Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Interactive Animation of Dynamic Manipulation

Yeuhi Abe and Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

Lifelike animation of object manipulation requires dynamic interaction between animated characters, objects, and their environment. These interactions can be animated automatically with physically based simulations but proper controls are needed to animate characters that move realistically and that accomplish tasks in spite of unexpected disturbances. This paper describes an efficient control algorithm that generates realistic animations by incorporating motion data into task execution. The end result is a versatile system for interactive animation of dynamic manipulation tasks such as lifting, catching, and throwing.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three Dimensional Graphics and RealismAnimation

1. Introduction

Animation of object manipulation involves complex physical interactions between characters, objects, and their environment. For example, a character holding one end of a rope must counteract forces applied at the other end by steadying its hands. These animations cannot be generated automatically using kinematic techniques because kinematics ignores dynamic interaction. However, physical simulation can generate these animations automatically as long as the character is controlled properly to accomplish the required manipulation.

Control should be derived from intuitive descriptions of manipulations tasks, which are often underspecified because most tasks can be accomplished in several ways. Holding a rope, for example, characterizes the motion of the hands but does not prescribe the motion for the rest of the body. Given an incomplete description, the control algorithm must accomplish the stated goals as well as complete the missing details to generate realistic animations.

Our control algorithm incorporates high-quality motion data to guide complex characters, with many degrees of freedom, through lifelike portrayals of common manipulation tasks. The algorithm, illustrated in Figure 1, complements intuitive descriptions of multiple manipulation tasks with recorded motion data to compute the joint torques required to manipulate objects within interactive physical simulation. The task descriptions are provided by the animator or a high-level state machine while the recorded motions are selected to include a few examples of preferred movement postures.

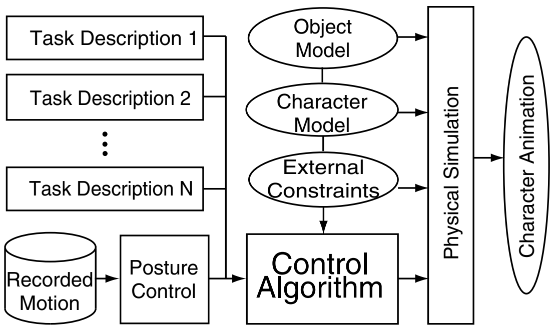

Figure 1: *Our control algorithm incorporates recorded motion data to accomplish multiple taks such as lifting, reaching, and throwing within interactive physical simulations.*

The key to our control algorithm and the primary contribution of this paper is a new formulation that accurately tracks lower priority movement postures without interfering with the higher priority manipulation tasks. This accuracy results in high-quality animations because the control incorporates recorded motion postures without compromising manipulation tasks. A unique feature of this approach is that it benefits from just a few motions even when new animations differ significantly from the recorded motion data. For example, a single motion of a person lifting a light object can be used to animate many lifts regardless of the object weight. In all cases, realistic timing emerges naturally as a consequence of task descriptions that limit the forces applied by the hands. Unlike kinematic control techniques, our control algorithm operates within a physical simulation allowing for dynamic interaction between characters and other simulated objects.

2. Background

Animations of dynamic manipulation must account for both the dynamics and the kinematics of tasks because static considerations alone will not generate lifelike motion [LWZB90]. 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 [RCB98, MK05, KG04]. Although this is effective when tasks can be restricted to small, well sampled manipulations, more general tasks require solutions to increasingly difficult or ill-posed machine-learning problems. To extend the range of a limited data set, current interactive applications rely on motion-editing tools that approximate dynamics with temporal smoothness [BW95, WP95, Gle97, CK00] because dynamically consistent editing tools have not been designed for interactive use [PW99,LP02,SP05]. In contrast, our work directly accounts for dynamics by controlling character motion within a physical simulation.

