MIT-CSAIL-TR-2006-012 February 28, 2006

Interactive Animation of Dynamic Manipulation

Yeuhi Abe and Jovan Popovic

Interactive Animation of Dynamic Manipulation

Yeuhi Abe Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

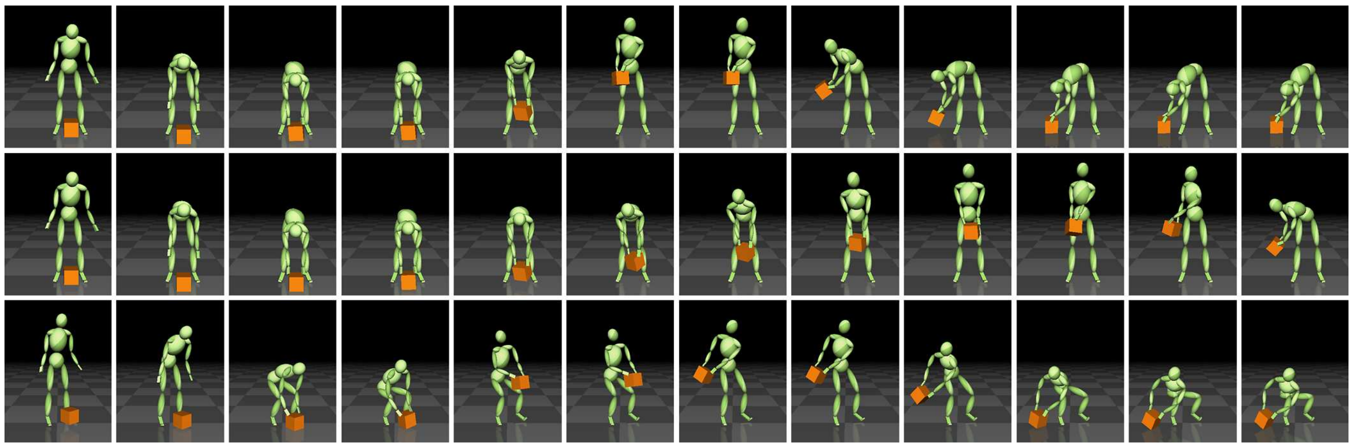

Figure 1: Our multi-task control algorithm directs a real-time simulation of a character to accomplish manipulations, such as displacing a box (top row). Manipulations are compactly described. In the above example, only four Cartesian goal positions are used to describe the motion of the hands and the box. The missing details are filled in with a secondary posture task that incorporates recorded motion postures from a similar performance. The control adapts naturally to changes in the environment. As expected, increasing the weight of the box (second row) produces a slower lift. The performance of the task can also be changed by using a different recorded motion in the posture task (third row).

Abstract

Lifelike animation of manipulation must account for the dynamic interaction between animated characters, objects, and their environment. Failing to do so would ignore the often significant effects objects have on the motion of the character. For example, lifting a heavy object would appear identical to lifting a light one. Physical simulation handles such interaction correctly, with a principled approach that adapts easily to different circumstances, changing environments, and unexpected disturbances. Our work shows how to control lifelike animated characters so that they accomplish manipulation tasks within an interactive physical simulation. Our new multi-task control algorithm simplifies descriptions of manipulation by supporting prioritized goals in both the joint space of the character and the task-space of the object. The end result is a versatile algorithm that incorporates realistic force limits and recorded motion postures to portray lifelike manipulation automatically.

CR Categories: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

Keywords: Motion Control,Physically Based Animation,Robotics

1 Introduction

Animated characters are often required to manipulate objects in their environment, but lifelike animation of object manipulation is much easier to describe than to execute. Descriptions are often underdetermined because most tasks can be accomplished in any number of ways. For example, the description of a reaching task may specify only hand positions while leaving it up to the imprecise notions of style and preference to determine the motion of the rest of the body.

Animation of object manipulation also involves complex physical interactions between characters, objects, and their environment. Consider, for example, a character steadying one end of a rope with its hands while counteracting the forces propagated through the rope from the other end. In interactive applications, kinematic techniques would generate an unresponsive character and in authoring applications, animators would have to create proper responses by hand. On the other hand, physical simulation could synthesize such animation procedurally, given proper control for the character.

Our work addresses both the problem of underdetermined motion descriptions and the problem of handling complex physical interaction by controlling characters manipulating objects within interactive physical simulations. Physical simulation frees animators from tedious accounting of physical effects and automatically supports dynamic, unpredictable scenarios in interactive applications. In the animation of passive natural phenomena (e.g. fluid, cloth, or hair), physical simulation has all but supplanted the more traditional kinematic methods because of the ease with which it handles simple but numerous micro interactions. In constrast, kinematic methods remain the technique of choice for the animation of active characters because of the many difficulties involved in designing motor control algorithms capable of generating lifelike motion.

Physically valid object manipulation, in its most general form, challenges the state-of-the-art of both simulation and control. Object manipulation requires simulation techniques that can accurately resolve numerous frictional contacts and collisions. Simulation must be robust and accurate because the success of fine manipulation often depends the arrangement of contact points and subtle applications of contact forces. Manipulation also requires careful planning

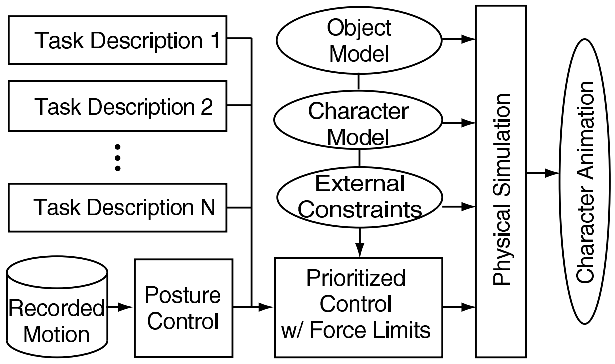

Figure 2: Algorithm Overview. Our control algorithm combines multi-priority task descriptions with low-priority posture preferences inferred from recorded motion to control an interactive physical simulation of a character manipulating objects.

of strategies that can accomplish requested tasks. The sheer complexity of bodies, hands, and arms demands a new generation of control algorithms capable of performing such strategies.

Plausible animation of manipulation need not tackle all these challenges at once. In this work we make the simplifying assumption that proper task descriptions are provided by the animator (or a high-level state machine) and that contacts are preserved automatically by clamping constraints that affix points on one object to another. We demonstrate that lifelike animations emerge even with these simplifying assumptions and we propose a versatile control framework that supports many of the extensions needed to solve the general manipulation problem (such as exact control over the forces exerted on hands and feet).

