\mathbf{c}}| < 0.01$ m/sec.
Face up: $-0.5 < v_y^f < 0.1$.
Contact: feet on ground, hip not on ground, hands not on ground.
Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
$\mathcal{E}$:
Before take off:
Leaning forward less than the user specified angle.
After take off:
Acceleration: $\ddot{c}_v < -9.8$ m/sec^2.
Contact with the ground in 2 seconds.
$\mathcal{O}$:
Contact: hands on ground.

Actions such as plunge and roll are well suited for a physics-based solution. The ballistic part of the motion and the collisions with the ground are solved automatically by simulating the effects of gravity and the collision forces. Although skilled animators are able to design accurate kinematic solutions for such motions, it requires considerable effort to produce kinematically the illusion of mass and the transfer of momentum. In addition, kinematic solutions must be manually adjusted to novel terrains. In contrast, physics-based controllers can likely be re-used for similar terrains. Fig. 13 and Fig. 14 show the same plunge and roll controller operating on two different terrains.

5 Animation results

We now present two sequences of autonomous and user-instructed actions that our virtual stuntmen are able to perform. The increased simulation efficiency of the 2D-

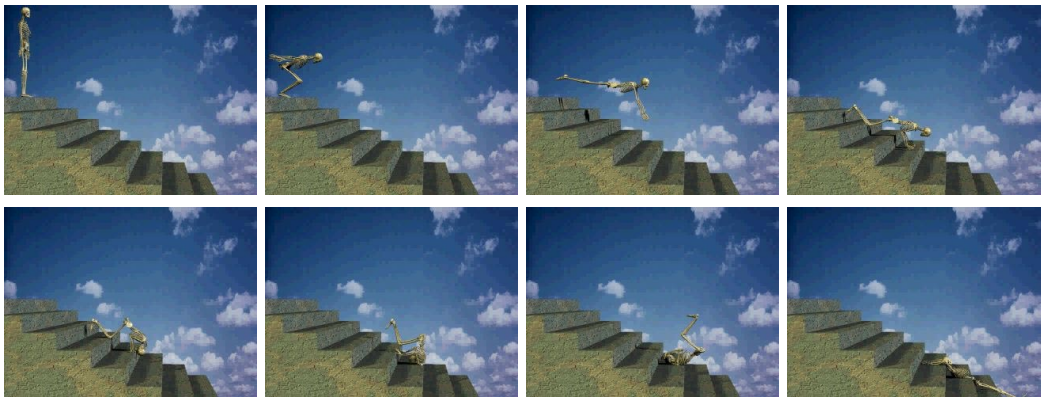


Fig. 13. Ouch!

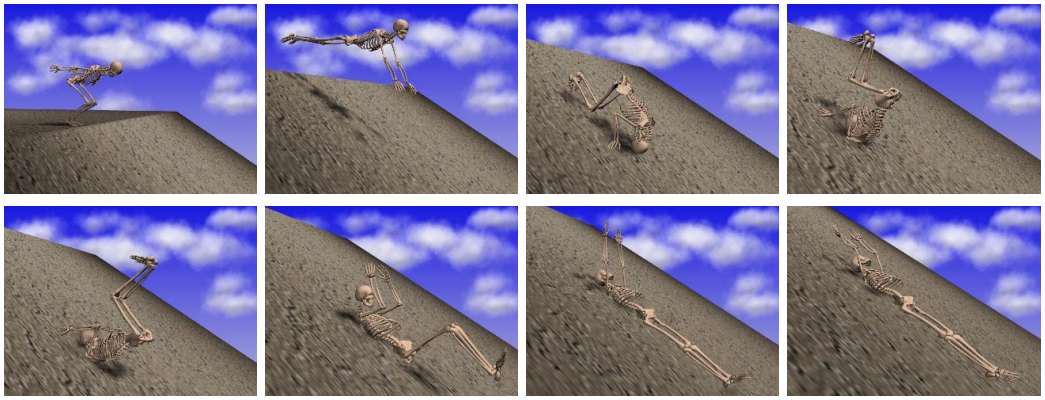


Fig. 14. Plunge and roll on a different terrain.

motion robot model permits a more productive controller design phase. In addition, the two dimensional case is more robust than the three dimensional one and it depends less on the specific character and ground model. However, the sequence we are able to achieve for the full scale three dimensional skeleton model shows that our method can be used successfully for both cases. Inevitably, robust controllers for complex motions will be developed by ourselves or by others. Our system can integrate these controllers as they become available and produce a powerful composite controller.