Preplanned motions can be executed in simulation using joint-space PD control, which tracks joint trajectories [ZH02, YCP03]. Joint-space control has also been successful in animation of lifelike locomotion and other activities [vdPFV90,RH91,HWBO95,GT95,LvdPF96,FvdPT01]. However, joint-space control techniques do not allow for precise control of the motion or forces applied to manipulated objects.Our control algorithm eases the animation of dynamic manipulation by explicitly accounting for object dynamics and supporting intuitive descriptions of motion and force limits directly in the Cartesian space of the objects being manipulated. We call this Cartesian-space control.

Cartesian-space control of manipulated objects allows for compact task description because it commands only the precise details of object manipulation. In animation, compact task descriptions are generally preferred in both manual [LWZB90] and automatic [KKKL94] task planning because they suppress irrelevant aspects of task execution. For example, inverse kinematics is often used, to infer full postures from a compact description of the motion of hands, making it easier to reuse performances by different (e.g., shorter or longer-armed) characters [YKH04]. Achieving lifelike postures, however, requires that such algorithms either incorporate recorded motion data or leverage prior results from neurophysiology or other studies of natural motion [KKKL94, RSC01, GMHP04, YKH04]. Our work addresses a similar problem but, unlike inverse kinematics, it incorporates motion data and dynamics to control characters in simulations with significant dynamics.

A popular approach to Cartesian space control is known as operational space control in the robotics literature [Kha87]. Similar to our approach, the operational space formulation simplifies control of complex humanoid robots with many degrees of freedom by decoupling the control needed to accomplish a task from the control of task-redundant degrees of freedom. Recently, the original operational space formulation was improved upon by Khatib and colleagues to enable accurate tracking of lower-priority tasks [KSPW04], but only for branching joint structures without closed-loop joint constraints [DSK05]. Our work offers an alternative to this approach that is more suitable to character animation. It enables accurate tracking of recorded motion data even with closed-loop joint structures. This is particularly important because closed-loop constraints emerge whenever a character places both feet on the ground, allowing our formulation to track motion data in these common cases.

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 2). These independent coordinates $\mathbf{q}$ allow for the dynamics of the character to be expressed in

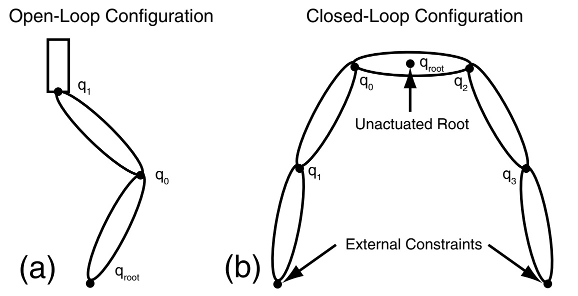

Figure 2: *In the unconstrained, open-loop configuration (a) the shape is fully described by independent coordinates* **q**, *whereas in the constrained, closed-loop configuration (b) no set of independent coordinates can describe the shape, so constraints must be handled in the dynamics.*

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 [FO00]. Physical simulations can evaluate and integrate these equations with one of several efficient algorithms, 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 [KB96] or recorded motion data [YCP03]. Manipulation tasks, however, are not easily described in joint space.

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.

As before, we compensate for nonlinearities by using inverse dynamics to transform task-space dynamics into a set of linear uncoupled 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 that minimizes the instantaneous kinetic energy [Kha87]. In contrast, our formulation will compute the complement joint torque $\bar{\tau}$ to incorporate motion data into 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 $\mathbf{J}_2 = D_\mathbf{q}\mathbf{x}_2$ and $\boldsymbol{\Omega}_2 = \mathbf{J}_2\mathbf{M}^{-1}$, 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 [NH86, Mac90], which inverts the singular value decomposition of $\boldsymbol{\Omega}_1$ (or $\boldsymbol{\Omega}_2\mathbf{P}_1$) 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.