The primary contribution of this work is a control algorithm that guides complex characters, with many degrees of freedom, through lifelike portrayals of common manipulation tasks, within physically simulated, interactive environments. Our algorithm, illustrated in Figure 2, simplifies the description of manipulation tasks by acting on multiple task descriptions simultaneously. The advantage of this approach is that it allows for a decoupled description of the precise motion details (e.g., the position of hands needed to grasp or carry objects) and the secondary motion objectives (e.g., the desired posture of the character).

The key to our versatile control is a new prioritized formulation that prevents lower priority posture tasks from interfering with higher priority manipulation tasks without fully sacrificing the ability to track lifelike postures accurately. The control algorithm can exploit accurate tracking in a secondary posture task to generate lifelike animations by incorporating recorded motion postures. A unique feature of this approach is that it benefits from motion data even when it does not obey recorded timings or sequence order. Thus, new motions can vary significantly from the recorded motion used to guide the posture. Realistic timing emerges naturally as a consequence of task descriptions that limit the force applied by the hands. For example, a recorded motion of a person lifting a light object quickly can be used to guide the posture of a simulated character lifting a heavy object slowly, even if a significantly different motion trajectory is used for the object.

2 Background

Compact task descriptions, which command only essential details such as hand position or applied force, are preferred in both manual [Lee et al. 1990] and automatic [Koga et al. 1994] task planning because they suppress irrelevant aspects of task execution. Animation systems can use inverse kinematics to infer full postures from such task descriptions, making it easy to reuse tasks in different environments or in performances by different (shorter or longer-armed) characters [Yamane et al. 2004]. Lifelike postures, however, require that such algorithms either incorporate recorded motion data or leverage prior results from neurophysiology and other studies of natural motion [Koga et al. 1994; Rose et al. 2001; Grochow et al. 2004; Yamane et al. 2004]. Our paper addresses a similar problem for animation of manipulations with significant dynamics such as catching, throwing, and other more complicated tasks. We show how to generate lifelike animation of dynamic manipulation by incorporating motion data into the control needed to accomplish tasks.

Animations of dynamic manipulation must account for both the dynamics and the kinematics of tasks because static considerations alone will not generate lifelike motion [Lee et al. 1990]. If dynamic considerations are ignored, lifting a heavy object will look identical to lifting a light object despite the fact that one task requires increased effort and a different motion. Motion learning techniques resolve this problem with data sets that explore variation in task performance. [Rose et al. 1998; Mukai and Kuriyama 2005; Kovar and Gleicher 2004]. This is especially effective when tasks can be restricted to small, well sampled manipulations such as lifting and reaching, but generalization to all types of manipulation requires solutions to remarkably difficult machine-learning problems [Duda et al. 2000]. To extend the range of a limited data set, current interactive applications rely on motion-editing tools that approximate dynamics with temporal smoothness [Bruderlin and Williams 1995; Witkin and Popović 1995; Gleicher 1997; Choi and Ko 2000] because dynamically consistent editing tools have not been designed for interactive use [Popović and Witkin 1999; Liu and Popović 2002; Sulejmanpasić and Popović 2005]. Our work shows how to account for dynamics in interactive animations of object manipulation within a physical simulation.

In interactive physical simulations, a simple replay of a precomputed task execution ignores complex physical interactions. Animations appear artificial because task execution proceeds blindly, ignoring the effects of other objects in the environment. Even if motions are preplanned to account for the weight of manipulated objects, playback is not affected by collisions or other dynamic interaction with the environment. The executions of such preplanned trajectories can be made reactive through the use of joint-space control that tracks computed trajectories [Zordan and Hodgins 2002; Yin et al. 2003]. Although joint-space control has also been successful in animation of lifelike locomotion and other activities [van de Panne et al. 1990; Raibert and Hodgins 1991; Hodgins et al. 1995; Grzeszczuk and Terzopoulos 1995; Laszlo et al. 1996; Faloutsos et al. 2001], many manipulation tasks are easier to describe in Cartesian coordinates. Our control algorithm eases the animation of dynamic manipulation by supporting task descriptions in both Cartesian and joint coordinates.

Cartesian control, also known as operational space control, linearizes dynamics of the operational point to enable direct description of manipulation tasks [Khatib 1987]. In addition, the operational space formulation simplifies control of complex characters with many degrees of freedom by decoupling the control needed to accomplish a task from the control of task-redundant degrees of freedom. Our control algorithm uses a similar approach to generate lifelike animations with *precise* control of task-redundant degrees of freedom.

Precise control requires a carefully formulated decoupling that accounts for the dynamics in the task-redundant degrees of freedom. Our approach is inspired by the recently proposed task-consistent decoupling of unconstrained dynamics for branching joint structures [Khatib et al. 2004a]. We reformulate this approach to simplify the implementation and extend it to decouple both unconstrained and constrained dynamics. These extensions are particularly important for animation of dynamic manipulation because constraints (or closed-loop joint structures) emerge whenever a character uses both hands or stands with both feet on the ground. Our simplified formulation is also a key to precise control that is required for lifelike animation. Unlike the alternative constrained formulation [Sapio and Khatib 2005], our technique enables both the decoupling of constrained dynamics and the precise control of task-redundant degrees of freedom. This allows us to control posture (or style of motion) precisely, using the available motion data.

Solutions to more general manipulation problems will require comprehensive efforts that incorporate current models of hands and their behavior [Albrecht et al. 2003; ElKoura and Singh 2003; Pollard and Zordan 2005]; planning strategies for manipulation and grasping [Yamane et al. 2004; Li and Pollard 2005]; and improved models of natural posture [Grochow et al. 2004; Mukai and Kuriyama 2005]. One contribution of our work is to show how these approaches could be unified to control whole-body motions in interactive animations of dynamic manipulation.

3 Control Algorithm

Our control algorithm computes the joint torques that cause animated characters to accomplish desired manipulations. The algorithm can be used with physical simulation to author new motions or to execute flexible motion control strategies interactively. It is particularly suitable for these purposes because it supports compact task descriptions and the prioritization of conflicting tasks, both of which can simplify the way that motion is commanded. For example, the control algorithm can favor natural postures at a low priority level without interfering with the primary manipulation task at a high priority level.

In this section, we derive the basic control algorithm for unconstrained, open-loop structures before extending it to the most practical case: constrained dynamics with unactuated degrees of freedom. The end result is a procedure that transforms complex nonlinear dynamics into simple second-order linear systems whose intuitive control is explained in Section 4.

