User-Controlled Physics-Based Animation for Articulated Figures

Evangelos Kokkevis, Dimitri Metaxas and Norman I. Badler
evangelo@graphics.cis.upenn.edu {dnm,badler}@central.cis.upenn.edu
Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104-6389

Abstract

We present a physics-based system for the guided animation of articulated figures. Based on an efficient forward dynamics simulator, we introduce a robust feedback control scheme and a fast two-stage collision response algorithm. A user of our system provides kinematic trajectories for those degrees of freedom (DOFs) of the figure they want direct control over. The output motion is fully generated using forward dynamics. The specified motion trajectories are the input to a control system which computes the forces and torques that should be exerted to achieve the desired motion. The dynamic controllers, designed based on the Model Reference Adaptive Control paradigm, continuously self-adjust for optimal performance in trajectory following. Moreover, the user is given a handle on the type and speed of reaction of the figure's controlled DOFs to sudden changes in their desired motion. The overall goal of our system is to provide a platform for generating and studying realistic, user controlled motion at interactive rates. We require minimal user involvement in specifying non-intuitive parameters.

1 Introduction

Although physics-based simulation is guaranteed to generate natural looking motion, it provides the user with very little control over the results produced. In kinematically controlled animation, the user is responsible for every aspect of the generated motion; however, no guarantees can be made about the result's physical correctness. With our work we are trying to provide the direct control of kinematic animation to dynamically generated motion, hence taking advantage of the strengths of both techniques.

Given a particular task, the degrees of freedom (DOFs) of a figure can be classified as primary or secondary. Primary DOFs are instrumental in achieving the goal and distinguish that task from all others. Secondary DOFs do not involve a particular goal and usually move involuntarily; however, they significantly enhance the overall realism of the motion. For example, for a human who is casually bending forward (Fig. 1), primary are the DOFs of the back. The arm DOFs can be considered as secondary since it is not imperative for their motion to meet any goal. In such a scenario, a user should be required to specify only the way the back is bending and automatically be provided with realistic arm motion. In this paper we present an animation system which, using a robust adaptive control scheme, can link the user specified kinematic requirements on the motion of the primary DOFs to a fully automated dynamic simulation of the whole figure.

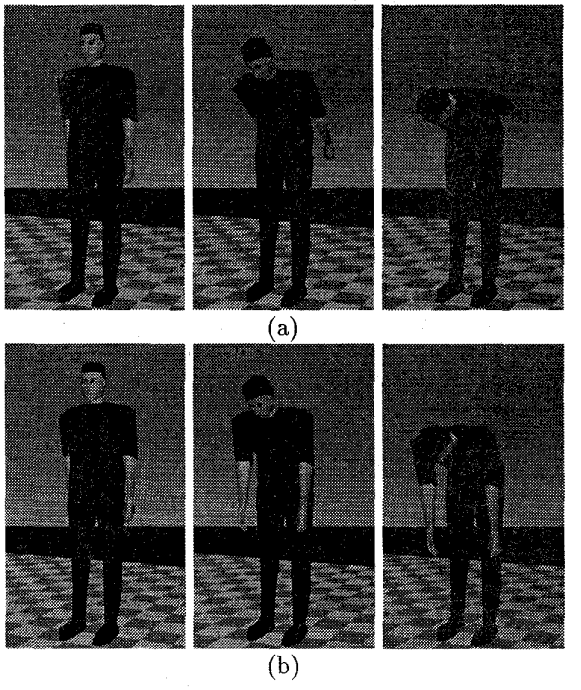


Figure 1: *Bending the back: (a) Pure kinematic control of the back, and (b) Dynamic simulation of the upper body. Notice how gravity positions the arms naturally naturally in (b).*

The desired motion of the primary DOFs is prescribed in the form of kinematic trajectories. Trajectories can be defined for either joint displacements or segment positions and orientations. For example, to animate a human figure, the user might choose to provide the desired knee and hip joint trajectory to generate a leg motion, and the position and orientation trajectory of the hand segment to generate an arm motion. Since ultimately the figure motion is generated dynamically, primary DOFs need to be

actuated—provided with a force or torque which will cause them to move. In our system, actuator forces and torques which attempt to drive the primary DOFs along the desired trajectories are computed through a self-correcting dynamic control system based on the *Model Reference Adaptive Control (MRAC)* paradigm. The robust MRAC scheme, while being simple to implement and computationally efficient, allows complex trajectories to be consistently followed. Furthermore, it provides the user with direct control over the speed and type of response of the actuated DOFs to sudden changes in their desired state.

The dynamic controllers will generate forces and torques to actuate the primary DOFs of a figure. The only other forces acting on the figure are external forces due to gravity and collisions. Collision handling is usually the bottleneck of interactive dynamic animation. In our system we devised an algorithm which breaks up the collision response problem into two distinct phases–an impact and a contact phase. The impact phase occurs when two objects first collide and results in an instantaneous change of the objects' velocities. The contact phase continues for as long as the objects are touching and produces the contact force between them that will prevent interpenetration. This two stage approach gives physically accurate collision behavior with relatively low computational cost. With all the forces acting on a figure known, a forward dynamics simulator completes the animation procedure and computes the figure motion.

The advantages of using dynamic simulation for animating a figure whose desired motion of the primary DOFs is already specified are the following: (1) Physical validity of the motion is guaranteed by the dynamic simulation which handles collisions and joint limits. (2) Natural passive motion of secondary DOFs is automatically generated, leaving the animator free to direct specific aspects of the active motion. (3) Feasibility of motions can be studied given realistic actuator limits. A desired motion will be followed as closely as possible within the specified physical constraints of the system. (4) Segment position and orientation trajectories provide powerful tools which do not suffer from the occasional jerkiness present in inverse kinematics solutions. They can be particularly useful in following trajectories provided by sensors of a motion-capture system.

Our system maintains low computational cost and small user involvement in the dynamic simulation. A recursive forward dynamics technique based on the work of Featherstone [8] was enhanced to effectively handle articulated figure collisions. There is no overhead in simulating the motion of new figures since their dynamic equations need not be derived symbolically. The dynamic control scheme used to generate driving forces for the simulation adjusts automatically to its assigned task and needs no special tuning from the user. In fact, generating an animation with our system requires no more effort than any kinematics-based system would.

The system described in this work is built as an extension to *Jack*, the human modeling and simulation package developed at the University of Pennsylvania [17, 1]. The user can take advantage of the advanced interactive manipulation features of Jack to specify complex kinematic trajectories and achieve enhanced motion through the dynamic simulation.