Recursive application of the same idea extends this control algorithm to multiple tasks. For example, additional tasks might limit the range of joint variables [Lié77] or maintain balance [ZH02]. 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:

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

where $\mathbf{P}_i = (1 - (\boldsymbol{\Omega}_i\mathbf{P}_{i-1})^+(\boldsymbol{\Omega}_i\mathbf{P}_{i-1}))$ and $\mathbf{P}_1 = (1 - \boldsymbol{\Omega}_1^+\boldsymbol{\Omega}_1)$. 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 offers an alternative to the formulation proposed in the robotics literature [KSPW04, SK05]. The two approaches differ in the formulation of secondary-task dynamics in Eq. (6). Unlike the robotics formulation, which requires differentiating the quantity called the task-consistent posture Jacobian $\mathbf{J}_{2|1} = \mathbf{J}_2\mathbf{P}_1$, our approach differentiates only the regular posture Jacobian $\mathbf{J}_2$, as seen in the last term of Eq. (6). This 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 without differentiating the complex projection matrix $\mathbf{P}_1$. Furthermore, it can be shown that both formulations do not interfere with high-priority tasks even as they track secondary tasks as accurately as possible. The difference between the two approaches becomes more pronounced in control of constrained dynamics because the analytic expression for the projection matrix, $\mathbf{P}_1$, becomes more complex, making it harder to compute the time derivative $\dot{\mathbf{J}}_{2|1}$, while our formulation eliminates this step entirely.

3.2. Constrained Dynamics

Constrained dynamics emerge whenever a character applies more than one limb to a fixed object in the environment. For example, standing with both feet on the ground 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.

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 $\mathbf{L} = D_\mathbf{q}\phi$. This expression allows for computation of the constraint torques by solving for the coefficients λ in the subspace [FO00]:

$$\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^\top\lambda = \mathbf{L}\mathbf{M}^{-1}\mathbf{h} - \dot{\mathbf{L}}\dot{\mathbf{q}} - \mathbf{L}\mathbf{M}^{-1}\tau. \tag{8}$$

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:

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

where $\boldsymbol{\Gamma} = \mathbf{L}^\top(\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^\top)^{-1}$ and $\boldsymbol{\Phi} = (\mathbf{1} - \boldsymbol{\Gamma}\mathbf{L}\mathbf{M}^{-1})$. This expression highlights the practical benefits of our control formulation (cf. Section 3.1.2). Instead of differentiating the new projection matrix $(1 - (\boldsymbol{\Omega}_1\boldsymbol{\Phi})^+(\boldsymbol{\Omega}_1\boldsymbol{\Phi}))$ as proposed in prior work [KSPW04, SK05], 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 $\mathbf{q}_a = \mathbf{S}\mathbf{q}$. 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:

$$\mathbf{S}^\top \tau + \tau_c = \mathbf{M}\mathbf{q} + \mathbf{h}. \tag{10}$$

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

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

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:

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

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 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 $\mathbf{f}$ using the expression for task-space dynamics in Equation (2):

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

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 3). The task-space force is measured in the usual units of force

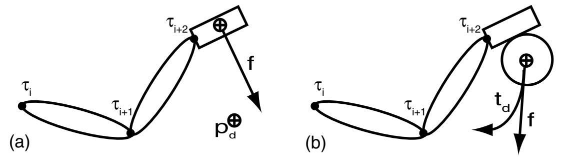

Figure 3: *Task-space forces guide the hand toward desired position $\mathbf{p}_d$ using stabilization control (a), or move the hand along a specified trajectory $\mathbf{t}_d$, optionally grasping an object (b). For every force $\mathbf{f}$ in task-space, there is an equivalent force τ in joint-space that will cause the same motion of the hand and visa-versa.*

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 $\hat{\mathbf{f}}$.