5.1 Robot sequence

The reduced dimensionality of the robot model allowed us to develop a relatively large number of controllers. The sequence shown in Fig. 15 involves 13 controllers: balance, prone-to-kneel, supine-to-kneel, kneel-to-crouch, crouch-to-stand, stand-to-sit, sit-to-crouch, protective-step, fall, walk, plunge-and-roll, double stance-to-crouch, and the default controller. The plunge-and-roll, stand-to-sit, sit-to-crouch and walk controllers bid for control of the character only under the direction of the user. The remaining controllers act autonomously.

Fig. 15. The robot sequence (images in raster order).

The simulation begins with the robot balancing in place. The user instructs the robot to sit down. The change of the desired goal from “null” to “sit” forces the supervisor controller to invoke the controller selection process. The pre-conditions of the plunge-and-roll, balance and stand-to-sit match the current state of the character. However, the plunge-and-roll controller bids for control only when its goal matches the user specified one. The balance controller bids for control with priority less than 10, since its goal is not “sit”. The stand-to-sit controller bids with priority higher than 10, and it therefore becomes the active controller. The robot sits on the toilet and it will stay there under the control of the stand-to-sit controller, unless something happens that either changes the state of the character or the desired goal. Soon enough, the user instructs the robot to “lean” which is the goal of the sit-to-crouch

controller. As soon as the robot is in a balanced crouch position the sit-to-crouch controller succeeds and the supervisor controller invokes a controller selection process again. The crouch-to-stand is the only controller that bids for control, and after it becomes active it takes the character to a stand position and succeeds. A new selection process allows the balance controller to take over. Then the user instructs the robot to "walk 5" which results in activation of the walk controller. The walk controller completes the five steps, bringing the character to a balanced, double stance position at the top of the stairs. The double stance-to-crouch controller takes over and brings the character to a crouch position, which allows the crouch-to-stand controller to become active followed by the balance controller. Then the user requests a "dive", which is performed by the plunge-and-roll controller. After the robot rolls over, the default controller takes over until the robot comes to rest. As soon as the velocity of the center mass is neglible, the supine-to-prone controller becomes active and brings the character to a kneeling position. Then the kneel-to-crouch controller takes over, followed by the crouch-to-stand and then the balance controller. The user throws a ball at the head of the robot, which throws the robot off balance and results in the activation of the protective-step controller, which takes a backward step. The step brings the character to a double stance position, which satisfies the pre-conditions of the double stance-to-crouch controller, which then becomes active and makes the robot assume a crouching position. As before, the crouch-to-stand controller followed by the balancing controller bring the robot to a quiet stance. This time the user throws the ball at the back of the robot's head, which results in a forward protective-step behavior, followed by the crouch-to-stand and, finally, the balance controller. The two protective-step behaviors show that the protective-step, crouch-to-stand and balance controllers are fairly robust. Finally, the user throws the ball with excessive force at the front of the robot's head. The impact induces a large acceleration to the center of mass of the character, which exceeds the pre-conditions of the protective-step controller. The robot reacts immediately with a fall behavior that attempts to use the arms to absorb the impact.

This sequence has been created interactively at about 6 frames per second. Unlike the simulation code which is highly optimized, the control code has not yet been profiled. We expect that profiling of the code will lead to improved efficiency. In addition, employing a semi-implicit method for the integration of the state instead of the current fourth order Runge-Kutta method might further improve the frame rate of the simulation.