The remaining sections of the paper describe the animation system in detail with special emphasis given to the collision response and the dynamic control algorithms. We provide the equations used to handle impact between articulated figures and a step-by-step account of how to build a MRAC system for trajectory following.

1.1 Related Work

Researchers have tried to tackle problems in generating physically correct purposeful motion for computer animation in a variety of ways and for a variety of applications. Raibert, Hodgins and their colleagues have developed dynamic control algorithms that deal with motion of machines and simulated humans [18, 10, 9]. Their controllers were hand-crafted to assure successful and natural looking motion for the models they used. Dynamic controllers tailored for particular simulated figures were also used by McKenna and Zelzer [15] and Bruderlin and Calvert [4]. In recent years, a number of techniques for automatically generating motion for particular behaviors have been presented [22, 5] which automatically determine an optimal trajectory, a suitable control algorithm or even a morphological structure for the simulated system [19]. These approaches completely free the user from specifying the details of the motion but unfortunately their use is limited to simple systems and basic behaviors.

Our work does not deal directly with generating a particular behavior on a certain figure. Instead we provide a general system for animating articulated figures by giving the user a substantial amount of low level control. Isaacs and Cohen [11] presented a system which blends kinematic and dynamic motion in a figure by removing the kinematically controlled DOFs from the dynamic formulation for the figure. Barzel and Barr [3] control the motion of figures through constraint forces although their algorithm is not particularly suited for articulated figures with complex joints since it provides no direct control over joint angles. Our work has similarities to the work of Lamouret and Gascuel [13] who use dynamic controllers to drive motion along kinematically specified paths in space and to synchronize the animation of different objects. However, in their system, articulated figure motion cannot be described through desired joint trajectories thus complicating user control of complex figures. Furthermore, the Proportional Derivative (PD) control scheme used for the actuators lacks the robustness and ability to adapt to a variety of dynamical systems and motions provided by adaptive control, as we will explain in a section 5.

2 System Description

Figure 2 shows the overall structure of the animation system. The user input is mostly limited to defining the objects and their geometry and providing the desired motion trajectories for the primary DOFs.

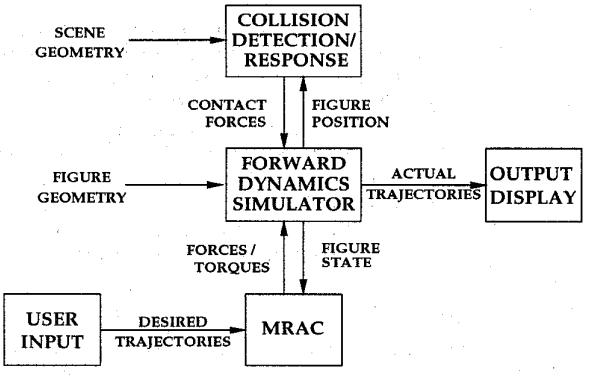


Figure 2: *The overall structure of the animation system.*

Given the specified trajectories and the figure state computed by the dynamic simulator, the MRAC calculates the forces and torques to be applied at the actuated DOFs. The collision response module interacts with the dynamic simulator to prevent penetrations between figures.

2.1 Figure Representation

A typical environment contains one or more *figures*. Each figure consists of rigid *segments* with defined geometric shape connected together through joints in a tree structure. Every segment has only one *parent segment* to which it is connected through its *parent joint*. The only exception is the *root segment* of the figure, which has no parent. Joints connecting the segments can have up to six degrees of freedom (*DOFs*), three translational and three rotational.

Attachment points on a segment are called *sites*. The root joint of each segment connects the segment's *root site* to one of its parent's sites. Every site is the origin of a Cartesian system attached rigidly to the segment. The Cartesian system attached to the root site is the *local frame* of the segment. The environment has a fixed coordinate system called the *inertial frame*.

Any kinematics based animation system should have the above parameterization to generate motion. For a dynamic simulation, the mass (or density) of each segment should be provided. Assuming uniform mass distribution, the center of mass and inertia tensor of each segment can be automatically computed from the segment's geometry. For added realism, a damping element can be added to the joints to simulate mechanical energy dissipation due to joint friction or muscular stiffness.

2.2 Desired Kinematic Trajectories

Trajectories can be specified for both joints and sites. A joint trajectory describes joint displacements over time and a site trajectory position or orientation over time. Joints or sites that have been assigned a trajectory are called *actuated* since there a force or torque acting on them will make them follow that trajectory. A joint actuator is internal to the figure and plays a role similar to a muscle or a motor. A site actuator acts much like a string on a marionette, providing an external force at the site. Typically, the user decides to power the primary DOFs which are the most important to a particular motion and allows the secondary DOFs to follow naturally. Kinematic trajectories reflect a desired motion and can be constructed in advance or while the animation is progressing. An interesting source for site trajectories is motion-captured data which can be used to drive the motion of a simulated articulated figure if sites are placed on the same positions on the figure as sensors on the real body. The advantage of using a dynamic simulation instead of playing back the raw kinematic data in this case is that we prevent physically incorrect situations (such as segment interpenetration) due either to inconsistencies between the real and the simulated world or sensor noise.

3 Efficient Forward Dynamics

A forward dynamics simulator computes the figure motion when all external and internal forces are known. For an interactive system, the simulator should be computationally efficient and able to handle arbitrary complex figures without user involvement. The forward dynamic simulator in our system is based directly on Featherstone's Articulated Body Method (ABM) [8], an efficient recursive procedure which accommodates a variety of joint types. The algorithm runs in time proportional to the number of DOFs for any articulated figure without closed loops. Contrary to the equivalent closed form Lagrangian formulation with implicit joint constraints ([6]), no symbolic preprocessing stage is necessary to develop the equations of motion and there is therefore no additional overhead for each new figure. Moreover, solutions of the closed form of the Lagrange equations typically have at least quadratic complexity.

The ABM algorithm is formulated in *spatial notation* which uses 6-dimensional vectors to represent quantities such as the velocity and acceleration of a rigid body. A spatial vector combines linear and angular components of physical quantities. The use of spatial vectors leads to elegant solutions in rigid-body dynamics problems with a significantly reduced number of equations. A complete coverage of spatial algebra can be found in Featherstone's work and we advise the reader who is unfamiliar with the spatial notation to read through the examples in Appendix A before continuing with the rest of the paper. All vector quantities in this paper appear in boldface and spatial vectors are denoted with a hat ($\hat{}$).