The method we have proposed so far only accounts for force limits in the Cartesian space of the primary task. But in nature force limits are a byproduct of limited muscle strength. Thus, more accurate models should limit forces in the joint-space of characters. Despite this fact, the method we propose has two advantages. First, the animation process is greatly simplified by allowing Cartesian space force limits; It is more intuitive to describe a character's strength by how much the character can lift than by the maximum torque each joint can exhert. Second, it is unclear how the motion of the primary task should gracefully degrade when force limits in joint space are reached. Simply clamping the torques will produce unstable motion. Our method always provides modified command vectors that produce manipulation compromise 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 few 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 [KGP02].

Other descriptions of the posture task are also possible. They could be derived from physiological measurements of muscular effort [KWDSS04, DSWKD05] or learned automatically from recorded motion data [GMHP04, MK05]. Our posture task is a simple variant of the latter choice, aiming to ease evaluation of our control technique rather than to improve upon existing posture models.

It should be noted that for realistic motions, posture activity cannot be treated completely independent of the primary task. For example, when lifting a heavy box, a person might choose to do so "with the knees" rather than "with the back" to reduce strain on the muscles. Despite this fact, decoupled motion control has proven a useful abstraction in animation, as demonstrated by the prevailance of inverse kinematic techniques for motion synthesis. As with inverse kinematics, our method depends upon intelligent choices for the posture that compliment the primary task. We leave to future work the development of more sophisticated posture tasks that actively adapt to the goals of the primary task.

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. Collisions and contacts are detected and resolved in the simulation. In particular, grasping and ground contacts are approximated 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. In this simulation the character attempts to steady its hands while holding onto a serial linkage approximating a chain. Task-space stabilization (c.f. Section 4.1) is used 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. Unlike with kinematic techniques, the character 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.

Lift. The box lifting simulation demonstrates our algorithm automatically adapting to the weight of objects and incorporating motion data (see Figure 4). Stabilization control is used to direct the motion of the hands by specifying keyframes that the hands should pass through. The hands are clamped to the box using simulation constraints between the rigid bodies. Although the control is aware of the box mass (and takes it into account), 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.

Box Interaction. The box interaction simulation demonstrates the necessity of dynamic interaction between the character and manipulated objects. The right hand of the character is replaced with a heavy pendulum mass and the desired position of the hand 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 manipulation control. But if, in addition, realistic force limits are not imposed, the character will always achieve the desired hand position perfectly without realistically reacting to the momentum of the pendulum mass. Both force limits and correct dynamics are required to produce believable manipulation.

Catch. In the ball catching simulation, the character catches balls of different weights, sizes and velocities. Stabilization control is used to position the character's hand approximately where the ball should be caught. 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 too quickly stabilizing the position of the hand, 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 attempts to catch 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. This 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 dynamic 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.

6. Conclusion

Our control algorithm directs complex characters in realistic performances of dynamic manipulation tasks within a physical simulation. The control adapts easily to dynamic disturbances and different environments that require significant deviation from motion data. 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. Finally, 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 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 the space of motions [LP02, SHP04]. Underactuated control is less critical in authoring applications where animators could be relied upon to provide feasible task descriptions.

The choice of Cartesian-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 [Lié77]. In our work, the posture task serves 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 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 [MLS94].

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. Stabilization and tracking control in Cartesian space simplifies motion specification, but it does not guarantee realism. 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.

7. Acknowledgments

The authors would like to thank Eugene Hsu and Daniel Vlasic for motion capture and multimedia assistance, respectively. Acknowledgments are also due to the following people for their valuable comments, insights and proofreads: Jiawen Chen, Marco Da Silva, Frédo Durand, Russ Tedrake, Robert Wang and the anonymous reviewers.