3.1 Unconstrained Dynamics

The dynamics of animated characters is modeled as a set of rigid body limbs constrained by a set of joints that link the limbs into a core body structure. When this structure forms a tree graph, also called an open-loop configuration, the pose of the character can be described by a set of independent joint variables (see Figure 3). These independent coordinates $\mathbf{q}$ allow for the dynamics of the character to be expressed in a standard numerical form:

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}), \tag{1}$$

where $\mathbf{M}$ is the joint-space inertia matrix and $\mathbf{h}$ is a nonlinear function of all acceleration-independent terms that computes the gravitational, centrifugal and Coriolis forces. Physical simulations can

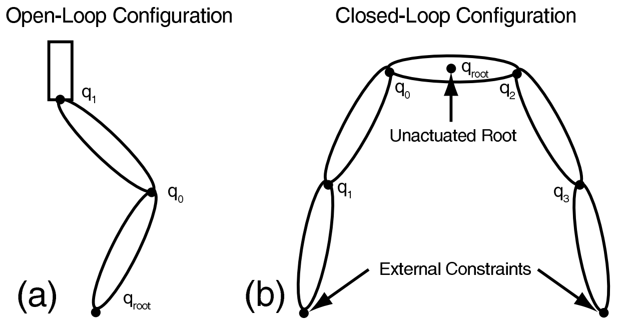

Figure 3: In the unconstrained, open-loop configuration (a) the shape is fully described by independent coordinates $\mathbf{q}$, whereas in the constrained, closed-loop configuration (b) no set of independent coordinates can describe the shape, so constraints must be handled in the dynamics.

evaluate and integrate these equations with one of several efficient algorithms [Featherstone and Orin 2000], but to animate active characters a control algorithm is still required to supply the joint torques τ needed to accomplish desired tasks.

3.1.1 Exact Linearization

Inverse dynamics simplifies design of control algorithms by compensating for complex nonlinear dynamics. The key idea is to transform the nonlinear equations of motion into a linear, second-order system. For example, by choosing joint torques of the form $\tau = \mathbf{M}\tau^* + \mathbf{h}$, the nonlinear Equation (1) is transformed into a set of linear, uncoupled second-order equations, $\ddot{\mathbf{q}} = \tau^*$. This transformation drastically simplifies systematic computation of command torques τ^* needed to accomplish joint-space tasks such as tracking procedurally generated trajectories [Ko and Badler 1996] or recorded motion data [Yin et al. 2003]. Manipulation tasks, however, are not easily described in joint space.

Alternatively, Cartesian coordinates, relative to the needed body part, can be used to intuitively describe manipulation tasks. It is possible to support such descriptions using inverse kinematics, but this approach ignores the dynamics of the task. Instead, our approach applies inverse dynamics in the Cartesian space to directly and intuitively control the task-space dynamics of manipulation tasks. We refer to this as task-space control. Given a differentiable expression $\mathbf{x}_1(\mathbf{q})$ for the position (or orientation) of some body part, we can compute its velocity $\dot{\mathbf{x}}_1 = \mathbf{J}_1\dot{\mathbf{q}}$ and its acceleration $\ddot{\mathbf{x}}_1 = \mathbf{J}_1\ddot{\mathbf{q}} + \dot{\mathbf{J}}_1\dot{\mathbf{q}}$ as a function of the Jacobian $\mathbf{J}_1 = \mathrm{D}_{\mathbf{q}}\mathbf{x}_1$. Combining the expression for task acceleration with Equation (1) allows us to express the dynamics in the Cartesian task space:

$$\mathbf{\Omega}_1\tau = \ddot{\mathbf{x}}_1 + \mathbf{\Omega}_1\mathbf{h} - \dot{\mathbf{J}}_1\dot{\mathbf{q}}, \tag{2}$$

where $\mathbf{\Omega}_1 = \mathbf{J}_1\mathbf{M}^{-1}$ can be thought of as the pseudoinverse of a task-space inertia matrix.[1]

As before, we compensate for nonlinearities by using inverse dynamics to transform task-space dynamics into a set of linear uncou-

[1] Standard robotics literature assumes a root joint placed at the base of a robot (i.e. foot of the character). By the virtual work principle, this placement implies a specific form for joint torques $\tau = \mathbf{J}_1^\top\mathbf{f}$ and a different pseudoinverse of the task-space inertia matrix $\mathbf{\Omega}_1 = \mathbf{J}_1\mathbf{M}^{-1}\mathbf{J}_1^\top$. We do not follow this convention here as it is simpler to implement animated characters with a root joint at the same location (i.e. the pelvis of the character), regardless of the supporting leg.

pled equations. Unlike the joint-space control, however, the systems of equations in task-space control is underdetermined requiring that we choose one of many possible torques. For example, the well known operational space formulation uses the pseudoinverse[2] that minimizes the instantaneous kinetic energy [Khatib 1987]. In contrast, our strategy does not depend upon a specific pseudoinverse but, instead, computes a complement joint torque $\bar{\tau}$ from a secondary posture task that incorporate motion data into the control of dynamic manipulations:

$$\tau = \mathbf{\Omega}_1^+(\mathbf{f}_1^* + \mathbf{\Omega}_1\mathbf{h} - \dot{\mathbf{J}}_1\dot{\mathbf{q}}) + \mathbf{P}_1\bar{\tau}, \tag{3}$$

where $\mathbf{\Omega}_1^+$ is any generalized pseudoinverse of $\mathbf{\Omega}_1$ and $\mathbf{P}_1 = (1 - \mathbf{\Omega}_1^+\mathbf{\Omega}_1)$ is the projection matrix onto the null space of $\mathbf{\Omega}_1$. Applying this joint torque to Equation (2), transforms the nonlinear task dynamics into a simple, second-order linear system, $\ddot{\mathbf{x}} = \mathbf{f}_1^*$, which eases description and control of manipulation tasks. The projection matrix ensures that the complement torque does not interfere with the primary manipulation task. Multi-task control, as described next, directs the remaining degrees of freedom to incorporate other tasks that control the posture of the character, for example.