In its most general form, an articulated structure in Featherstone's formulation consists of links connected together with single degree of freedom joints, each with a corresponding joint axis $\hat{\mathbf{s}}$. A multiple degree of freedom joint can be represented as a chain of the appropriate number of single degree of freedom joints. The state of a figure can be fully described by the position, $\hat{\mathbf{p}}_R$, and velocity, $\hat{\mathbf{v}}_R$, of the root link and the displacement, q_l, and its derivative, $\dot{q}_l$, of the joint attached to each link l. The velocity of any link l, $\hat{\mathbf{v}}_l$, can be recursively obtained from the velocity of its

parent link p, $\hat{\mathbf{v}}_p$, the joint axis $\hat{\mathbf{s}}$ and $\dot{q}_l$ using:

$$\hat{\mathbf{v}}_l = \hat{\mathbf{v}}_p + \dot{q}_l \hat{\mathbf{s}}_l.$$

In the next sections we will describe the additions to the basic form of the forward dynamic simulator needed to complete our animation system.

4 Collision handling

One of the most important features of a dynamical simulation is the ability to automatically handle collisions between objects. Modeling impact and contact is the bottleneck of a dynamic simulation. There is two distinct aspects to collision handling: *detection* and *response*. In a simulated world, collisions between segments can be detected by examining the segment geometry, position and orientation. This process needs to be repeated at every step of the simulation and can be very costly for geometrically complex environments. When a collision is detected, appropriate action must be taken to ensure proper response. In a physics-based simulation, repulsive forces between the colliding rigid objects prevent interpenetration of the objects.

Several algorithms, varying in sophistication and efficiency, exist for detecting collision between polygonal surfaces [16]. In our system we use a fast bounding box check to pick which segment pairs could be in contact. Only for these segment pairs, a more accurate technique based on the work of [7] is applied to obtain the contact position if one exists.

The most popular technique for dealing with collision response in a dynamically simulated environment is inserting a fictitious spring-damper element at the contact point between the two segments and allowing small penetrations of the segment geometries [18, 9, 15]. The repulsive force is a function of the penetration distance and velocity. A spring-damper element is easy to implement and computationally inexpensive. Ideally, a spring should be stiff enough to allow only minimal penetration between objects even when they collide with high velocities. However, stiff springs lead to a stiff dynamical system that requires slow integration. Choosing the spring and damper constants is non-trivial since a good choice depends on the physical properties and collision velocities of the colliding objects. An interesting but more involved technique for the computation of contact forces between rigid non-articulated bodies without using spring-damper elements is described in [2].

The method we use for dealing with collision response is automated and has small effect on the dynamic simulator's integration rate. We divide the collision of two objects in two stages, the *impact* and the *contact* stage. At the impact stage, which lasts an infinitesimally short time, the velocity of the involved objects changes instantaneously whereas their position and orientation remain constant. Once a new collision is detected during the course of the forward dynamic simulation, we temporarily halt the simulation, compute new object velocities from their current velocities, and restart the simulation. When the simulation restarts, the objects are still in contact since their position and orientation did not change during the impact. If the collision is elastic and the objects separate in the next time instant, then no further steps need to be taken. If however the colliding objects do not separate and for as long as they stay in contact, repulsive forces are generated to prevent interpenetration.

For the contact stage we use a spring-damper element at every contact point. However, springs generating contact forces after impact need not be stiff. A mild spring can adequately prevent interpenetration of objects since the relative velocity of the contact points is always zero after the impact. A reasonable choice for the spring and damper constants can work for a variety of objects and motions.

In our simulated environment there can be collisions between two dynamically animated figures as well as between a dynamical and a stationary (non-dynamical) figure such as a wall or a floor. Two segments can come in contact in multiple contact points. The following section provides a description of the equations needed for the impact stage of our collision response algorithm.

4.1 Impact stage

Impact occurs when two objects collide and results in an instantaneous change in their velocities. Using the principle of conservation of momentum one can formulate an analytical solution for the impact between two arbitrary articulated figures. For strong collisions, an analytical solution is advantageous compared to the spring-damper approach, because it bypasses the problem of large collision forces. At the time of the impact, a linear system of equations is formulated and solved to obtain the post-impact velocities of the figures. The equations presented in this section grew out of the work in [16] and are adapted to match the dynamic figure representation required by Featherstone's algorithm. All the quantities in this section are assumed to be expressed in the inertial frame.

As a simple first case, we will consider a rigid object colliding with a fixed flat surface (see Figure 3(a)). If $\hat{\mathbf{I}}$ is the spatial inertia of the object then the relationship between the spatial velocity $\hat{\mathbf{v}}^-$ before and $\hat{\mathbf{v}}^+$ after the impact is given by

$$\hat{\mathbf{I}}\hat{\mathbf{v}}^+ = \hat{\mathbf{I}}\hat{\mathbf{v}}^- + \hat{\mathbf{R}}_c, \tag{1}$$

where $\hat{\mathbf{R}}_c = [\mathbf{R}_c{}^T \ \mathbf{t}_c{}^T]^T$ is the unknown collision impulse from the surface to the object. If O is the origin of the object and C is the collision point, then the velocity of point C is given by

$$\hat{\mathbf{v}}_C = [\mathbf{v}_C{}^T \ \omega^T]^T = {}_C\mathbf{X}_O \hat{\mathbf{v}}.$$

where ${}_C\mathbf{X}_O$ is a transformation from point O to point C. The collision frame is defined by three orthogonal unit vectors, $\vec{k}$ perpendicular to the contact surface and $\vec{i}$ and $\vec{j}$ in the plane of the surface. With the elasticity coefficient of the collision $e \in [0, 1]$ we use the following relationship to compute the perpendicular to