References

[BW95] Bruderlin A., Williams L.: Motion signal processing. In *Computer Graphics (Proceedings of SIGGRAPH 95)* (Aug. 1995), Annual Conference Series, ACM SIGGRAPH, pp. 97–104. 2

[CK00] Choi K.-J., Ko H.-S.: Online motion retargetting. *Journal of Visualization and Computer Animation 11*, 5 (Dec. 2000), 223–235. 2

[DSK05] De Sapio V., Khatib O.: Operational space control of multibody systems with explicit holonomic constraints. In *International Conference on Robotics and Automation (ICRA)* (2005), pp. 2961–2967. 2

[DSWKD05] De Sapio V., Warren J., Khatib O., Delp S.: Simulating the task-level control of human motion: a methodology and framework for implementation. *The Visual Computer 21*, 5 (2005), 289–302. 6

[FO00] Featherstone R., Orin D. E.: Robot dynamics: Equations and algorithms. In *International Conference on Robotics and Automation (ICRA)* (2000), pp. 826–834. 3, 4

[FvdPT01] Faloutsos P., van de Panne M., Terzopoulos D.: Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Annual Conference Series, pp. 251–260. 2

[Gle97] Gleicher M.: Motion editing with spacetime constraints. In *1997 Symposium on Interactive 3D Graphics* (Apr. 1997), pp. 139–148. 2

[GMHP04] Grochow K., Martin S. L., Hertzmann A., Popović Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 522–531. 2, 6

[GT95] Grzeszczuk R., Terzopoulos D.: Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 95* (Aug. 1995), Annual Conference Series, pp. 63–70. 2

[HWBO95] Hodgins J. K., Wooten W. L., Brogan D. C., O'Brien J. F.: Animating human athletics. In *Proceedings of ACM SIGGRAPH 95* (Aug. 1995), Annual Conference Series, pp. 71–78. 2

[KB96] Ko H.-S., Badler N. I.: Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications 16*, 2 (1996), 50–59. 3

[KG04] Kovar L., Gleicher M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 559–568. In Press. 2

[KGP02] Kovar L., Gleicher M., Pighin F.: Motion graphs. *ACM Transactions on Graphics 21*, 3 (July 2002), 473–482. 6

[Kha87] Khatib O.: A unified approach to motion and force control of robot manipulators: the operational space formulation. *International Journal of Robotics Research 3*, 1 (1987), 43–53. 2, 3

[KKKL94] Koga Y., Kondo K., Kuffner J., Latombe J.-C.: Planning motions with intentions. In *Proceedings of SIGGRAPH 94* (July 1994), Computer Graphics Proceedings, Annual Conference Series, pp. 395–408. 2

[KSPW04] Khatib O., Sentis L., Park J.-H., Warren J.: Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics 1*, 1 (2004), 29–43. 2, 4

[KWDSS04] Khatib O., Warren J., De Sapio V., Sentis L.: *Human-Like Motion From Physiologically-Based Potential Energies*, vol. XII of *On Advances in Robot Kinematics*. Springer, New York, 2004, ch. Humanoids and Biomedical Applications. 6

[Lié77] Liégeois A.: Automatic supervisor control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics 7*, 12 (1977), 868–871. 4, 8

[LP02] Liu C. K., Popović Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (July 2002), 408–416. 2, 8

[LvdPF96] Laszlo J. F., van de Panne M., Fiume E. L.: Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Annual Conference Series, pp. 155–162. 2

[LWZB90] Lee P., Wei S., Zhao J., Badler N. I.: Strength guided motion. In *Computer Graphics (Proceedings of SIGGRAPH 90)* (Aug. 1990), vol. 24, pp. 253–262. 2

[Mac90] Maciejewski A. A.: Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications 10*, 3 (1990), 63–71. 4

[MK05] Mukai T., Kuriyama S.: Geostatistical motion interpolation. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 1062–1070. 2, 6

[MLS94] Murray R. M., Li Z., Sastry S. S.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, 1994. 8

[NH86] Nakamura Y., Hanafusa H.: Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control 108* (1986), 163–171. 4