3.1.2 Multi-Task Control

Multi-task control compensates for the nonlinear dynamics in both high priority and low priority tasks, allowing for precise and intuitive control of manipulations and the style with which they are performed. We again use inverse dynamics to linearize the dynamics of secondary tasks, but we cannot use Equations (1–3) because secondary tasks are affected by the joint torque $\tau_1 = \mathbf{\Omega}_1^+(\mathbf{f}_1^* + \mathbf{\Omega}_1\mathbf{h} - \dot{\mathbf{J}}_1\dot{\mathbf{q}})$ needed to accomplish the primary manipulation task and, also, by the projection matrix $\mathbf{P}_1$ that prevents secondary-task torque $\bar{\tau}$ from interfering with the higher priority tasks:

$$\tau_1 + \mathbf{P}_1\bar{\tau} = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}. \tag{4}$$

Depending on the type of secondary task, we can compensate for nonlinear dynamics by applying inverse dynamics in joint space or in task-space. If the task is to track joint values in the motion data, the joint torques are easiest to compute from command torque τ_2^* in joint coordinates:

$$\mathbf{P}_1\bar{\tau} = \mathbf{M}\tau_2^* + \mathbf{h} - \tau_1. \tag{5}$$

Whereas, if the task is more easily expressed in terms of Cartesian coordinates $\mathbf{x}_2(\mathbf{q})$, the joint torques are computed from the Cartesian command vector $\mathbf{f}_2^*$:

$$\mathbf{\Omega}_2\mathbf{P}_1\bar{\tau} = \mathbf{f}_2^* + \mathbf{\Omega}_2\mathbf{h} - \mathbf{\Omega}_2\tau_1 - \dot{\mathbf{J}}_2\dot{\mathbf{q}}, \tag{6}$$

where $\mathbf{J}_2 = \mathrm{D}_\mathbf{q}\mathbf{x}_2$ and $\mathbf{\Omega}_2 = \mathbf{J}_2\mathbf{M}^{-1}$, analogous to expressions in the primary-task control.

The derivation of both equations is analogous to the exact linearization of primary-task dynamics. It also clarifies that the joint-space control is a special case of task-space control, as seen by using the identity matrix for the task Jacobian in Equation (6). In both formulations, the singular projection matrix restricts the computed torque $\bar{\tau}$ to the set that does not interfere with the control of the primary task. In our implementation, we compute such torques with the singularity-robust pseudoinverse [Nakamura and Hanafusa 1986; Maciejewski 1990], which inverts the singular value decomposition of $\mathbf{P}_1$ (or $\mathbf{\Omega}_2\mathbf{P}_1$) after eliminating singular vectors with small singular values (e.g. less than 0.001 threshold in our implementation). This prevents large torques in singular directions that can result in an unstable simulation.

[2] $\mathbf{\Omega}_1^+ = \mathbf{J}_1\mathbf{M}^{-1}(\mathbf{J}_1\mathbf{M}^{-1}\mathbf{J}_1^\top)^{-1}$

Recursive application of the same idea extends this control algorithm to multiple tasks. For example, additional tasks might limit the range of joint variables [Liégeois 1977] or maintain balance [Zordan and Hodgins 2002]. Given a set of Cartesian coordinates $\{\mathbf{x}_1(\mathbf{q}), \ldots, \mathbf{x}_n(\mathbf{q})\}$ and a set of associated command vectors $\{\mathbf{f}_1^*, \ldots, \mathbf{f}_n^*\}$, the multi-task control computes the joint torque τ_i that executes the i-th task at a lower priority than the previous $(i - 1)$ tasks:

$$\tau_i = \tau_{i-1} + (\mathbf{\Omega}_i\mathbf{P}_{i-1})^+(\mathbf{f}_i^* + \mathbf{\Omega}_i\mathbf{h} - \mathbf{\Omega}_i\tau_{i-1} - \dot{\mathbf{J}}_i\dot{\mathbf{q}}),$$
$$\tau_1 = \mathbf{\Omega}_1^+(\mathbf{f}_1^* + \mathbf{\Omega}_1\mathbf{h} - \dot{\mathbf{J}}_1\dot{\mathbf{q}}),$$

where $\mathbf{P}_i = (1 - (\mathbf{\Omega}_i\mathbf{P}_{i-1})^+(\mathbf{\Omega}_i\mathbf{P}_{i-1}))$ and $\mathbf{P}_1 = (1 - \mathbf{\Omega}_1^+\mathbf{\Omega}_1)$. This iterative algorithm naturally resolves task conflicts by executing lower priority tasks with torques that do not interfere with the higher priority tasks.

Our formulation of multi-task control improves upon the recent generalization of operational space control in robotics literature [Khatib et al. 2004a; Sentis and Khatib 2004], which inspired much of our work. The fundamental difference is rooted in the formulation of dynamics for secondary tasks. In contrast to the original formulation, which requires differentiation of the quantity called the task-consistent posture Jacobian $\mathbf{J}_{2|1} = \mathbf{J}_2\mathbf{P}_1$, our formulation differentiates only the regular posture Jacobian $\mathbf{J}_2$, as seen by comparing the second term of Equation (20) in the original formulation [Khatib et al. 2004a] and the last term of Equation (6) in our formulation. This seemingly small difference has a profound impact on the ease of implementation and practical application of multi-task control to animation of dynamic manipulation. Unlike the expression $\dot{\mathbf{J}}_{2|1}\dot{\mathbf{q}}$ with the task-consistent posture Jacobian, our expression $\dot{\mathbf{J}}_2\dot{\mathbf{q}}$ can be computed simply and efficiently[3] without differentiating the complex expression for the projection matrix. The difference between the approaches is more pronounced for control of constrained dynamics because the analytic expression for the projection matrix, $\mathbf{P}_1$, becomes even more complex, as will be shown next.

3.2 Constrained Dynamics

Constrained dynamics emerge whenever a character applies more than a single limb to any one object. For example, standing with both feet on the ground or lifting an object with both hands establishes contact constraints that relate joint variables of one limb to those of the other. These dependencies make it impossible to describe characters with an independent set of joint variables, as was assumed throughout the previous subsection. Instead, we reformulate our control algorithm to use a set of *dependent* joint variables along with a set of constraint torques τ_c that enforce relationships imposed by contact constraints:

$$\tau + \tau_c = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}, \tag{7}$$

where all expressions retain the meaning from the standard formulation of unconstrained dynamics. The derivation of our control algorithm proceeds by computing the constraint torques prior to exact linearization of constrained dynamics.