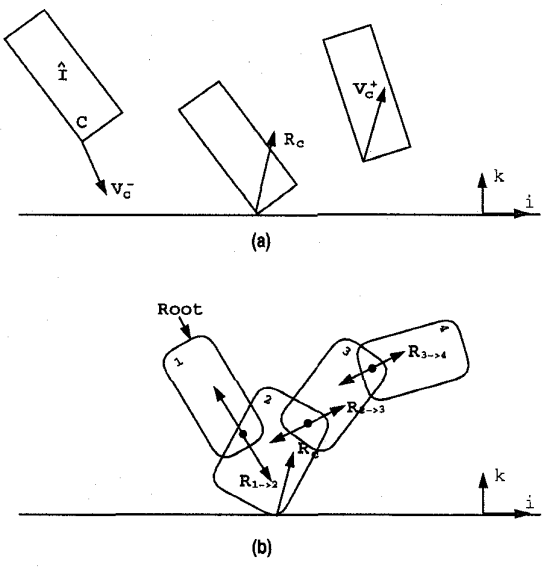


Figure 3: *Impact of (a) a simple figure and (b) an articulated figure with a fixed surface. The collision impulse is* $\mathbf{R}_c$. *The law of action and reaction holds for impulses between connected segments in a figure.*

the surface post-impact velocity of the contact point:

$$\mathbf{v}_C^+ \cdot \vec{k} = e(\mathbf{v}_C^- \cdot \vec{k}). \tag{2}$$

When $e = 0$ the collision is inelastic and when $e = 1$ the collision is perfectly elastic. If there is no friction at the contact point then there can be no impulse along the $\vec{i}$ and $\vec{j}$ directions:

$$\mathbf{R}_c \cdot \vec{i} = \mathbf{R}_c \cdot \vec{j} = 0 \tag{3}$$

and if we assume infinite friction then the velocity along the surface will be zero:

$$\mathbf{v}_C \cdot \vec{i} = \mathbf{v}_C \cdot \vec{j} = 0. \tag{4}$$

For the time being we do not handle arbitrary friction coefficients during the impact stage.

So far we have found nine linear equations (six from Eq. 1, one from Eq. 2 and two from Eqs. 3 or 4) with twelve unknowns (six for $\hat{\mathbf{v}}^+$ and six for $\hat{\mathbf{R}}_c$). The remaining three equations are obtained from the dependency of the linear and angular components of $\hat{\mathbf{R}}_c$ which take the form:

$$\mathbf{t}_c = \mathbf{r}_C \times \mathbf{R}_c \tag{5}$$

where $\mathbf{r}_C$ is the position of point C in the inertial frame.

This basic principle of momentum conservation can be readily extended from the simple solid object to an arbitrary articulated figure (see Figure 3(b)). For each segment l, the general form of Eq. 1 is:

$$\hat{\mathbf{I}}_l \hat{\mathbf{v}}_l^+ = \hat{\mathbf{I}}_l \hat{\mathbf{v}}_l^- + \hat{\mathbf{R}}_{p\to l} + \sum_{all\ d} (-\hat{\mathbf{R}}_{l\to d}) + \sum_{all\ c} \hat{\mathbf{R}}_c. \tag{6}$$

A segment l receives an impulse $\hat{\mathbf{R}}_{p\to l}$ from its parent, p, through their connecting joint. From the law of action and reaction, p receives an impulse $-\hat{\mathbf{R}}_{p\to l}$ of equal magnitude but opposite direction, and hence for each segment l we need to add the impulses received from all its children d, $-\hat{\mathbf{R}}_{l\to d}$. Finally, each active contact point c on the segment contributes an external impulse $\hat{\mathbf{R}}_c$. A joint cannot transfer impulse in a direction along its axis $\hat{\mathbf{s}}$, and therefore for each degree of freedom f of the root joint of segment l we need an equation of the form:

$$\hat{\mathbf{s}}_f^S \hat{\mathbf{R}}_{p\to l} = 0 \tag{7}$$

To compute the joint velocities right after the impact, for each DOF f we also need the equation which relates the velocity of a segment to the velocity of its parent:

$$\hat{\mathbf{v}}_l^+ = \hat{\mathbf{v}}_p^+ + \sum_f \dot{q}_f \hat{\mathbf{s}}_f \tag{8}$$

The equations relating to the contact point velocity and impulse are the same as in the simple rigid object case. For each contact point c in every segment we need Equations 2, 3 or 4 and 5. The equations described form a complete system which can be solved to obtain the post impact segment and joint velocities. Although for a complex figure the system to be solved is large, the coefficient matrix is sparse. A sparse matrix technique can be used to achieve efficiency.

Only minor modifications to the above scheme are required to generalize it for impacts between two dynamic figures or between segments of the same figure: when an impulse $\hat{\mathbf{R}}_c$ is applied to one of the two contacting segments, an impulse $-\hat{\mathbf{R}}_c$ is applied to the other. The absolute velocity of the contact point, $\hat{\mathbf{v}}_C$, is replaced by the relative velocity of the contact points of the two segments. We have also worked out the details for handling different types of common contacts between polygonal surfaces, involving edges or planes instead of single points but a full description of these is beyond the scope of this paper.

5 Dynamic Control

An animation system needs to provide means for controlling the motion to be generated. Contrary to kinematics based systems where motion is driven directly by setting kinematic trajectories such as object positions and joint angles over time, physics-based systems use as input only forces and torques. For a user, specifying a desired object trajectory in Cartesian coordinates is generally more intuitive than giving the force that would make the object move along it. If it were not for the dynamic controllers, forward dynamics would almost be useless for generating purposeful motion! A dynamic controller links the direct kinematic control with realistic looking dynamic animations.

A dynamic controller acts on a specific dynamic system ranging in complexity from a single joint to multi-DOF articulated figure. The controller and the dynamic system together are called a *control system.*

The inputs to the controller are the desired values of the system's variables, and the output is a set of forces or torques which, when applied to the system, attempt to create the desired effect on the actual values of the same variables. The goal of the control system is to minimize the discrepancy between the desired and the actual values. For example, if the dynamic system is a human arm and the desired motion is the elbow flexion, then a dynamic controller can produce the required torque at the elbow joint to flex it as desired.

There are two main classes of controllers, the *open loop* and the *feedback* controllers. Open loop controllers base their output only on the desired state of the dynamic system and are much less powerful than the feedback controllers which take into account the actual state of the system as well. A typical example of an open loop control strategy for articulated figures is *inverse dynamics* which, given the desired joint angles, velocities and accelerations of the figure, computes the required internal torques to achieve them.

The simplest to implement and most commonly used type of closed loop controller is the *Proportional Derivative (PD)* controller [11, 18, 9, 10]. The output torque of the PD controller is proportional to the difference in position and velocity between the desired and the actual state, which makes it behave similarly to a spring-damper system:

$$f = K_p(x_{desired} - x_{actual}) - K_d(\dot{x}_{desired} - \dot{x}_{actual})$$

Since a PD controller assumes nothing about the dynamical characteristics of the system to which it is applied, its successful performance relies heavily on the fine tuning of its two parameters, namely, the proportional and the derivative gains, K_p and K_d. Tuning of the controller needs to be done through trial and error and the gains depend on the characteristics of both the system and the desired motion. Gains that work fine for a slow motion could be inappropriate for a motion involving high accelerations and vice versa. An animation system using PD controllers would require laborious manual tuning of the gains to successfully achieve a desired motion.

