Christoph W. Borst*
Arun P. Indugula
Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette LA 70504-4330

*Correspondence to
cborst@cacs.louisiana.edu

A Spring Model for Whole-Hand Virtual Grasping

Abstract

We present a physically-based approach to grasping and manipulation of virtual objects that produces visually realistic results, addresses the problem of visual interpenetration of hand and object models, and performs force rendering for force-feedback gloves, in a single framework. Our approach couples a simulation-controlled articulated hand model to tracked hand configuration using a system of linear and torsional virtual spring-dampers. We discuss an implementation of our approach that uses a widely available simulation tool for collision detection and response. We pictorially illustrate the resulting behavior of the virtual hand model and of grasped objects, discuss user behavior and difficulties encountered, and show that the simulation rate is sufficient for control of current force-feedback glove designs. We also present a prototype system for natural whole-hand interactions in a desktop-sized workspace.

1 Introduction

1.1 The Grasping Problem

Various interaction techniques are useful in virtual environments. Some of the basic tasks that need to be supported are navigation of an environment, selection of objects or options, exploration of objects, and manipulation of object pose or other properties. In some applications, these interactions are adequately supported by specialized devices and techniques that do not necessarily duplicate real-world interactions, for example, see the summary by Bowman, Kruijff, LaViola, and Poupyrev (2001). However, in other applications, it is important to duplicate real-world behavior. For example, a training system for a manual assembly task benefits from a realistic system for object grasping and manipulation. A variety of sensing gloves and other hand-tracking technologies has been developed to provide the whole-hand input needed for such applications, but the quality of simulated grasping and manipulation has been limited.

A main source of difficulty in virtual grasping is that a real hand cannot be expected to obey the laws of physics with respect to a simulated environment. That is, a virtual environment cannot physically constrain real hand motion like a real environment would. This can result in unrealistic hand motion accompanied by visually distracting artifacts such as a hand model sinking into a virtual object and a virtual grasp appearing frictionless. Visual feedback can be improved by keeping the virtual hand model outside of the object, but then the visual feedback no longer accurately represents the real hand configuration. A related problem is that physical sensations of friction and weight are important

Presence, *Vol. 15, No. 1, February 2006,* 47–61

for human grasp control (Johansson, 1998) but are not provided by virtual objects, further limiting the realism of a user's grasping motions. Finally, limited accuracy of whole-hand input devices and of geometric hand models also limits achievable interaction quality.

Haptic feedback devices alleviate the first two problems somewhat, but do not solve them. Glove-based force feedback devices have been shown to improve grasping control (Fabiani & Burdea, 1996), but they have severely limited degrees of freedom (typically one normal force per fingertip) and only a limited object stiffness can be simulated by their actuators due to mechanical and control characteristics. We therefore do not expect glove-based devices to provide accurate motion constraints or sensations in the near future.

1.2 Our Work

In this paper, we present a system for whole-hand grasping and manipulation using dynamic simulation. We prefer the generality of a physically-based approach over approaches that depend on the heuristic analysis of grasp stability or user intent, since these tend not to solve the problem in a general way and only handle selected cases. A physically-based approach can also take advantage of forthcoming improvements in physical simulation tools, which will in turn improve the quality of the results, while improving a heuristic approach would require development of more complex heuristics or further work on the basic grasping mechanism itself. A grasping system that is integrated with physical simulation tools will also naturally extend from rigid body interactions to interactions with deformable bodies.

We also sought to develop an approach that would reduce distracting visual artifacts. Due to the phenomenon of visual capture (Welch & Warren, 1980) and the high sensitivity of users to visual interpenetration artifacts (Burns et al., 2005), we expect visual feedback that is not an exact representation of real hand configuration to be less distracting to users than an exact representation that suffers from the visual hand-object interpenetration artifact.

Finally, we wanted the grasping system to perform force rendering for force-feedback gloves. Our grasping approach includes force rendering for devices such as the Immersion CyberGrasp and the Rutgers Master, but it can also be used without the force-feedback component.

In the next section, we summarize previous work on virtual grasping and on haptic rendering for glove-type devices. In Section 3, we present a system we are developing for natural interactions in a desktop-sized environment. Section 4 describes our physically-based grasping approach. Section 5 discusses our experiences with the approach and evaluates simulation speed.

The main contributions described in this paper are:

- We present a spring model that, in a single framework, supports virtual grasping and manipulation, addresses the problem of visual artifacts described above, and provides a new force rendering approach for force-feedback gloves.
- We evaluate an implementation of our spring model using a widely-available simulation tool for collision detection and response. Specifically, we illustrate the resulting grasp behavior, discuss user behavior and difficulties encountered, and analyze simulation rate.

2 Previous Work

The previous section outlined our design philosophy and motivated the use of a physically-based approach rather than a heuristic approach. In this section, we discuss previous related work. Note that we do not discuss grasping approaches for autonomous control of robots or virtual humans because these focus on the different problem of grasp *planning*. Concepts such as force closure and form closure from robotics can be useful for heuristic approaches that need to identify when grasping has occurred, and planning concepts can be useful for systems such as the grasping automata described by Boulic, Rezzonico, and Thalmann (1996), but our approach falls in neither of these categories.

2.1 Point-Based Grasping Approaches

Early grasping systems typically fixed an object to the hand's coordinate system once a simple test de-

tected a grasp. For example, Iwata's point-based test (Iwata, 1990) checked 16 points on a hand model for contact with a virtual object and considered the object to be captured when it was simultaneously contacted by points on both the thumb and any other finger. Bergamasco, Degl'Innocenti, and Bucciarelli (1994) introduced physically-based object response with a point-based force calculation. Control point grids placed on palmar sides of virtual hand segments were used to compute a force vector that included static friction, dynamic friction, normal contact forces, and external forces. Recently, Hirota and Hirose (2003) developed a manipulation system using point-based collision response forces from a much larger number of points on a hand model. Collision detection and response were based on the motion of interface points into a tetrahedral mesh model of a manipulated object. A video accompanying their paper demonstrated realistic object motion in response to various manipulations.

One problem with point-based approaches is that collision detection is not accurate unless a very large number of points is used to densely cover the hand. Another problem is that force applied to the manipulated object can be underestimated for sharp or wedge-shaped objects, as pointed out by Hirota and Hirose. We have observed a similar problem in the use of grids of points for force rendering for force-feedback gloves (Borst & Volz, 2003). Finally, the point-based approaches described here did not prevent visual hand-object interpenetration.