[3] For example, the value of $\dot{\mathbf{J}}\dot{\mathbf{q}}$ can be computed with a small modification to the standard iterative formula for computing accelerations $\mathbf{a}_i$ of each link in a non-branching jointed structure. The iterative formula composes accelerations of each parent link $\mathbf{a}_{\lambda(i)}$ with the relative acceleration expressed in the motion subspace $\mathbf{s}_i$ of the joint: $\mathbf{a}_i = \mathbf{a}_{\lambda(i)} + \dot{\mathbf{s}}_i\dot{\mathbf{q}}_i + \mathbf{s}_i\ddot{\mathbf{q}}_i$ [Featherstone and Orin 2000]. Comparison with the alternative expression for acceleration $\mathbf{a} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$ reveals that by eliminating one term $\mathbf{s}_i\ddot{\mathbf{q}}_i$—the only term dependent on joint accelerations $\ddot{\mathbf{q}}_i$—from the iterative acceleration formula, we obtain an iterative formula for computing the value of $\dot{\mathbf{J}}\dot{\mathbf{q}}$.

The constraint torques are determined by a set of algebraic equations $\phi(\mathbf{q}) = 0$, which may, for example, model non-slipping contact by attaching limbs to objects in the environment. The entire set of constraints determines the structure of the constraint torques by prescribing the valid subspace $\tau_c = \mathbf{L}^\top \lambda$ as a function of the constraint Jacobian matrix $\mathbf{L} = \mathrm{D}_{\mathbf{q}}\phi$. This expression allows for computation of the constraint torques by solving for the coefficients λ in the subspace [Featherstone and Orin 2000]:

$$\mathbf{LM}^{-1}\mathbf{L}^\top \lambda = \mathbf{LM}^{-1}\mathbf{h} - \dot{\mathbf{L}}\dot{\mathbf{q}} - \mathbf{LM}^{-1}\tau. \qquad (8)$$

Given the expression for constraint torques, the derivation of our control algorithm proceeds as before by applying inverse dynamics to compensate for nonlinear dynamics in joint-space or task-space. For example, the control torques for the primary task $\mathbf{x}_1(\mathbf{q})$ are computed from the Cartesian command vector $\mathbf{f}_1^*$ using the following relationship:

$$\mathbf{\Omega}_1\mathbf{\Phi}\tau = \mathbf{f}_1^* + \mathbf{\Omega}_1\mathbf{h} + \mathbf{\Omega}_1\mathbf{\Gamma}(\dot{\mathbf{L}}\dot{\mathbf{q}} - \mathbf{LM}^{-1}\mathbf{h}) - \dot{\mathbf{J}}_1\dot{\mathbf{q}} \qquad (9)$$

where $\mathbf{\Gamma} = \mathbf{L}^\top(\mathbf{LM}^{-1}\mathbf{L}^\top)^{-1}$ and $\mathbf{\Phi} = (\mathbf{1} - \mathbf{\Gamma}\mathbf{LM}^{-1})$. This expression highlights the practical benefits of our control formulation (cf. Section 3.1.2). Instead of differentiating the new projection matrix $(1 - (\mathbf{\Omega}_1\mathbf{\Phi})^+(\mathbf{\Omega}_1\mathbf{\Phi}))$ as proposed in prior work [Khatib et al. 2004a; Sentis and Khatib 2004], our multi-task control is just as easily applied to both unconstrained and constrained dynamics.

3.3 Unactuated Joints

The joint structure of many animated characters includes passive, unactuated joints. The most common example is the six degree of freedom root joint that determines the global translation and orientation of the character. Unlike an active joint that propels limbs with its torques, the root joint does not apply torques or forces to propel the character directly: instead the global motion arises as a consequence of interaction with the ground and the environment.

We adjust our control algorithm by defining a selection matrix $\mathbf{S}$ that extracts actuated joints $\mathbf{q}_a$ from the full set of joint variables $\mathbf{q}_a = \mathbf{Sq}$. For example, the $(n-6) \times n$ matrix $S = [\mathbf{0} \mid \mathbf{1}_{n-6}]$ extracts all but the first six joint variables. Its transpose maps the joint torques into a vector that agrees with the dimension of joint variables, allowing us to rewrite constrained dynamics for characters with unactuated joints:

$$\mathbf{S}^\top\tau + \tau_c = \mathbf{Mq} + \mathbf{h}. \qquad (10)$$

The remaining steps in the derivation of our control algorithm are analogous to Section 3.2.

4 Task Description

Compact descriptions, which command only essential details such as hand position or applied force, accelerate animation of manipulation tasks and allow for easy, automated motion specification in interactive applications. Instead of setting and readjusting many keyframes, animators can describe just the required task, adjust a few intuitive parameters, and run a simulation to generate a new motion. Lifelike animations emerge automatically, much like in passive physical simulations, and adapt immediately to changes in the environment (e.g., different object motion or weight) or limitations of the character (e.g., locked joints or muscle strength).

Our control algorithm supports compact task descriptions by decoupling complex non-linear dynamics to allow for simplified motion commands in both joint-space and Cartesian task-space. As in keyframe animation systems, joint-space coordinates ease the description of tasks that require specific joint configurations such as poses from recorded motion data and Cartesian task-space coordinates allow for direct control of body parts needed to manipulate objects. The exact linearization of dynamics explained in the last section transforms the nonlinear problem into a simple second-order linear system. In this section we rely on this reduction to systematize descriptions of common manipulation tasks.

4.1 Manipulation

Our descriptions of manipulation tasks rely on two fundamental control primitives: stabilization, which directs characters towards prescribed values such as desired object locations; and tracking, which follows prescribed trajectories, such as those that describe the desired motion of manipulated objects. Both stabilization and tracking provide a way of choosing the command vector $\mathbf{f}^*$ (c.f. Section 3) that will accomplish various manipulation goals. Many other choices of the $\mathbf{f}^*$ are possible, but we have deliberately used simple choices to highlight the functionality of our control formulation, rather than confuse the details with complex motion planning strategies.

Since spatial configurations of manipulated objects are described relative to the global Cartesian coordinate frame, their manipulation is easiest to describe in Cartesian coordinates. We express manipulation tasks in Cartesian (or task-space) coordinates by using forward kinematics to compute the position (or orientation), $\mathbf{x}(\mathbf{q})$, of relevant body parts. If a character needs to reach for an object or to carry it to another location, we use stabilization to direct its hands to their desired location $\mathbf{x}_d$. Stabilization creates a motion that progressively eliminates the error between the current and desired configurations, $\mathbf{x}(\mathbf{q}) - \mathbf{x}_d$, by utilizing the command vector

$$\mathbf{f}^* = k(\mathbf{x}_d - \mathbf{x}(\mathbf{q})) - 2\sqrt{k}\dot{\mathbf{x}}(\mathbf{q}). \qquad (11)$$

Substituting this command vector into the second-order linear system, described in the last section, reveals a critically damped system whose speed of convergence is controlled by the gain coefficient k. Animators can increase the gain to create stiffer motions that accomplish tasks quickly or decrease it to create more relaxed motions. In our animations, we selected gains manually to showcase relaxed, more reactive animations, but in the future gains could also be set automatically according to measured human responses.