5.1 Model Reference Adaptive Control

The *adaptive* control systems used in this work have evolved from the need to implement high performance control systems that assume little about the dynamic characteristics of the system to be controlled. The fundamental characteristic of adaptive control systems is the presence of a supplementary feedback loop acting upon a performance index of the control system. Among the many solutions which have been proposed to make a control system 'adaptive', a special class called *Model Reference Adaptive Control (MRAC)*, uses the innovative idea of a reference model to specify the desired system performance [14, 20, 21].

The advantages of using MRAC to generate the forces and torques driving a figure along kinematically specified trajectories are many: (1) MRAC systems are easy to implement and computationally efficient, (2) Since MRAC assumes little about the system it is being applied to, one basic controller design can be used to successfully control a variety of dynamic systems, (3) The self-adjusting nature of the controller relieves the user from explicitly setting gains or other non-intuitive parameters, and (4) MRAC design gives the user direct control over the ideal behavior of the controlled system. For example, an animator could vary the speed in which figures should react to changes in their desired motion, thus simulating a variety of muscular response times for animated characters.

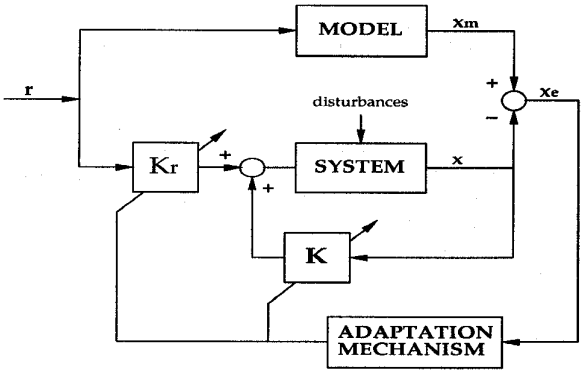


Figure 4: *The structure of a MRAC. The model and the actual system run in parallel. The difference in their state is fed to the adaptation mechanism which subsequently modifies the gains* $\mathbf{K}$ *and* K_r

Figure 4 shows a typical configuration of a model reference adaptive control system. The reference model which specifies a given index of performance in terms of inputs and model states is a part of the control system itself. The MRAC tries to force the state of the system x to follow the model state x_m even in the presence of disturbances or open loop differences. The adaptation mechanism modifies the control variables $\mathbf{K}$ and K_r and synthesizes an auxiliary input signal to assure good model following.

The dynamics of an articulated figure with n DOFs can be written as:

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{x} + \mathbf{B}(\mathbf{q})\mathbf{u} \qquad (9)$$

where $\mathbf{q}$ is the $n \times 1$ vector of the root segment position and the joint displacements, $\mathbf{x} = \left[\mathbf{q}^T \dot{\mathbf{q}}^T\right]^T$ is the $2n \times 1$ state vector and $\mathbf{u}$ is the vector of external and internal forces acting on the figure. Matrices $\mathbf{A}$ and $\mathbf{B}$ can be symbolically derived from the figure structure and a *coupled* MRAC can be designed to control the motion of any number of the figure's DOFs.

Alternatively, the same dynamic system with n DOFs can be written as n 1-DOF dynamic simpler systems each one dealing with a single DOF q of the original system. The equation of each simple system can be written as :

$$\dot{\mathbf{x}}_q = \mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{x}_q + \mathbf{B}(\mathbf{q})u_q \qquad (10)$$

or

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_{q1}(\mathbf{q}) & -a_{q2}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ b_{q1}(\mathbf{q}) \end{bmatrix} u_q.$$

The n simple systems are coupled and therefore scalars a_{q1}, a_{q2} and b_{q1} depend on the values of all the DOFs.

This set of dynamic systems can be controlled using a *decoupled* MRAC approach: A simple controller can be attached to each DOF of the figure that needs to behave in a specified manner. Decoupled MRACs have been proven to be robust and nearly as effective as the coupled MRAC while at the same time they are easier to implement and are computationally more efficient. For each DOF q of the figure, we construct a *reference model* system which has the form:

$$\dot{\mathbf{x}}_{mq} = \mathbf{A}_{mq}\mathbf{x}_{mq} + \mathbf{B}_{mq}r_q, \tag{11}$$

and the model following error for q is defined as:

$$\mathbf{x}_{eq} = \mathbf{x}_{mq} - \mathbf{x}_q \tag{12}$$

To complete the MRAC, we define the control input u_q of Eq. 10 to be:

$$u_q = \mathbf{K}_q(t)\mathbf{x}_q + K_{rq}(t)r_q, \tag{13}$$

where $\mathbf{K}_q(t)$ is a 1×2 vector and $K_{rq}(t)$ scalar. The objective of the MRAC is to force x_{eq} asymptotically to zero in a controlled fashion and achieves that by setting the proper values of the adaptive gains $\mathbf{K}_q$ and K_{rq}.

We devote the following section to present a step by step procedure on how to synthesize a MRAC to do trajectory following for a single DOF of a dynamic figure. The decoupled MRAC approach is elegant for controlling multiple DOFs at the same time, can be achieved by assigning a different MRAC to each DOF.

5.2 Sample MRAC synthesis

This section gives a 'cookbook' description for a procedure which results in a successful MRAC synthesis. Model Reference Adaptive Control is a powerful technique and the interested reader should consult the original published work in [14, 20, 21] for more details.