2.2 Approaches to Address Hand-Object Interpenetration

Several methods have been proposed for addressing hand-object interpenetration. Zachmann and Rettig described a grasping system based on heuristic analysis of contact (Zachmann & Rettig, 2001). Their approach included a minimization process to find a joint angle vector that kept fingers outside a grasped object. Boulic et al. instead adjust finger posture by opening the hand one joint at a time to move fingers out of a grasped object (Boulic et al., 1996). In their grasping approach, spherical sensors on the hand model guide a grasping automaton that controls grasp after user intent is determined, and grasped object motion is kinematic. In a system for manipulation of virtual objects using virtual chopsticks (Kitamura, Higashi, Masaki, & Kishino, 1999), an object was considered grasped when both chopsticks contacted it, and finger angles were adjusted such that the chopsticks did not penetrate the object visually.

Ullmann and Sauer (2000) used a physically-motivated heuristic for establishing grasps. For two-handed grasping, their system displayed both "ghost hands" that matched tracked hand configuration and solid hand models that remained attached to the outside of the grasped object. Immersion's Virtual Hand Toolkit includes an algorithm, also termed "ghost hand," that adds an offset vector to tracker position to prevent hand-object interpenetration. The offset vector is then gradually reduced to zero when interpenetration would no longer result (DesRosiers, Gomez, Tremblay, & Ullrich, 2001). This appears similar in principle to a recently proposed Credible Avatar Limb Motion technique (Burns et al., 2005).

None of these approaches incorporated realistically simulated dynamics to the extent we would like to see. However, in animation research, interpenetration of human-controlled virtual articulated structures into their environments has been addressed by using dynamic controllers or virtual springs to drive dynamic simulations based on human input (e.g., see Kokkevis, Metaxas, & Badler, 1996; Westenhofer & Hahn, 1996). Interpenetration is similarly prevented for unarticulated virtual tools coupled to haptic device handles by spring-dampers used in several force-feedback rendering algorithms (e.g., McNeely, Puterbaugh, & Troy, 1999). This force-feedback rendering technique has also been generalized and applied to a low-DOF articulated laproscopy simulator by per-segment couplings that link segments of the virtual tool to segments of a physical device (Meseure, Lenoir, Fonteneau, & Chaillou, 2004). Outside of haptics research, virtual springs have been used for interaction techniques that couple tracked hand-held devices to manipulated objects controlled by physical simulation and collision response (Froehlich, Tramberend, Beers, Agrawala, & Baraff, 2000; Koutek

& Post, 2001). Our grasping system also couples the physical world to a dynamic simulation using virtual springs, although our spring configuration is tailored to the whole-hand grasping problem and is not a direct application of any of these methods.

2.3 Force-Feedback Rendering in Grasping Systems

Most work in force-feedback rendering has focused on unarticulated virtual tools. One common technique is a constraint-based approach (e.g., Ruspini, Kolarov, & Khatib, 1997; Zilles & Salisbury, 1995), to base force on an interface point's distance from another point constrained to move along the surface of a penetrated object. Other basic approaches include ray-based (Basdogan, Ho, & Srinivasan, 1997) and voxmap (McNeely et al., 1999) techniques.

Iwata's grasping system added force feedback by transmitting maximum available torque to fingers that contacted a rigid virtual object, and computed force and moment for palm feedback based on virtual palm and object configurations. Bergamasco's point-based system computed interaction forces that included components useful for a hand force-feedback system. Popescu, Burdea, and Bouzit (1999) applied a point-based algorithm from other work (Ho, Basdogan, & Srinivasan, 1997) to gridpoints of meshes placed on virtual fingertips, summing mesh forces and applying a force mapping step that mapped resulting force to an actuator's line of action. Immersion's Virtual Hand Toolkit computes force for the glove-mounted CyberGrasp device, but details of its rendering method are not available.

3 Desktop Environment

Our grasping approach is part of a prototype system to support natural whole-hand interactions in a desktop-sized workspace. Figure 1 shows our physical display. A 21-inch monitor is placed at a 45° angle above a mirror to effectively place visual feedback in the workspace under the mirror (a summary of mirror displays and their advantages is found in Mulder & Boschker, 2004). The monitor refresh rate is 120 Hz to support stereoscopic viewing via CrystalEyes LCD shutter glasses. Our OpenGL-based visual rendering system includes projective shadows to provide additional depth cues. Anecdotally, both stereoscopy and shadows appear to improve the user's ability to locate an object when initiating a grasp, as would be expected from work by Hu, Gooch, Creem-Regehr, and Thompson (2002) and earlier work summarized therein.

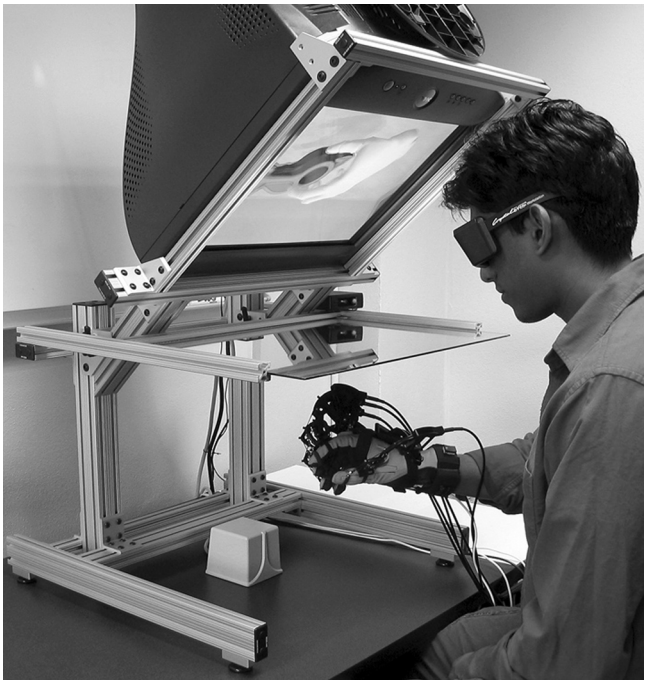

Figure 1. *Desktop environment.*

Hand configuration is tracked by a 22-sensor CyberGlove and an Ascension MiniBird magnetic tracker that is synchronized with the monitor refresh to stream readings at 120 Hz. Our hand modeling software uses a 2270-triangle geometric hand model from Viewpoint Data Labs that we modified to include a deformable palm (other than this, it consists of rigid pieces). The deformation supports the trapeziometacarpal joint in the joint model. We implemented a common joint model: Each of the four fingers has a 2-dof metacarpophalangeal (MCP) joint for abduction and flexion at the first knuckle and a 1-dof interphalangeal (IP) joint at each of the remaining two knuckles for

flexion. The thumb has a 2-dof trapeziometacarpal joint in the palm, a 1-dof MCP joint for flexion at the first knuckle, and a 1-dof IP joint for flexion at the second knuckle. The CyberGlove only provides three abduction sensors for the four fingers, so we compute middle finger abduction as a linear combination of index and ring finger abductions.