[PW99] Popović Z., Witkin A. P.: Physically based motion transformation. In *Computer Graphics (Proceedings of SIGGRAPH 99)* (Aug. 1999), Annual Conference Series, ACM SIGGRAPH, pp. 11–20. 2

[RCB98] Rose C., Cohen M. F., Bodenheimer B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40. 2

[RH91] Raibert M. H., Hodgins J. K.: Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings of SIGGRAPH 91)* (July 1991), Annual Conference Series, ACM SIGGRAPH, pp. 349–358. 2

[RSC01] Rose C. F., Sloan P.-P. J., Cohen M. F.: Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum 20*, 3 (2001), 239–250. 2

[SHP04] Safonova A., Hodgins J., Pollard N.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 514–521. 8

[SK05] Sentis L., Khatib O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics 2*, 4 (2005), 505–518. 4

[SP05] Sulejmanpasić A., Popović J.: Adaptation of performed ballistic motion. *ACM Transactions on Graphics 24*, 1 (Jan. 2005), 165–179. 2

[vdPFV90] van de Panne M., Fiume E., Vranesic Z.: Reusable motion synthesis using state-space controllers. In *Computer Graphics (Proceedings of SIGGRAPH 90)* (Aug. 1990), Annual Conference Series, ACM SIGGRAPH, pp. 225–234. 2

[WP95] Witkin A., Popović Z.: Motion warping. In *Computer Graphics (Proceedings of SIGGRAPH 95)* (Aug. 1995), Annual Conference Series, ACM SIGGRAPH, pp. 105–108. 2

[YCP03] Yin K., Cline M., Pai D. K.: Motion perturbation based on simple neuromotor control models. In *Pacific Conference on Computer Graphics and Applications (PG)* (2003), pp. 445–449. 2, 3

[YKH04] Yamane K., Kuffner J. J., Hodgins J. K.: Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 532–539. 2

[ZH02] Zordan V. B., Hodgins J. K.: Motion capture-driven simulations that hit and react. In *Symposium on Computer Animation (SCA)* (July 2002), pp. 89–96. 2, 4

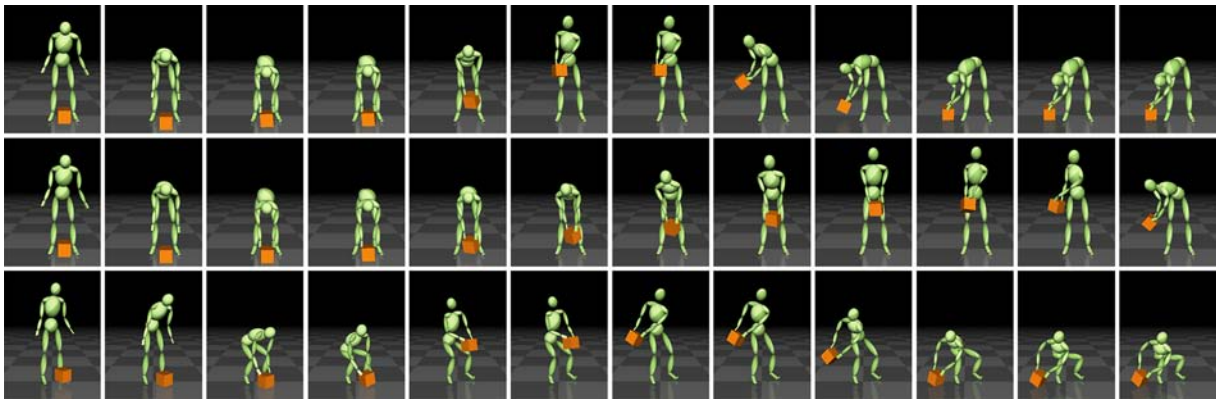

Figure 4: *Our control algorithm directs a real-time simulation of a character to accomplish manipulations, such as displacing a box (top row). Manipulations are compactly described. In the above example, only four Cartesian goal positions are used to describe the motion of the hands and the box. The 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).*