Tracking is used when more precise execution is required. For example, a character tossing an object must release the object at a prescribed location with a precise velocity. In such a case, we use tracking to direct the character's hands along the trajectory $\mathbf{x}_d(t)$ required to generate the required toss velocity. As in stabilization, tracking eliminates the error between the current and desired trajectories by computing the command force $\mathbf{f}^*$ needed for a critically damped system:

$$\mathbf{f}^* = k(\mathbf{x}_d(t) - \mathbf{x}(\mathbf{q})) + 2\sqrt{k}(\dot{\mathbf{x}}_d(t) - \dot{\mathbf{x}}(\mathbf{q})) + \ddot{\mathbf{x}}_d. \qquad (12)$$

4.2 Force Limits

Force limits restrict the magnitude of applied manipulation forces. This ensures that commands are not accomplished with unrealistic joint torques. For example, a heavy object is lifted slower than a light object because of the limits imposed on the application of

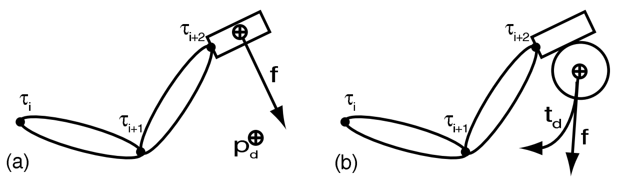

Figure 4: Task-space forces guide the hand toward desired position $\mathbf{p}_d$ using stabilization control (a), or move the hand along a specified trajectory $\mathbf{t}_d$, optionally grasping an object (b). For every force $\mathbf{f}$ in task-space, there is an equivalent force τ in joint-space that will cause the same motion of the hand and visa-versa.

the upward force. In nature, force limits are a function of muscle strength, but, in animation, force limits are more intuitively specified in the Cartesian task space. Our control algorithm can be extended to impose such limits by thresholding the task-space forces needed to perform each command.

Given a command vector $\mathbf{f}^*$, we can compute the required task-space force $\mathbf{f}_l$ using the expression for task-space dynamics in Equation (2):

$$\mathbf{f} = (\mathbf{J}_l\mathbf{M}\mathbf{J}_l^\top)^{-1}(\mathbf{f}^* + \mathbf{\Omega}_l\mathbf{h} - \dot{\mathbf{J}}_l\dot{\mathbf{q}}). \tag{13}$$

The task-space force $\mathbf{f}$ should be thought of as the external force that must act, in the absence of internal joint torques, to create the motion commanded by the vector $\mathbf{f}^*$ (Figure 4). The task-space force is measured in the usual units of force and its maximum magnitude can be adjusted intuitively to control the strength of manipulations. When the task-space force exceeds a preset value, its thresholded value $\hat{\mathbf{f}}$ can be used in place of the original command vector. If thresholding occurs, the Equation (13) is inverted to solve for the command vector $\hat{\mathbf{f}}^*$ that corresponds to the thresholded task-space force $\hat{\mathbf{f}}$. In our experiments, these modified command vectors generate animations with manipulation compromises similar to those observed in nature.

4.3 Posture

Most manipulation tasks can be accomplished in a number of ways, particularly by complex characters with many degrees of freedom. Although task descriptions command the motion of hands and other body parts, redundancies in body construction allow for variations that are evident in natural motion. The multi-level control formulation allows for systematic description of such variation with posture tasks. As a lower priority task, posture control parameterizes variations without interfering with higher priority manipulation tasks.

Variations depend on many factors including strength, personal preferences, and style. We model these variations by incorporating motion data into a posture task that favors recorded poses. This is implemented as a stabilization task in joint-space, where momentary goal configurations are computed with a nearest-neighbor search through a couple seconds of similar motion capture data. The similarity between poses is computed using the horizontal translation- and vertical rotation-invariant distance between synthetic markers affixed to each body part, as first proposed by Kovar and colleagues [Kovar et al. 2002].

Other descriptions of the posture task are also possible. They could be derived from physiological measurements of muscular effort [Khatib et al. 2004b; Sapio et al. 2005] or learned automatically from recorded motion data [Grochow et al. 2004; Mukai and Kuriyama 2005]. We deliberately choose a simple posture model instead of the more powerful and involved alternatives. In doing so we simplify evaluation of our control formulation: the lifelike motions seen in our examples are the result of proper control, not of a carefully learned model. Nevertheless, motion data is essential for resolving task redundancies and our work shows how to incorporate this knowledge into controls for physical simulation of manipulation tasks.

5 Results

The performance of our control algorithm was evaluated within the Open Dynamics Engine (*www.ode.org*), an open source, high performance library for simulating rigid multibody dynamics. In each experiment, a compact description commands the task for a complex character with 44 degrees of freedom. The control algorithm incorporates postures from supplied motion data to complete the missing details and directs the character in accomplishing each tasks. All collisions and contacts are detected and resolved in the simulation. In particular, we approximate grasping and ground contacts with clamping constraints that affix points on one body to the other. All simulations, including the control computation, run at interactive rates on a 2.8 GHz Pentium 4, with 60 or more updates per second, depending on the task complexity. All animations are included in the accompanying live video.

Chain Interaction. The chain interaction simulation is a simple demonstration of the immediate benefits gained by incorporating physical effects into animation of manipulation tasks. We command the character to steady its hands while it holds on to a serial linkage approximating a chain. The command uses the task-space stabilization to maintain a fixed hand motion as the other end of the chain is tugged and pulled by forces controlled interactively by a mouse-based interface. The secondary posture task keeps the character close to the initial posture. The strength with which the character resists the motion of the chain can be adjusted easily with control of the single gain parameter of the task-space stabilization command. Unlike with the kinematic methods, the character's entire body reacts to the motion of the chain. In particular, the motion of the legs, while subtle, contributes to a convincing portrayal of this manipulation task.

Box Interaction. The box interaction simulation demonstrates the immediate contribution of force limits to lifelike performances of manipulation tasks. The right hand is replaced with a heavy pendulum mass whose desired position is controlled interactively with a mouse-based interface. The dynamics of the pendulum mass are modeled as that of a body part connected to the arm with an unactuated joint. Stabilization control in task-space is used to bring the arm to the desired position. A secondary posture control references motion capture of a similar motion. This causes the character's posture to vary naturally with the action of the primary control task; the character crouches when the hand is low, stands when the hand is high, and appears balanced even though no explicit balance control is utilized. When the momentum of the pendulum is large, a force limit prevents the character from achieving the desired arm position. However, when the pendulum slows, the force required to achieve the desired position falls below the specified limit and the character can achieve the desired position flawlessly. Note that such precise control is not possible without accounting for the dynamics of the object in the exact linearization of the task-space dynamics. On the other hand, if realistic force limits are not imposed, the character will always achieve desired positions perfectly without realistically reacting to the momentum of the pendulum mass. Both force limits and correct dynamics are required to produce believable manipulation.