Accurate hand geometry and good calibration of glove sensors are important for accurate grasping, particularly for precision grasps. The standard automatic CyberGlove calibration did not provide the desired accuracy, although it is valuable when quick calibration is needed. We implemented a more time-consuming procedure requiring manual calibration of joint gains and offsets based on visual comparison of our hand model's joint angles to those of the real hand. We have noticed that one remaining factor limiting accuracy is the cross-coupling effect of sensors (recent work by Kahlesz, Zachmann, & Klein, 2004 provides further insight into this cross-coupling effect). To address the need for accurate hand model geometry, we have parameterized the hand model geometry to allow manual adjustment of segment sizes, but an automatic system for capturing accurate hand geometry is still needed.

For haptic feedback, we have considered a CyberGrasp device, a Rutgers Master device, and low-cost vibrotactile feedback using our own controller. Figure 1 shows the use of the CyberGrasp device. Its exoskeleton significantly reduces freedom of motion in our small workspace. The Rutgers Master solves this problem, but it has a limited finger motion range that makes certain grasps difficult (Borst & Volz, 2005), although this may be improved by visually exaggerating flexions. Since both of these devices are hand-grounded, they support internal grasping forces but do not properly simulate external forces such as gravity. The use of low-cost vibrotactile elements placed on the fingertips offers a more comfortable solution—this approach provides no forces but may still provide useful sensations while preserving full motion range. Performance data presented later in this paper were obtained using the CyberGlove with no haptic device present. Our grasping system is intended to support interactions both with and without the use of haptic devices.

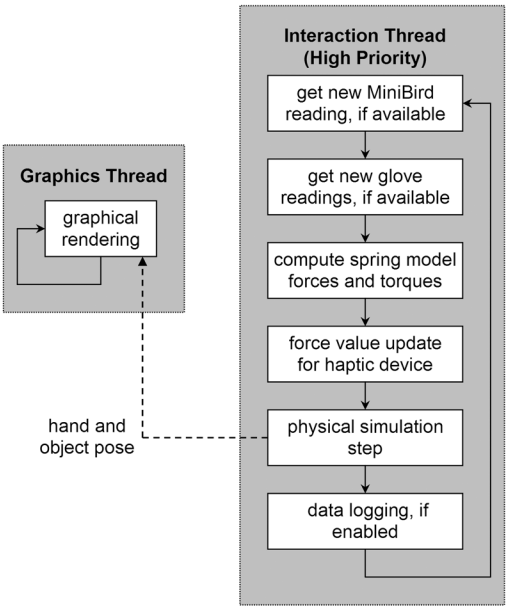

Figure 2. *Software overview.*

Our software consists of two main threads, as shown in Figure 2, running on a PC with dual 2.4 GHz Pentium 4 processors and an NVIDIA Quadro FX 3000 graphics card. The graphics thread contains a loop that performs graphical rendering. The remaining operations are performed by the interaction thread, which is assigned a high thread priority. Our custom MiniBird driver obtains a position-quaternion pair from the palm tracking system. Finger joint angles are obtained using Immersion's low-level CyberGlove driver, and these are then transformed by the offset and gain from our calibration data. Our spring model uses joint angles to compute forces and torques for a dynamic simulation and performs force feedback rendering as described in Section 4. Force values are reported to the haptic device driver, if enabled. The dynamic simulation is stepped forward by one time interval for each interaction loop iteration.

4 Grasping Approach

4.1 Overview

Our approach to grasping is to couple a dynamic hand simulation to the tracked hand configuration using

a system of virtual linear and torsional spring-dampers. This approach is inspired by the use of virtual couplings in force feedback rendering and by the use of spring models to drive articulated structures for animation, as summarized in Section 2.2. We present a spring model for an articulated hand that supports whole-hand grasping and manipulation, addresses the problem of visual interpenetration, and performs force feedback rendering.

We refer to the two hand configurations as the *spring hand* and the *tracked hand*. The visual hand model seen by users reflects the simulation-controlled spring hand configuration. As a result of the simulation, the spring hand tends to follow the tracked hand. However, collision detection and response prevent the spring hand from penetrating grasped objects, in addition to producing forces and torques for the physically-based response of grasped objects.

Figure 3 illustrates the tracked hand and the spring hand as wireframe and shaded hand models, respectively. A subset of the linear and torsional spring-dampers is also shown. We use a total of 21 torsional and 6 linear spring-dampers. There is one torsional element for each of the 20 dof provided by finger joints (illustrated only for the index finger), one torsional element and one linear element for the base of the hand (both illustrated), and one linear element for each of the five fingertips (not illustrated).

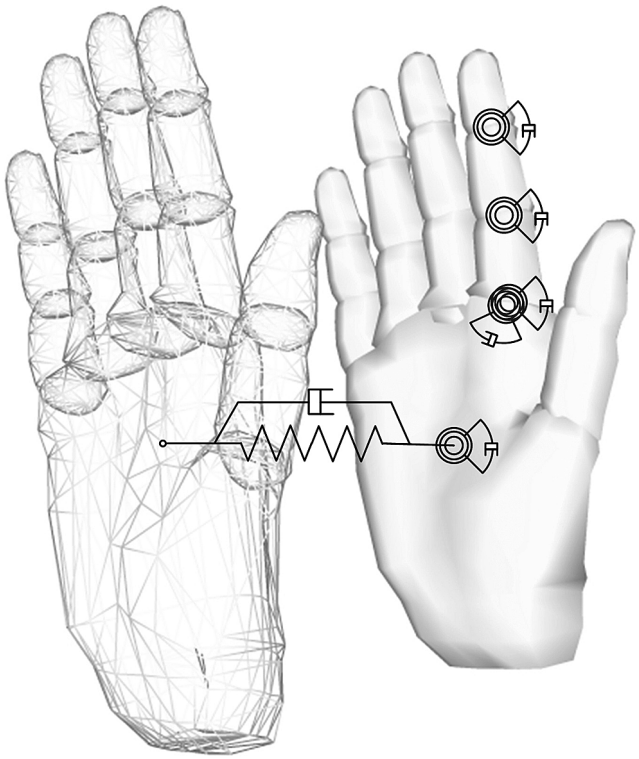

Figure 3. *The tracked hand (left), the spring hand (right), and some of the virtual spring-dampers used to couple them.*

4.2 Torsional Finger Joint Springs

For each degree of freedom of the hand joint model described in Section 3, the spring model contains a torsional spring-damper that introduces torque of magnitude

$$\tau = k(\theta_t - \theta_s) - b(\omega_s - \omega_t)$$

where

k, b are the spring and damping constants,

θ_t, θ_s are tracked joint angle and spring model joint angle, and

ω_s, ω_t are angular velocities of these joints.