Assume that a joint trajectory is given from which we can extract the joint angle and velocity at any time instant. The goal is to synthesize a MRAC which generates the torque within specified limits which would move the actual joint close to the trajectory in a controlled manner. The steps to be followed are:

1. *Design the reference model*: The dynamical systems we are interested in are driven by finite forces and therefore cannot change their state instantaneously. The greater the force applied to the system, the faster its state changes. Given a particular goal position, if the force supplied is too low, the system will move too slowly. If the force is too great, the system will reach the goal quickly but will not be able to stop in time and will overshoot, as shown in Figure 5. The model system will act as an index of performance of the actual system. We therefore need to design it so that it behaves the way we would like the actual system to behave. The MRAC will try to minimize the difference between the model and the actual system's state, thus driving the actual system as closely as possible to its ideal behavior. The model system is defined from Eq. 11 where

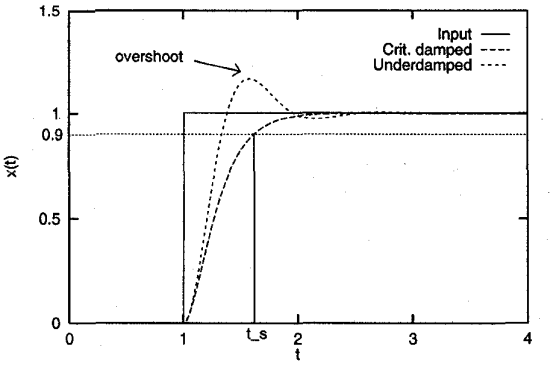


Figure 5: *Different responses to a step input. Underdamped motion reaches the desired state faster but overshoots. Critically damped motion provides the fastest response without overshooting. The settling time t_s is shown for the critically damped case.*

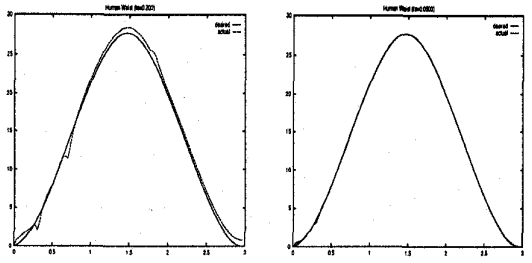
$$\mathbf{A}_{mq} = \begin{bmatrix} 0 & 1 \\ -a_{mq1} & -a_{mq2} \end{bmatrix} \quad \mathbf{B}_{mq} = \begin{bmatrix} 0 \\ b_{mq} \end{bmatrix}.$$

For good trajectory following, a critically damped behavior of the model system gives the fastest response possible with no overshoot. Critically damped behavior can be achieved by setting: $a_{mq1} = b_{mq} = \lambda^2$, $a_{mq2} = 2\lambda$ with $\lambda = 4/t_s$ where t_s is the required *settling time* of the system [12]. The settling time can be set by the user to adjust the speed of response of the controlled system. Typically the settling time is defined as the time taken for the system to reach within 10% of its desired value. The lower the settling time, the faster the system has to move to its desired state which translates to higher force or torque generated by the controller. By setting lower settling time for a figure's actuated DOFs, an animator can achieve quick responses. A high settling time would give a sluggishly moving figure.

2. *Generate the model input*: With the model system defined, the controller is ready. The model and the actual systems will be integrated at the same time to observe the difference in their states. At each time instant, the input r_q to the model system is determined by inverting the reference model dynamics:

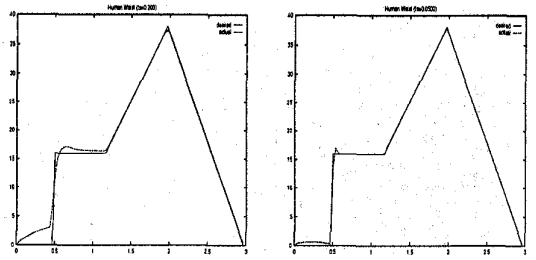
$$r_q = \mathbf{B}_{mq}{}^{+}(\dot{\mathbf{x}}_{dq} - \mathbf{A}_{mq}\mathbf{x}_{dq}) \tag{14}$$

where $\mathbf{B}_{mq}{}^{+} = [0 \;\; 1/b_{mq}]$. This input will instantaneously drive the model system towards the desired state x_{dq} obtained from the user provided desired trajectory of DOF q.



(a) $t_s = 0.2$ (b) $t_s = 0.05$

Figure 7: *Trajectory following for the human waist on a smooth motion. The lower the value of the settling time t_s the more accurate the following.*



(a) $t_s = 0.2$ (b) $t_s = 0.05$

Figure 8: *Trajectory following for the human waist on a discontinuous motion. By varying t_s one gets different response time of the joint to changes in the desired state.*

3. *Compute Control Signal*: The control signal u_q is computed from the following equations ([14]):

$$\mathbf{x}_{eq} = \mathbf{x}_{mq} - \mathbf{x}_q$$

$$y = \begin{bmatrix} \frac{16}{t_s^2} & \frac{4}{t_s} \end{bmatrix} \mathbf{x}_{eq} \quad (15)$$

$$\mathbf{K}_q(t) = \int_0^t \alpha\, y\, \mathbf{x}_q^T\, d\tau + \beta y \mathbf{x}^T \quad (16)$$

$$K_{rq}(t) = \int_0^t \alpha\, y\, r\, d\tau + \beta y r \quad (17)$$

$$u_q = \mathbf{K}_q(t) + K_{rq}(t) r_q$$

Equations 16 and 17 are the heart of the adaptation process and guarantee the asymptotic convergence of x_{eq} to 0 [20]. Constants α and β depend on the characteristics of the dynamic system but can be chosen within a wide range of values and do not have a significant effect on the trajectory following performance of the controller.

To enhance the realism of a dynamic animation it is necessary to limit the outputs of the controllers just as the outputs of mechanical motors and human muscles are limited. In the MRAC scheme, when the control signal computed from Eq. 13 is higher than the maximum allowed torque then the integration in Eqs. 16 and 17 must be halted and the old values of the integrals should be kept. The output of the controller is set to the maximum allowed value and the integration is stopped until the signal returns to within its specified range.

6 Results

We have conducted a series of experiments to test our system and determine possible limitations. We have applied the forward dynamic simulator with the collision handling module to a variety of articulated figures, from simple pendulums to a human figure model. For simple articulated figures with up to five segments, computation time is not a problem; typically simulations can go faster than real time. As figures get more complicated, the running time increases but interactive rates can be still be maintained especially when the number of impacts is not excessive. Impact equations for a complex figure result in a large linear system which is expensive to solve. However, impacts do not cause a decrease in the integration time step like stiff springs do; furthermore, they preserve the numerical stability of the dynamic system much better. A difficult test case for the collision response system is the simulation of a soldier falling on his back after being shot in the chest. The velocity of the soldier's back when he hits the ground combined with the large mass of his body give rise to high impact forces. Very slow integration would be required if springs were used to handle the impact. Our system handles the impact and the contact at the back and the legs without a significant delay. The motion shown in Figure 6 is generated by giving a large initial impulse at the middle of the soldier's upper torso and letting him fall to the ground.