Lift. The box lifting simulation demonstrates our algorithm automatically adapting to the weight of objects and incorporating motion data. The high priority control changes in stages. While reaching for the box, stabilization control is used to bring the hands to their holding position. Once the hands are in place they are clamped to the box using the position constraint of the rigid body simulation. This forms another closed loop about the arms in addition to the one about the feet, both of which we explicitly account for with our control algorithm. The inertial properties of the box are know, but force limits prevent the character from lifting heavy boxes quickly or even at all. A secondary posture task favors postures from recorded motion data of a similar lifting motion. When we use different recorded data, the performance of the same task description adapts automatically. Instead of lifting with the back the character lifts the object with the knees. This confirms that our multi-task control decouples primary and secondary tasks and accomplishes each to the greatest extent possible.

Catch. In a ball catching simulation, the character catches balls of different weights, sizes and velocities. First, stabilization control is used to position the character's hand approximately where the ball should be caught. Then, when the ball is close to the hand, tracking control is used to match the hand velocity to that of the ball. If contact is detected, the ball is clamped to the hand with a simulation constraint. Finally, stabilization is used to bring the ball back to where the catch was made. The arm configuration varies naturally with the hand position because the posture task incorporates a short 10-second sequence of arm placement in various catch locations. As the weight of the ball increases, the character reacts naturally. Again, force limits prevent the use of extreme joint torques that might be capable of stabilizing the arm around the catch location regardless of the object weight. Instead, the arm motion slows down the ball before returning to its commanded location.

Catch and Toss. The catch and toss simulation demonstrate a performance of a more complex manipulation task. The character catches an object before tossing it along the prescribed trajectory. The simulation requires three inputs: the plane in which the character catches the object, the position and velocity at the point of release, and a motion capture sequence of a similar catch-and-throw motion. The commands in this animation are similar to those in the lifting and catching animations except for the trajectory tracking used to toss the object. The trajectory is a Hermite curve that is fully specified by the initial and final positions and velocities. The parameterization of the curve was choosen for simplicity and looks reasonable for this motion, but it should be noted that the realism of the resulting motion does depend upon the tracking trajectory and, thus, other choice would generate less believable motion. The controller is robust to changes in the velocity and angle of the caught object, the weight, size and shape of the object, and the specified direction and velocity that the object should be thrown. All reasonable settings of these parameters create a plausible motion with different, nonlinear effects. For instance, if the weight of the object is large, the character will not be able to control the object as accurately, causing collisions between the object and the character, but still tracking the trajectory as closely as possible.

Multi-Task Tracking. In a demonstration similar to the lifting example, we include a medium priority task (in-between the primary and posture tasks) that commands the pelvis center along a circular trajectory. The control algorithm tracks all three tasks while resolving their conflicts in a prioritized fashion. This demonstrates how our multi-task formulation accounts for the effects of higher-priority tasks to allow for precise control of lower-priority tasks, even with low tracking gains. Ultimately, this is what enables accurate tracking of recorded postures, which greatly contributes to the realism of our animations. Figure 5 compares our task-space control and a previous formulation for constrained dynamics [Sapio and Khatib 2005]. As the figure shows, our control algorithm converges quickly to desired trajectories even when the previous formulation diverges. These results agree with similar experiments conducted for unconstrained dynamics [Khatib et al. 2004a]. Both these experiments and ours confirm that accurate control of lower-priority tasks must account for the effects of higher-priority tasks. In some of our experiments, the error incurred otherwise lead to unstable behavior.

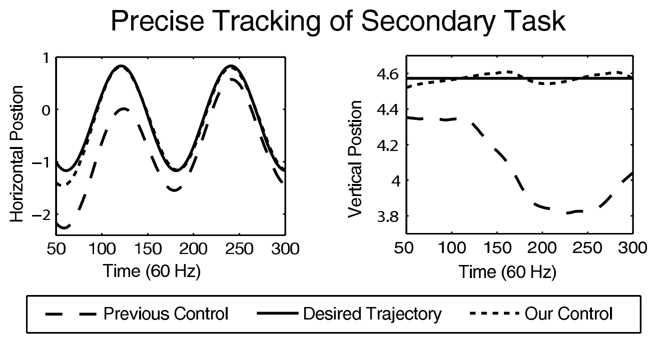

Figure 5: This graph demonstrates the capacity of our multi-task control to accomplish multiple tasks. Here we compare the tracking accuracy in the secondary task that commands a point on the pelvis along a circular trajectory. The control of the primary task, which steadies the object held by two hands, affects the tracking of this secondary tasks. If these effects are ignored, as in the previous task-space formulation for constrained dynamics [Sapio and Khatib 2005], the tracking suffers. Our formulation takes these effects into account to enable superior tracking.

6 Conclusion

Our control algorithm directs complex characters in realistic performances of dynamic manipulation tasks. The multi-task formulation supports intuitive task descriptions in joint space or task space. The tasks are executed at multiple priority levels to ensure that lower-priority tasks do not interfere with higher priority manipulation goals. The accurate tracking of lower-priority tasks capitalizes on recorded motion postures to generate lifelike motions from compact task descriptions with many missing details. The control adapts easily to dynamic disturbances and different environments that require significant deviation from motion data.

The control algorithm cannot guarantee successful performance of all manipulation tasks. Temporary underactuation (loss of control over some degrees of freedom) will impede manipulation even when it could be accomplished with the remaining degrees of freedom. For example, although a character could jump to reach an object, our control algorithm cannot look ahead to pre-plan the torques needed for such a jump. Although a general solution to underactuated control problems for complex characters is still an open problem, offline optimization has enjoyed some success particularly after simplifying equations of motion [Liu and Popović

2002; Safonova et al. 2004]. Underactuated control is less critical in authoring applications where animators could be relied upon to provide feasible task descriptions.