The direction of the torque vector is the direction of the spring model's joint axis. Note ω_s and ω_t are computed from differences in successive joint angles, and ω_s may alternatively be available directly from the dynamic simulation. The dynamic simulation generally runs at a higher rate than the tracker update rate, so ω_t is a difference divided by the number of simulation iterations performed between receipt of two tracker readings. A similar adjustment is made in other tracked angular and linear velocities described in Section 4.

As a result of these finger joint springs, the spring hand's joint angles tend to resemble those of the tracked hand.

4.3 Torsional and Linear Palm Springs

The base of the spring hand (center of the palm) is controlled by both torsional and linear spring-dampers

so that it follows the tracked hand. The linear component computes a force applied at the base of the hand

$$\mathbf{f}_{\mathbf{palm}} = k_T(\mathbf{p}_t - \mathbf{p}_s) - b_T(\mathbf{v}_s - \mathbf{v}_t)$$

where

k_T, b_T are the translational spring and damping constants,

$\mathbf{p}_t$, $\mathbf{p}_s$ are base positions of the tracked and spring hands, and

$\mathbf{v}_s$, $\mathbf{v}_t$ are base velocities of the spring and tracked models.

The torsional component is more complex and we present its restoring torque and damping torque separately, noting these torque vectors do not generally have the same direction. Our approach uses a unit quaternion representation for orientations and computes incremental rotations from them in a manner analogous to the use of differences to compute displacements and velocities from positions. As background, note that for two unit quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$ representing orientations with respect to the same fixed frame, $(\mathbf{q}_1^*\mathbf{q}_2)$ describes the rotation from $\mathbf{q_1}$ to $\mathbf{q_2}$ with respect to a $\mathbf{q_1}$-rotated frame ($\mathbf{q}_1^*$ is the quaternion conjugate of $\mathbf{q_1}$). Furthermore, $(\mathbf{q}_2\mathbf{q}_1^*)$ describes the same rotation with respect to the fixed frame.

Let $\mathbf{q}_t$ and $\mathbf{q}_s$ represent current orientations of the tracked palm and spring palm, respectively, with respect to a fixed frame. The system computes the orientation of the tracked palm with respect to the spring palm as:

$$\mathbf{q} = \mathbf{q}_s^*\mathbf{q}_t$$

The axis $\mathbf{a_q}$ and angle $\theta_\mathbf{q}$ are extracted from $\mathbf{q}$ and a restoring torque is applied to the palm with torque vector direction given by $\mathbf{a_q}$ and magnitude given by

$$\tau_{\text{restoring}} = k_R(\theta_q)$$

where k_R is the rotational spring constant.

The resulting torque vector is in the spring hand's frame of reference, but conversion to other frames is straightforward.

To compute the damping term, the system first computes angular velocity descriptions for the spring hand and tracked hand with respect to a world frame as

$$\mathbf{q}_{\boldsymbol{\omega s}} = \mathbf{q}_{\mathbf{s1}}\mathbf{q}_{\mathbf{s0}}^* \text{ and } \mathbf{q}_{\boldsymbol{\omega t}} = \mathbf{q}_{\mathbf{t1}}\mathbf{q}_{\mathbf{t0}}^*$$

where

$\mathbf{q_{s1}}$, $\mathbf{q_{t1}}$ are the current spring and tracked model orientations, and

$\mathbf{q_{s0}}$, $\mathbf{q_{t0}}$ are their previous orientations.

Depending on the simulation tool used, better results may be obtained by computing $\mathbf{q}_{\boldsymbol{\omega s}}$ from an angular velocity description in the current simulation state. The orientation $\mathbf{q_{t0}}$ refers to a previously received tracker reading, and multiple simulation cycles may have passed between readings. Therefore, the velocity $\mathbf{q}_{\boldsymbol{\omega t}}$ is corrected by extracting the axis and angle, dividing the angle by the number of simulation iterations performed between receipt of tracker readings, and returning the result to quaternion form as $\mathbf{q}_{\boldsymbol{\omega t,adjusted}}$. A relative velocity description is then computed as

$$\mathbf{q}_{\boldsymbol{\omega}} = \mathbf{q}_{\boldsymbol{\omega s}}\mathbf{q}_{\boldsymbol{\omega t,adjusted}}^*$$

The axis $\mathbf{a}_{\mathbf{q}\boldsymbol{\omega}}$ and angle $\theta_{\mathbf{q}\boldsymbol{\omega}}$ are extracted from $\mathbf{q}_{\boldsymbol{\omega}}$ and a damping torque is applied to the palm with torque vector direction given by $\mathbf{a}_{\mathbf{q}\boldsymbol{\omega}}$ (world-referenced) and magnitude given by

$$\tau_{\text{damping}} = -b_R(\theta_{q\omega})$$

where b_R is the damping constant.

A caveat must be noted for the torsional spring model. The calculation of angular velocity finds an incremental rotation that rotates from one sampled orientation to the next. There are multiple such rotations—this can be seen by adding 2π to a rotation angle or by replacing a rotation of angle θ with a rotation of angle $-(2\pi - \theta)$. We choose the smaller magnitude rotation, since per-cycle differences in hand orientation are limited in practice by human motion limits and low-pass noise filtering in the tracking system. With the unit quaternion representation, this is done by modifying the quaternion multiplication operation to negate all terms of the result quaternion if its scalar component is negative (indicating a rotation angle with magnitude above π). Note that only a limited angular velocity magnitude is therefore represented, and this limitation is not unique to the quaternion representation (for tracked

hand velocity, it is a sampling problem). Quaternion multiplication is similarly modified for the computation of relative orientation $\mathbf{q}$ for restoring torque, but not for the calculation of relative velocity $\mathbf{q}_\omega$.

4.4 Linear Fingertip Springs

Five additional spring-damper elements were added for further control and to improve realism for certain grasp shapes. Consider, for example, a grasp in which the palmar sides of all finger phalanges of the tracked hand intersect with an object. This intersection can result in spring hand MCP flexion joints being extended to a point where the fingertips are lifted away from the object surface. The linear fingertip springs described here pull the spring hand's fingertips toward the tracked hand's fingertips to create a more realistic grasp shape in which the fingers appear wrapped around the object.

For each of the fingertips and the thumbtip, there is a linear spring-damper that computes a force applied at the tip:

$$\mathbf{f_{tip}} = k_{tip}(\mathbf{p}_{t,tip} - \mathbf{p}_{s,tip}) - b_{tip}(\mathbf{v}_{s,tip} - \mathbf{v}_{t,tip})$$

where

k_{tip}, b_{tip} are the spring and damping constants,

$\mathbf{p}_{t,tip}$, $\mathbf{p}_{s,tip}$ are a point on the tracked hand's fingertip and a corresponding point on the spring hand's fingertip, both computed by a forward kinematics routine, and

$\mathbf{v}_{s,tip}$, $\mathbf{v}_{t,tip}$ are the velocities of these points.