5.2 Skeleton sequence

The three dimensional sequence shown in Fig. 1 is created interactively. The skeleton is equipped with the following controllers: balance, fall, roll-over, prone-crouch, crouch-to-stand and the default controller. All controllers are autonomous in this case; as the skeleton goes though different configurations, it automatically reacts

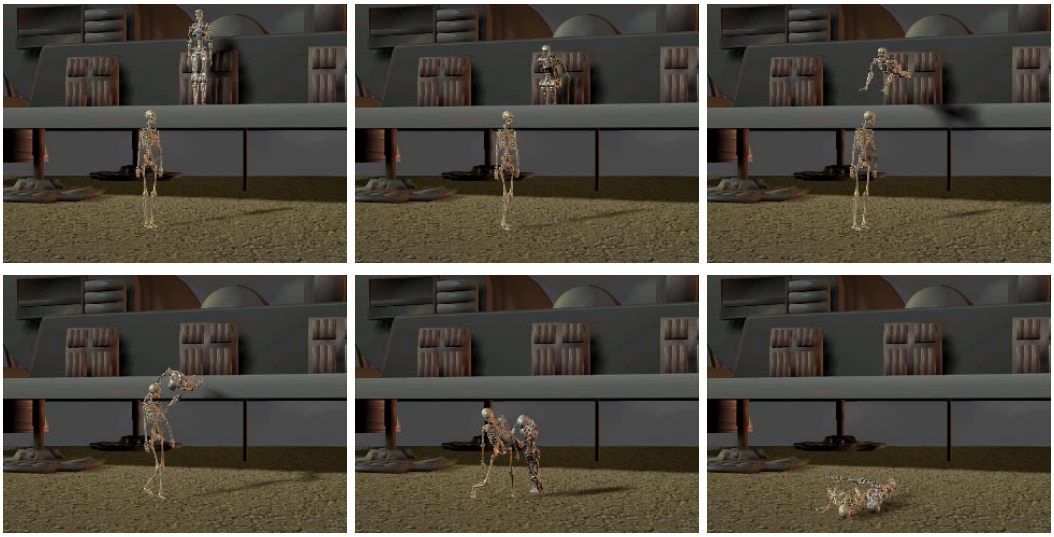


Fig. 16. Two interacting virtual characters.

to the current situation activating the most appropriate controller among those that are available. First, the user pushes the character backwards. The composite controller activates a fall behavior that tries to absorb the shock. With the character in a supine configuration, the roll-over controller brings the skeleton to a prone position which makes it possible for the prone-to-crouch controller to take over. When the character reaches a crouching posture, the prone-to-crouch controller succeeds and the crouch-to-stand controller brings the character to an upright position, which allows the balance controller to take over again. This sequence is less robust than the 2D counterpart. The prone-to-crouch and the crouch-to-stand controllers are particularly sensitive to the ground model.

An earlier version of this sequence uses a different version of the crouch-to-stand controller which cannot achieve a proper crouching position. It simply uses the left leg of the character to kick up, which resulted in a forward dive that made the character switch to a fall behavior. What was of interest was the ability of the character to fall, roll-over and return to the same kicking position over and over again, as if determined to get up, demonstrating that the roll-over controller is fairly robust. The simulation runs at approximately 2 frames per second.

5.3 Multiple characters

Our framework and associated animation system support multiple characters. Each character can have a unique composable controller scheme or share one with other characters. Fig. 16 shows an example of two interacting three dimensional characters. Currently the motion of each character is computed by a separate simulator. The simulations are performed in a layered fashion. Layered simulation has synchronization problems, Baraff [3], but it works sufficiently well for cases where

accuracy is not critical.