The choice of Cartesian (or task-space) control eases the description of many manipulation tasks but it also introduces the possibility of artificial algorithmic underactuation. Whenever a jointed structure approaches a singular configuration, the task-space control temporarily loses actuation over some degrees of freedom. This underactuation is artificial because it is strictly a function of the chosen joint-angle parameterization; it never appears in the joint space. In authoring applications, these situations could be avoided with intelligent task descriptions, but a more general solution would impose joint limits in the highest priority task to avoid kinematic singularities [Liégeois 1977]. In our work, the posture task servers as a partial substitute to joint limits by keeping the character out of unnatural configurations, but this approach would ultimately fail for extreme postures.

The control algorithm assumes that all contacts are maintained regardless of the applied joint torques. This control strategy is successful for the simulation of some tasks but the control algorithm will need to maintain these contacts explicitly before it can generate animations with realistic locomotion or balance. This extension will likely fit into our control formulation naturally because additional task commands can maintain contact constraints by ensuring that contact forces remain within the required friction cones [Murray et al. 1994].

The control of contact forces brings out the more general need to systematize task descriptions beyond the use of stabilization and tracking, the two command primitives we relied upon in all of our experiments. For example, in our throwing experiments, the hand motions were directed to follow prescribed trajectories even though natural throwing motions are rarely so precise. New commands should also support alternative, less-detailed task descriptions that incorporate motion data to fill in missing details automatically. Our use of recorded motion postures has only enticed a more systematic inclusion of general motion invariants. For example, low-priority posture tasks could incorporate recorded velocities, accelerations, and forces to generate even better performances of dynamic manipulation tasks.

References

Albrecht, I., Haber, J., and Seidel, H.-P. 2003. Construction and animation of anatomically based human hand models. In *Symposium on Computer Animation (SCA)*, 98–109.

Bruderlin, A., and Williams, L. 1995. Motion signal processing. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, ACM SIGGRAPH, Annual Conference Series, 97–104.

Choi, K.-J., and Ko, H.-S. 2000. Online motion retargetting. *Journal of Visualization and Computer Animation 11*, 5 (Dec.), 223–235.

Duda, R. O., Hart, P. E., and Stork, D. G. 2000. *Pattern Classification*, 2nd ed. John Wiley & Sons, Inc., New York.

ElKoura, G., and Singh, K. 2003. Handrix: animating the human hand. In *Symposium on Computer Animation (SCA)*, 110–119.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, 251–260.

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

Gleicher, M. 1997. Motion editing with spacetime constraints. In *1997 Symposium on Interactive 3D Graphics*, 139–148.

Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (Aug.), 522–531.

Grzeszczuk, R., and Terzopoulos, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 95*, Annual Conference Series, 63–70.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. 1995. Animating human athletics. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, 71–78.

Khatib, O., Sentis, L., Park, J.-H., and Warren, J. 2004. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics 1*, 1, 29–43.

Khatib, O., Warren, J., Sapio, V. D., and Sentis, L. 2004. *Human-Like Motion From Physiologically-Based Potential Energies*, vol. XII of *On Advances in Robot Kinematics*. Springer, New York, ch. Humanoids and Biomedical Applications.

Khatib, O. 1987. A unified approach to motion and force control of robot manipulators: the operational space formulation. *International Journal of Robotics Research 3*, 1, 43–53.

Ko, H.-S., and Badler, N. I. 1996. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications 16*, 2, 50–59.

Koga, Y., Kondo, K., Kuffner, J., and Latombe, J.-C. 1994. Planning motions with intentions. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, 395–408.

Kovar, L., and Gleicher, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3 (Aug.), 559–568. In Press.

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. *ACM Transactions on Graphics 21*, 3 (July), 473–482.

Laszlo, J. F., van de Panne, M., and Fiume, E. L. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 155–162.

Lee, P., Wei, S., Zhao, J., and Badler, N. I. 1990. Strength guided motion. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, vol. 24, 253–262.

Li, Y., and Pollard, N. S. 2005. A shape matching algorithm for synthesizing humanlike enveloping grasps. In *IEEE/RAS International Conference on Humanoid Robots*, 442–449.

Liégeois, A. 1977. Automatic supervisor control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics 7*, 12, 868–871.

Liu, C. K., and Popović, Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (July), 408–416.

Maciejewski, A. A. 1990. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications 10*, 3, 63–71.

Mukai, T., and Kuriyama, S. 2005. Geostatistical motion interpolation. *ACM Transactions on Graphics 24*, 3 (Aug.), 1062–1070.

Murray, R. M., Li, Z., and Sastry, S. S. 1994. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Boca Raton.

Nakamura, Y., and Hanafusa, H. 1986. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control 108*, 163–171.

Pollard, N. S., and Zordan, V. B. 2005. Physically based grasping control from example. In *Symposium on Computer Animation (SCA)*, 311–318.

Popović, Z., and Witkin, A. P. 1999. Physically based motion transformation. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, ACM SIGGRAPH, Annual Conference Series, 11–20.

Raibert, M. H., and Hodgins, J. K. 1991. Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, ACM SIGGRAPH, Annual Conference Series, 349–358.

Rose, C., Cohen, M. F., and Bodenheimer, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

Rose, C. F., Sloan, P.-P. J., and Cohen, M. F. 2001. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum 20*, 3, 239–250.

Safonova, A., Hodgins, J., and Pollard, N. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (Aug.), 514–521.

Sapio, V. D., and Khatib, O. 2005. Operational space control of multibody systems with explicit holonomic constraints. In *International Conference on Robotics and Automation (ICRA)*, 2961–2967.

Sapio, V. D., Warren, J., Khatib, O., and Delp, S. 2005. Simulating the task-level control of human motion: a methodology and framework for implementation. *The Visual Computer 21*, 5, 289–302.

Sentis, L., and Khatib, O. 2004. Prioritized multi-objective dynamics and control of robots in human environments. In *IEEE/RAS International Conference on Humanoid Robots*, vol. 2, 764–780.

Sulejmanpasić, A., and Popović, J. 2005. Adaptation of performed ballistic motion. *ACM Transactions on Graphics 24*, 1 (Jan.), 165–179.

van de Panne, M., Fiume, E., and Vranesic, Z. 1990. Reusable motion synthesis using state-space controllers. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, ACM SIGGRAPH, Annual Conference Series, 225–234.

Witkin, A., and Popović, Z. 1995. Motion warping. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, ACM SIGGRAPH, Annual Conference Series, 105–108.

Yamane, K., Kuffner, J. J., and Hodgins, J. K. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics 23*, 3 (Aug.), 532–539.

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

Zordan, V. B., and Hodgins, J. K. 2002. Motion capture-driven simulations that hit and react. In *Symposium on Computer Animation (SCA)*, 89–96.