4.5 Force Rendering for Force Feedback

An established force rendering technique for single-point devices such as the PHANToM is to reflect forces from a virtual spring-damper coupling. We extend this concept to the articulated hand to render forces for force-feedback gloves. However, due to limited degrees of freedom of these devices, the forces and torques of our spring model must be reduced for mapping to the glove actuators. Currently we assume one degree of freedom per fingertip and compute a scalar force magnitude at each fingertip (thumbtip) as:

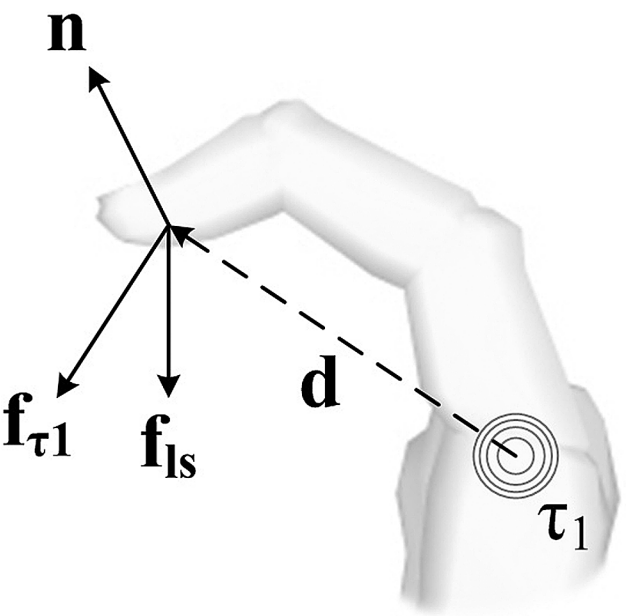

Figure 4. *Components of force-feedback rendering.*

$$f_{feedback} = -(a_1\mathbf{f}_{\tau 1} + a_2\mathbf{f}_{\tau 2} + a_3\mathbf{f}_{\tau 3} + a_4\mathbf{f}_{ls}) \cdot \mathbf{n}$$

where

a_1, a_2, a_3, and a_4 are user-chosen coefficients,

$\mathbf{f}_{\tau 1}$, $\mathbf{f}_{\tau 2}$, and $\mathbf{f}_{\tau 3}$ are three forces related to flexion torques as described below,

$\mathbf{f}_{ls}$ is a modified $\mathbf{f}_{tip}$ as described below, and

$\mathbf{n}$ is a unit vector describing the actuator's direction of force.

The equation, further illustrated by Figure 4, blends four force components and maps them onto the actuator's line of action. The simplest component is the fourth component, which projects the linear fingertip spring-damper force onto the actuator's line of action. The force $\mathbf{f}_{ls}$ is similar to the force $\mathbf{f}_{tip}$ described in Section 4.4, except that we remove the influence of palm base by not considering palm base configuration during forward kinematics for computing $\mathbf{f}_{ls}$. The force $\mathbf{f}_{tip}$ can be used instead, depending on user preference.

The first three terms provide further control over sensations by mapping flexion torques to force-feedback

vectors. In our earlier work (Borst & Indugula, 2005), the first three terms were simply proportional to the three flexion joint torques in the spring model's finger. We have extended this to further consider the grasp geometry, and illustrate this for the first flexion torque, used to compute $\mathbf{f}_{\tau 1}$ (the others are similar). Let τ_1 be the torque magnitude computed for the first flexion joint as described in Section 4.2, and $\mathbf{d}$ be a vector from the center of the spring model's joint to the fingertip point $\mathbf{p}_{s,tip}$. The vector $\mathbf{f}_{\tau 1}$ is found with direction given by the cross product of the joint axis and $\mathbf{d}$, and with magnitude $\tau_1/|\mathbf{d}|$.

4.6 Implementation Notes

In this paper, we do not deal with the details of collision detection, collision response, or numerical integration. This is because we obtained satisfactory results by using existing simulation tools to perform these steps. Basic simulation techniques, including the handling of articulated structures, are summarized by Coutinho (2001). We expected to obtain reasonable results with existing tools provided they include support for articulated structures, robust collision detection, a good friction/contact model, and implicit integration for stability. We have implemented our spring model with two freely available simulation tools: the Open Dynamics Engine (www.ode.org) and the Novodex Physics SDK V2.1.1 (www.novodex.com). The main advantage of ODE is that its source code is available, so its methods can be understood and modified. Novodex currently provides better support for polygon meshes, but minimal information is available about the approaches it uses. In the remainder of the paper, we discuss the Novodex-based implementation, which we found to produce more stable results.

4.6.1 Novodex-Based Implementation. Novodex supports various object representations, which include simple primitives, mesh, convex mesh, and mesh with associated pmap. The mesh and mesh + pmap representations were chosen for grasped objects based on our desire to support existing polygon mesh objects (the pmap is a voxelized representation used to supplement a mesh-mesh collision detection procedure—implementation details are not available). Line-swept spheres were chosen to represent finger segments after we observed that also representing the finger segments as mesh or mesh + pmap produced disturbing simulation errors (specifically, joint constraints not being maintained during grasping). The line-swept spheres represent our visual hand model's segments closely but not exactly.

Each of the 20 dof provided by finger joints was implemented using a Novodex hinge joint. Joints with two dof were implemented as two hinge joints connected by a point mass. The Novodex SDK provides its own hinge joint springs, but they do not directly match our torsional spring model for finger joints, as their damping behavior is not based on the relative angular velocities between two models. To combine our desired behavior with the Novodex springs, we used the following approach: we computed torque as in Section 4.2 and set the Novodex spring's displacement angle to be proportional to this torque magnitude, while setting a fixed high Novodex spring constant and a fixed zero Novodex damping constant (these constants are separate from the constants we used to compute torque). This results in the Novodex spring computing a restoring torque that is proportional to the torque computed by our model, which includes both restoring and damping terms.

To apply the fingertip spring forces and palm spring force and torque, we used Novodex functions to directly apply the forces and torques computed by our spring equations.

4.6.2 Parameter Value Selection and Related Issues. We tune spring model parameters based on common "rules of thumb" for simulation and force rendering systems and based on observation of spring model behavior. For fast response of the spring model during free motion, the spring constant should be high and the hand model mass should be low. The masses we assign to hand model segments are roughly proportional to segment size, with a significant mass added to the palm since there is no supporting arm model and the combined finger joint torques otherwise produce unre-

alistic movement or oscillation of the palm. Damping constants are set to low values. Spring constants can also be tuned to affect hand configuration during grasping. For example, suppose users find errors in MCP joints to be more tolerable than errors in IP joints; MCP springs can then be made relatively weak.