The robot standing on the platform, Fig. 16, is instructed to dive while the skeleton takes a step and balances in place. The timing of the tackle is scripted; the robot is not aware of the presence of the skeleton, it simply collides with the skeleton. The sequence of controllers for the respective characters is as follows:

Skeleton: Balance $\rightarrow$ Step $\rightarrow$ Balance $\rightarrow$ Protective Step $\rightarrow$ Fall $\rightarrow$ Default
Robot: Balance $\rightarrow$ Plunge $\rightarrow$ Default

Given the specialist controllers and the framework to put them together, constructing the tackle example was a simple matter of scripting the scene. The complete script that we used to produce this example is given in [8].

Our collision actuator considers the complete geometric models of each character. Each of these models has more than sixteen thousand triangles. When the two models collide the simulation becomes very slow, approximately 0.1 frames per second.

6 Conclusion

The challenges of physics-based controller design and 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 prototype virtual stuntman that has been developed based on a controller composition framework that we have previously introduced. Our stuntman can be instructed to perform interesting motor tasks and at the same time it can react autonomously to situations that arise. To our knowledge, our stuntman is the first to demonstrate a dynamic anthropomorphic 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 performing 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 activities, 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, two beckoning problems are the high-level planning of motor actions and the intelligent integration of controllers which affect only subsets of a character's degrees of freedom and can therefore execute in parallel.

Acknowledgements

We wish to thank Joe Laszlo, Victor Ng-Thow-Hing, Michael Neff and Glenn Tsang for useful discussions and suggestions and J. A. Murphy for providing the robot model. This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada and from Communications and Information Technology Ontario.

References

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

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

[3] David Baraff and Andrew Witkin. Partitioned dynamics. Technical report, Carnegie Mellon University, 1997.

[4] B.W. Bergemann and H.C. Sorenson. Dynamic analysis of the kip on the high bar. In J. Terauds and D.B. (Hrsg.) Daniels, editors, *Science in Gymnastics*, pages 44–54. Academic Publishers, Del Mar, California, 1979.

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

[6] B. L. Day, M. J. Steiger, P. D Thompson, and C. D. Marsden. Effect of vision and stand width on human body motion when standing: Implications for afferent control of lateral sway. *Journal of Physiology*, 469:479–499, 1993.

[7] 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.

[8] Petros Faloutsos. *Composable Controllers for Physics-based Character Animation*. PhD thesis, Univeristy of Toronto, Department of Computer Science, Toronto, Canada, 2001.

[9] 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.

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

[11] 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.

[12] Plamen Gatev, Sherry Thomas, Thomas Kepple, and Mark Hallet. Feedforward ankle strategy of balance during quiet stance in adults. *Journal of Physiology*, 514(3):915–928, 1999.

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

[14] 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.

[15] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. SD/FAST. Symbolic Dynamics, Inc., 1991.

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

[17] 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, New Orleans, LA, August 1996.

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

[19] Victor Ng-Thow-Hing and Petros Faloutsos. Dance: Dynamic animation and control environment. `www.dgp.toronto.edu/DGP/DGPSoftware.html`.

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

[21] 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.

[22] 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.

[23] K. Perlin. An image synthesizer. *Computer Graphics*, 19(3):287–296, July 1985.

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

[25] 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.

[26] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In Andrew Glassner, editor, *Computer Graphics (SIGGRAPH 94 Proceedings)*, Computer Graphics Proceedings, Annual Conference Series, pages 43–50, July 1994.

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

[28] 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, Dallas, TX, August 1990.

[29] Herman van der Kooij, Bart Koopman, Ron Jacobs, Thomas Mergner, and Henk Grootenboer. Quantification of sesonry information in human balance control. In *Proceedings of the 20th Annual International COnference of the IEEE Engineering in Medicine and Biology Society*, volume 20, pages 2393–2396, 1998.

[30] David A. Winter. Anthropometry. In *Biomechanics and Motor Control of Human Movement*, chapter 3, pages 51–74. John Wiley and Sons, Inc., second edition, 1990.

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

[32] Ge Wu. Distinguishing fall activities from normal activities by velocity characteristics. *Journal of Biomechanics*, 33:1497–1500, 2000.