The adaptive controllers proved to be powerful in driving the controlled DOFs along desired trajectories. A number of trajectories were used to drive the joints and sites of articulated figures. As expected, for smooth desired trajectories there was almost no error (Figure 7). On discontinuous trajectories like the one shown in Figure 8, the error varies depending on the choice of the settling time t_s of the model reference system. Lower settling time forces faster response to changes in the state, while higher settling time makes the system take to adjust and results to greater error. For both figures we used a human figure model which was dynamically animated from the waist up. We actuated the waist joint and we allowed the rest of the joints (arms and upper torso) to move freely. Despite the disturbances created by the free motion of the rest of the joints, the waist followed the desired trajectory accurately. This shows the power of the MRAC to continuously adapt to the system it controls and to overcome external disturbances.

Figure 6 demonstrates an attempt to follow a physically incorrect trajectory which involves penetration between the arms and a table. Although kinematically the motion is valid, when played back dynamically the contacts are detected and penetration is pre-

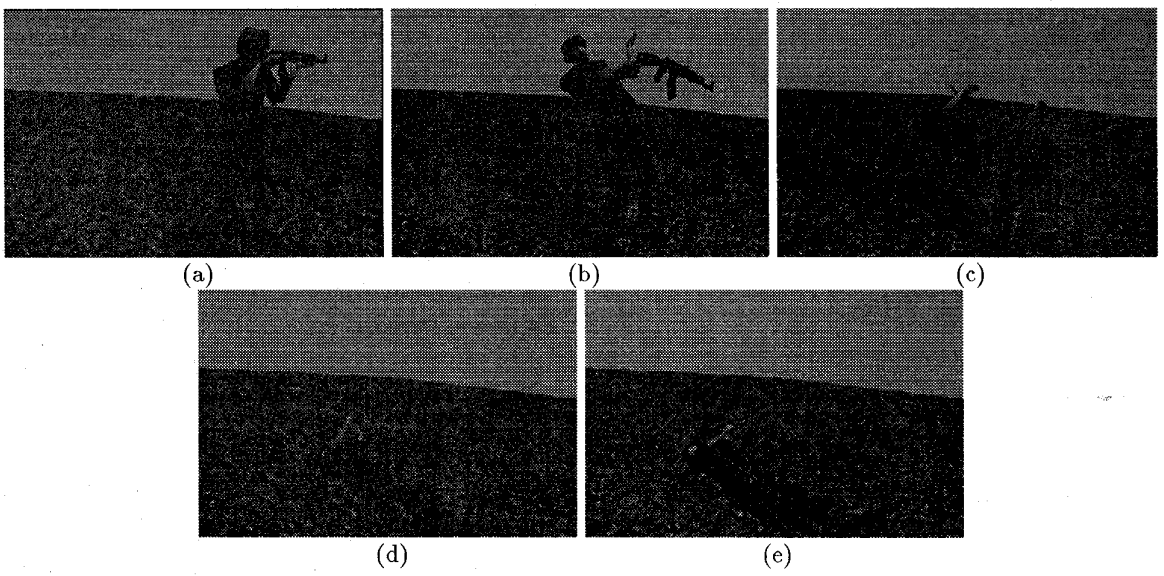


Figure 6: *A wounded soldier. The soldier receives a large impulse at his upper chest and falls on the ground.*

vented. A motion trajectory is originally provided for the shoulders and the elbow. Once the contact forces prevent any further motion, the output of the joint controllers reaches its maximum value and stays there, stubbornly trying to make the arms follow the desired motion.

7 Conclusion

In this paper we present an animation system capable of generating physically correct, user-controlled motion for articulated figures. We introduced a fast, two-stage collision response algorithm, which together with an efficient forward dynamics simulator enable interactive animation rates. Robust, self-correcting control units generate forces which drive the motion of actuated DOFs of the figure along user specified kinematic trajectories. The user can have direct control over the response characteristics of the actuated parts of the figure, without having to tune unintuitive parameters of the dynamic system. Kinematic trajectories can be given either as joint displacements or as segment positions or orientations thus providing a variety of ways to specify desired motion.

The dynamic controllers used in this work do well in following specified kinematic trajectories. We are currently working on using them to generate lower body motion and eventually dynamic human locomotion given specific leg motion trajectories. Moreover, we are experimenting on using the positions and orientations of sensors from our motion capturing system to generate site trajectories and drive an articulated figure. The results we have so far are promising. As a future direction, we would like to investigate ways to automatically generate new or adjust existing trajectories while an animation is progressing. The new trajectories would depend on high level goals (such as maintaining balance or reaching for an object) and the current dynamic state (velocities, external forces) of the animated figure.

8 Acknowledgments

We would like to thank Prof. Zoe Doulgeri for the invaluable insights she provided us on the design and implementation of the MRAC. This research is partially supported by DMSO and U.S. Air Force WPAFB/HRGA DAAH04-94-G-0402, ARPA DAMD17-94-J-4486, and ARO DURIP DAAH04-95-1-0023.

APPENDIX A: Examples of spatial notation

This is a brief summary of the important spatial quantities used in this paper. It is by no means a complete list but should be sufficient for the understanding of our work.

- *Velocity*: The instantaneous velocity of a rigid body with respect to an origin O may be described by the linear velocity $\mathbf{v}_O$ of a point moving with the object, instantaneously coincident with O and the object's angular velocity ω. In spatial notation we say that $\hat{\mathbf{v}}$ is the spatial velocity of the rigid body, where $\hat{\mathbf{v}} = [\omega^T \ \mathbf{v}_O{}^T]^T$.

- *Joint Axis*: The joint axis is a spatial vector which defines the direction and nature of motion allowed by the joint. The relative velocity of two bodies is obtained by multiplying the joint axis by the scalar joint velocity. A revolute joint which allows rotation about an axis $\mathbf{s}$ is represented as $\hat{\mathbf{s}} = [\mathbf{s}^T \ \mathbf{0}^T]^T$. A prismatic joint which allows pure translation along axis $\mathbf{s}$ has the form $\hat{\mathbf{s}} = [\mathbf{0}^T \ \mathbf{s}^T]^T$.