Parameter values also affect haptic feedback quality. Established heuristics again suggest spring constants should be set as high as possible without producing perceived instabilities, and for a small amount of damping to be added. Even during free motion, the coupling can result in spring hand mass properties being reflected to the user as force feedback. We adopt the strategy of setting feedback force to zero for any finger that does not collide with an object. Furthermore, tracked hand configuration can be used to directly set spring hand configuration until contact is established, at which time the spring model can be enabled until no more contact occurs and spring hand configuration returns to closely match tracked hand configuration.

In the Novodex implementation, much tuning of simulation parameters was required to obtain desirable behavior, and this included parameters beyond the spring model parameters. We set friction constants high to provide good grasp stability but not so high that excessive sticking resulted. We set a restitution coefficient for the hand segments to zero to prevent any elastic bounce of objects on hand surfaces. Other parameters, such as time step, depend heavily on the choice of spring model parameters and desired environment characteristics.

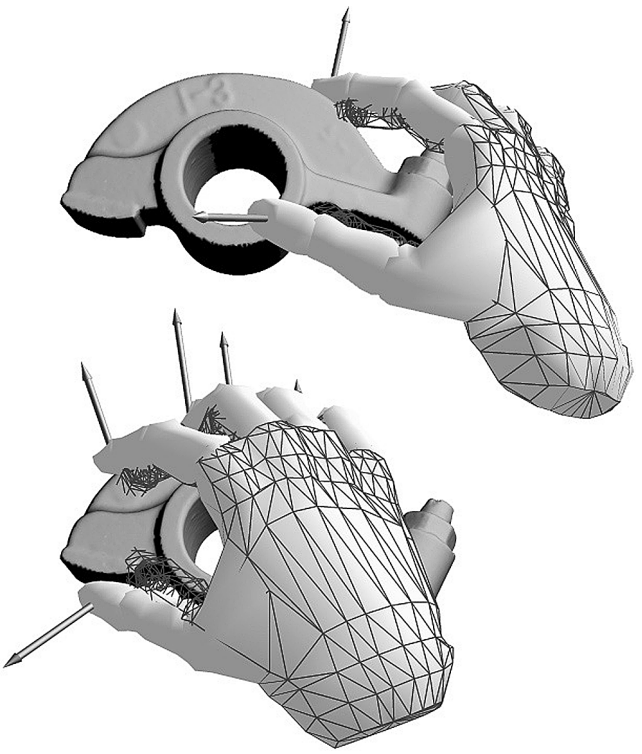

Figure 5. *Grasps of a static rocker arm showing tracked hand (mesh), spring hand (solid), and force feedback vectors.*

5 Results

In this section, we illustrate the behavior of the grasping system pictorially, discuss observed user behaviors, and discuss the performance obtained with the Novodex simulation tool.

5.1 Examples of Grasping and Manipulation

We used our system to perform interactions with a variety of rigid virtual objects. Figure 5 shows the behavior of the tracked and spring hand models during contact with a static object. The tracked hand, shown as a wireframe mesh, intersects the object. The spring hand, shown as a solid model, remains outside the object in a more realistic grasp configuration. Vectors pointing away from the fingernails show the result of force rendering. Their magnitudes are described in Section 4.5, and their directions correspond to the lines of action typical of force-feedback glove actuators.

Figure 6 illustrates manipulation of various objects as seen in a video presented at the IEEE VR 2005 conference (see http://www.cacs.louisiana.edu/~cborst/grasp/). The video shows the potential for visually realistic hand model behavior and object dynamics during grasping and manipulation. Readers familiar with work by Hirota and Hirose (2003) will note similarities in both the choice of objects and in the interactions performed—these are intentional, as their work inspired us to use a

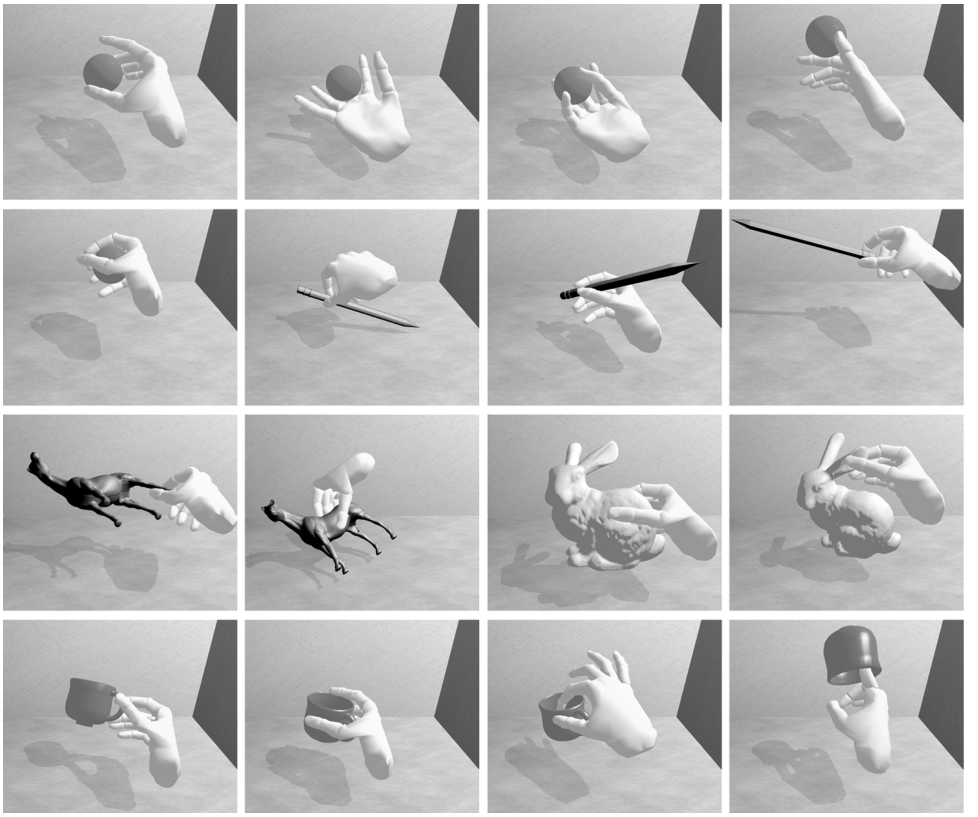

Figure 6. *Grasping and manipulation of various objects.*

dynamics-based approach and we wanted to compare results.

5.2 User Behavior and Difficulties Encountered

We have not conducted a formal user study, but we have observed the behavior of users during demonstrations in our lab and discuss results based on these informal observations. The users were not previously experienced with virtual grasping. Most demonstrations were given without the use of a haptic feedback device, due primarily to user time constraints. We intended our approach to be applicable with or without such devices.

All users of our system have successfully performed basic grasping and lifting of objects, but there is a wide variation between users in the success of their initial grasping attempt. Users encountering difficulties appear to improve quickly with subsequent attempts.

Users who encountered difficulty during their initial grasping attempts did so primarily for one of two reasons. First, some users attempted grasps using unrealistic behaviors such as pressing the palm into a virtual object rather than real-world behaviors such as grasping with the fingertips. Pressing the palm into an object does not result in a successful grasp in our virtual world (or in the real world). Users no longer exhibit this behavior once they have succeeded in grasping with more appropriate techniques. A second and more common problematic behavior is that some users tended to close their fingertips completely during grasps, regardless of object size. This complicates the release of grasps and the manipulation of grasped objects. When the real fingers are deep within the virtual object and no longer closely match the visual feedback, an object can appear to stick to the hand since the user must now use exaggerated finger motions to release or manipulate the object. A similar problem occurs in other grasping approaches.

Table 1. *Object Complexity and Resulting Performance*

Object	Verts	Tris	Cycle Time (msec)		
			Min	Ave	Max
Box primitive	—	—	0.2	0.60	1.1
Sphere primitive	—	—	0.2	0.57	1.4
Geosphere (ball)	642	1280	0.2	0.72	2.2
Teacup	3749	7494	0.3	0.80	2.5
Bunny	7788	15201	0.2	0.85	3.2
Rocker arm	10045	20088	0.3	1.10	4.4
Hi-res bunny	35947	69451	0.4	1.48	7.6
Hi-res rocker	40177	80354	0.3	1.69	14.6

Users who use a "light touch" to grasp objects are the most successful at interacting, and some users exhibit this behavior even in their initial grasping attempt. Users who encounter the difficulties described above typically improve quickly once they are advised to grasp objects with the fingertips and use a light touch. This allows users to successfully perform manipulations as shown in Figure 6. The ball, rocker arm, and bunny objects appeared to be the simplest to manipulate. Users were able to grasp these objects naturally with various grasp types, cradle the ball object or roll it in the hand, and toss and catch the ball.

Some remaining difficulties were observed during lifting by small handles such as a horse hoof or the teacup handle and in interactions with the pencil. Precision manipulations require precise tracking and hand geometry. Due to imperfections in joint angle sensing and hand model geometry, it can be difficult to achieve a thumb, index finger, and middle finger configuration that allows all three digits to contact the pencil as they would in real-world pencil use.

5.3 Performance using Novodex

The Novodex-based implementation of our spring model successfully supports a wide range of grasping interactions. The simulation is stable for grasping of basic objects with the desired light touch. For stress testing, we intentionally moved fingers deep into complex objects and tried to push the hand through objects resting on the floor. This type of behavior sometimes resulted in fingers popping into the model or moving erratically, particularly for the teacup model when fingers were pushed sideways into the rim edge from above. This suggests a possible limitation of Novodex for simulating thin objects such as the teacup sides. We initially suspected this was due to the use of a voxelized pmap representation, but found the same result using a mesh without the pmap representation. These problems are avoided by a proper grasping technique, and the stability of simulation engines is continually improving.

We measured simulation speed for grasps of various objects, as summarized in Table 1, using the PC configuration described in Section 3. The sphere and box primitives do not correspond to any previously shown objects, but were included to consider results obtainable with simple primitives. We recorded a timestamp once every tenth simulation step using a one-msec timer, so the reported minimum and maximum values are ten-cycle averages. Each table entry corresponds to an interaction sequence lasting about 60 sec. About one-third of this time was spent grasping the object with only the thumb tip and index finger tip, and another third was spent grasping the object with the whole hand to maximize collisions. We also plotted measured cycle times for all interactions as illustrated in the examples of Fig-

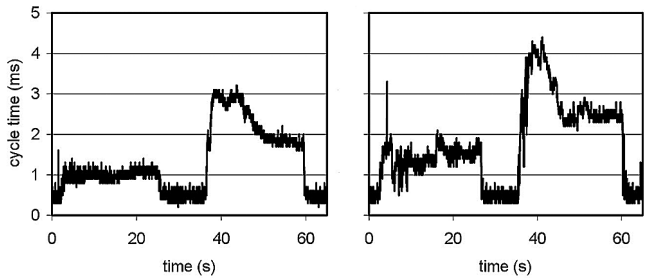

Figure 7. *Cycle times during bunny interaction (left) and rocker arm interaction (right) corresponding to Table I entries.*

ure 7. The time range of about 5 to 25 sec corresponds to the two-finger grasp. The time range of about 40 to 60 sec corresponds to the whole-hand grasp.

The simulation rates obtained for all but the highest-complexity meshes are reasonable for glove-based feedback. An update rate of 1000 Hz is often cited as a requirement for haptic rendering. However, the force-feedback gloves in our lab have low mechanical bandwidths at the fingertips that relax this requirement. According to Bouzit, Burdea, Popescu, and Boian (2002), mechanical bandwidth at the fingertip is 40 Hz for the CyberGrasp and 10 Hz for the Rutgers Master, and the Master performs valve control at 300 Hz. The CyberGrasp is used with a CyberGlove that reports hand configuration at about 150 Hz. We therefore expect force rendering rates of a few hundred updates per second to be sufficient for these devices and conclude that these were maintained for triangle meshes of up to several thousand vertices.

For the purpose of evaluating performance, no mechanism was used to control simulation rate. In practice, a stable simulation rate is needed to avoid perceivable changes in simulation speed and to provide consistent control behavior for force feedback. This can be accomplished by synchronization with an internal timer or an external device such as the tracker, as done by Hirota and Hirose (2003). Our tracker driver supports this synchronization.

6 Conclusion and Future Work

In this paper, we presented a spring model for virtual grasping that couples tracked hand configuration to a virtual hand model controlled by physical simulation. This approach supports whole-hand grasping and manipulation of virtual objects while preventing visually distracting hand/object interpenetrations. Such a grasping approach is general in that, unlike most recent approaches, it does not rely on heuristics to estimate user intent or grasp state. We also presented a new force rendering equation that uses the forces and torques of the spring model to render forces (or intensity levels) for haptic feedback gloves. We have used our grasping model in a prototype VR system for natural interactions with virtual objects in a desktop-sized workspace.

We evaluated the performance of the grasping approach when implemented using a widely available simulation tool for collision detection and response. The simulation rate was high enough to support control of force-feedback gloves with grasped objects consisting of up to several thousand triangles. We graphically illustrated the quality of the resulting grasps and identified some cases for which difficulties can still be expected.