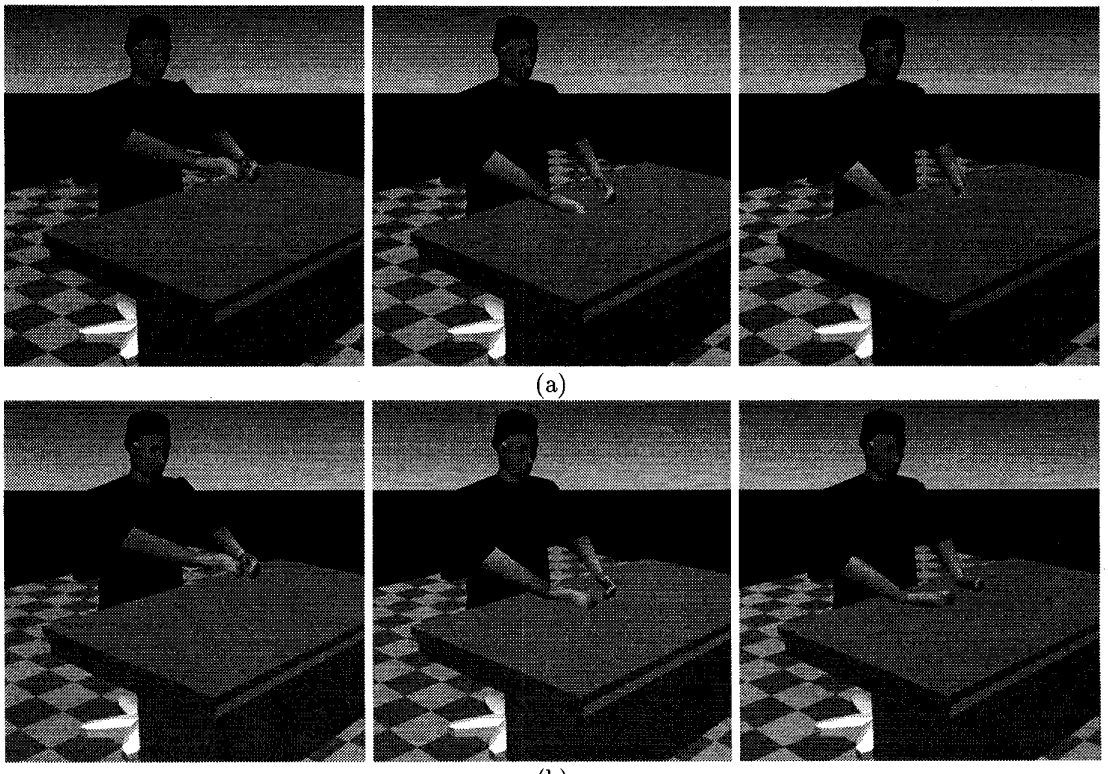


Figure 9: *Following a physically incorrect motion trajectory: (a) Pure kinematics allows the motion, whereas (b) Dynamic simulation prevents interpenetration between the arms and the desk.*

- *Force*: Any number of forces acting on a rigid body can be represented as a force $\mathbf{f}$ acting along a line passing though the origin O together with a couple τ_O. The spatial representation of the reduced forces is $\hat{\mathbf{f}} = [\mathbf{f}^T \ \tau_O{}^T]^T$.

- *Spatial Coordinate Transformation*: A spatial transformation matrix ${}_P\mathbf{X}_O$ is a 6×6 matrix transforming a spatial quantity from frame O to frame P.

- *Spatial Rigid-Body Inertia*: The spatial rigid-body inertia $\hat{\mathbf{I}}$ is a 6×6 matrix which transforms the spatial velocity $\hat{\mathbf{v}}$ of a rigid body into its spatial momentum $\hat{\mathbf{P}}$ through $\hat{\mathbf{P}} = \hat{\mathbf{I}}\hat{\mathbf{v}}$.

- *Spatial Transpose*: The spatial transpose of a vector $\hat{\mathbf{a}} = [\mathbf{a}^T \ \mathbf{a}_0{}^T]^T$ is a 1×6 vector $\hat{\mathbf{c}}^S = [\mathbf{a}_0^T \ \mathbf{a}^T]$.

References

[1] Norman I. Badler, Cary B. Phillips, and Bonnie Lynn Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, 1993. ISBN 0-19-507359-2.

[2] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 23–34. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[3] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 179–188, August 1988.

[4] Armin Bruderlin and Thomas Calvert. Goal-directed, dynamic animation of human walking. In *Computer Graphics (SIGGRAPH proceedings)*, volume 23, pages 233–242. ACM, July 1989.

[5] Michael F. Cohen. Interactive spacetime control for animation. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 293–302, July 1992.

[6] John J. Craig. *Introduction to robotics: mechanics and control.* Addison-Wesley, 1989.

[7] D.W. Johnson E.G. Gilbert and S.S. Keerthi. A fast procedure for computing the distance between objects. *IEEE Journal of Robotics and Automation*, 1988.

[8] Roy Featherstone. *Robot dynamics algorithms.* Kluwer Academic Publishers, 1987.

[9] Jessica K. Hodgins, Paula K. Sweeney, and David G. Lawrence. Generating natural-looking motion for computer animation. In *Proceedings of Graphics Interface '92*, pages 265–272, May 1992.

[10] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 71–78. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[11] Paul M. Isaacs and Michael F. Cohen. Mixed methods for complex kinematic constraints in dynamic figure animation. *The Visual Computer*, 4(6):296–305, December 1988.

[12] B.C. Kuo. *Automatic Control Systems.* Prentice Hall, 1991.

[13] Alexis Lamouret and Marie-Paule Gascuel. Scripting interactive physically-based motions with relative paths and synchronization. In *Proceedings of Graphics Interface '95*, pages 18–25, May 1995.

[14] I.D. Landau. *Adaptive control, the model reference approach.* Marcel Dekker, New York, 1979.

[15] Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. In *Computer Graphics (SIGGRAPH proceedings)*, volume 24. ACM, August 1990.

[16] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 289–298, August 1988.

[17] Cary B. Phillips and Norman I. Badler. Interactive behaviors for bipedal articulated figures. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 359–362, July 1991.

[18] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 349–358, July 1991.

[19] Karl Sims. Evolving virtual creatures. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 15–22. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[20] D.P. Stoten and H. Benchoubane. Empirical studies of an mrac algorithm with minimal control synthesis. *International Journal of Control*, 51(4):823–849, 1990.

[21] D.P. Stoten and H. Benchoubane. The decentralized minimal control synthesis algorithm. *International Journal of Control*, 56(4):967–983, 1992.

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