Since force feedback can improve grasp control (Fabiani & Burdea, 1996) and impose at least some motion constraints, we expect force feedback can aid users in achieving the light touch discussed in Section 5.2. A more formal evaluation is needed to determine if this benefit of force-feedback gloves outweighs the cost of added motion limits and discomfort. Alternatives to force feedback may also be explored in future work. One approach is the use of visual indicators for users who tend to close the fingers too far. For example, instead of constraining the visual model to remain completely outside an object, it could be allowed to sink in slightly when interpenetration exceeds some threshold, or components of tracked configuration could be shown as in Figure 5. Another alternative is to manipulate the tracked hand's joint angles when interpenetration exceeds some threshold, adding an offset that is maintained for some time to bring the tracked configuration closer to the spring configuration so that smaller finger movements are sufficient to release a grasped object. Unlike the ghost hand technique in DesRosiers et al. (2001), this modified tracked hand should not avoid interpenetration completely.

Grasping interactions with the pencil model and small

object features can benefit from improved joint sensor accuracy and from a technique for determining accurate hand model geometry for arbitrary users. Another approach to deal with such cases and possibly improve grasping quality overall is to reconsider the use of heuristics, but to incorporate heuristics in a manner that preserves the benefits of the physically-based approach. We propose a hybrid system in which heuristics are used not to influence the virtual hand model directly or to make discrete control decisions, but to influence the spring model and simulation parameters in a more transparent manner to tune them to the particular interaction being performed.

Formal evaluation of grasp quality and of the haptic feedback approaches mentioned in Section 3 is desirable. Finally, more precise guidelines need to be developed for tuning simulation and spring model parameters to produce the best possible grasping behavior and force feedback. Ideally, these would include theoretical techniques such as those proposed elsewhere for tuning spring-damper parameters in other force-feedback approaches, for example, as by Adams and Hannaford (1999).

Acknowledgments

The mirror stand in our desktop environment is based on a design in Colin Ware's lab at the University of New Hampshire. Our MiniBird driver was based on a Flock of Birds driver by Greg Schmidt. We obtained geometric models online from the Stanford University Computer Graphics Laboratory (bunny), Cyberware (rocker arm), the Georgia Institute of Technology (horse originally from Cyberware), and www.3dcafe.com (uncredited pencil and teacup).

References

Adams, R. J., & Hannaford, B. (1999). Stable haptic interaction with virtual environments. *IEEE Transaction on Robotics and Automation, 15*(3), 465–474.

Basdogan, C., Ho, C., & Srinivasan, M. A. (1997). A ray-based haptic rendering technique for displaying shape and texture of 3D rigid objects in virtual environments. *Winter Annual Meeting of ASME. Dynamic Systems and Control Division, 61*, 77–84.

Bergamasco, M., Degl'Innocenti, P., & Bucciarelli, D. (1994). A realistic approach for grasping and moving virtual objects. *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 717–724.

Borst, C. W., & Indugula, A. P. (2005). Realistic virtual grasping. *IEEE Virtual Reality Conference*, 91–98, 320.

Borst, C. W., & Volz, R. A. (2003). Observations on and modifications to the Rutgers Master to support a mixture of passive haptics and active force feedback. *Eleventh Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems,* 430–437.

Borst, C. W., & Volz, R. A. (2005). Evaluation of a haptic mixed reality system for interactions with a virtual control panel. *Presence: Teleoperators and Virtual Environments, 14*(6), 677–696.

Boulic, R., Rezzonico, S., & Thalmann, D. (1996). Multi-finger manipulation of virtual objects. *ACM Symposium on Virtual Reality Software and Technology,* 67–74.

Bouzit, M., Burdea, G., Popescu, G., & Boian, R. (2002). The Rutgers Master II—New design force-feedback glove. *IEEE/ASME Transactions on Mechatronics, 7*(2), 256–263.

Bowman, D. A., Kruijff, E., LaViola, J. J., & Poupyrev, I. (2001). An introduction to 3-D user interface design. *Presence: Teleoperators and Virtual Environments, 10* (1), 96–108.

Burns, E., Razzaque, S., Panter, A. T., Whitton, M. C., McCallus, M. R., & Brooks, F. P. (2005). The hand is slower than the eye: A quantitative exploration of visual dominance over proprioception. *IEEE Virtual Reality Conference,* 3–10.

Coutinho, M. G. (2001). *Dynamic simulations of multibody systems.* New York: Springer-Verlag.

DesRosiers, H., Gomez, D., Tremblay, M., & Ullrich, C. (2001). *VirtualHand v.25 programmer's guide.* Palo Alto, CA: Virtual Technologies, Inc.

Fabiani, L., & Burdea, G. (1996). Human performance using the Rutgers Master II force feedback interface. *IEEE Virtual Reality Annual International Symposium,* 54–56.

Froehlich, B., Tramberend, H., Beers, A., Agrawala, M., & Baraff, D. (2000). Physically-based manipulation on the responsive workbench. *IEEE Virtual Reality Conference,* 5–11.

Hirota, K., & Hirose, M. (2003). Dexterous object manipula-

tion based on collision response. *IEEE Virtual Reality Conference,* 232–239.

Ho, C., Basdogan, C., & Srinivasan, M. (1997). Haptic rendering: Point- and ray-based interactions. *Second PHANToM Users Group Workshop.*

Hu, H. H., Gooch, A. A., Creem-Regehr, S. H., & Thompson, W. B. (2002). Visual cues for perceiving distances from objects to surfaces. *Presence: Teleoperators and Virtual Environments, 11*(6), 652–664.

Iwata, H. (1990). Artificial reality with force-feedback: Development of desktop virtual space with compact master manipulator. *Computer Graphics, 24*(4), 165–170.

Johansson, R. S. (1998). Sensory input and control of grip. *Novartis Foundation 218: Sensory Guidance of Movement,* 45–59.

Kahlesz, F., Zachmann, G., & Klein, R. (2004). Visual-fidelity dataglove calibration. *Computer Graphics International,* 403–410.

Kitamura, Y., Higashi, T., Masaki, T., & Kishino, F. (1999). Virtual chopsticks: Object manipulation using multiple exact interactions. *IEEE Virtual Reality Conference,* 198–204.

Kokkevis, E., Metaxas, D., & Badler, N. I. (1996). User-controlled physics-based animation for articulated figures. *Computer Animation,* 16–26.

Koutek, M., & Post, F. H. (2001). Spring-based manipulation tools for virtual environments. *Joint IPT/EGVE Conference,* 61–70.

McNeely, W. A., Puterbaugh, K. D., & Troy, J. J. (1999). Six degree-of-freedom haptic rendering using voxel sampling. *International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH),* 401–408.

Meseure, P., Lenoir, J., Fonteneau, S., & Chaillou, C. (2004). Generalized god-objects: A paradigm for interacting with physically-based virtual worlds. *Computer Animation and Social Agents,* 215–222.

Mulder, J. D., & Boschker, B. R. (2004). A modular system for collaborative desktop VR/AR with a shared workspace. *IEEE Virtual Reality Conference,* 75–82.

Popescu, V., Burdea, G., & Bouzit, M. (1999). Virtual reality simulation modeling for a haptic glove. *Computer Animation,* 195–200.

Ruspini, D. C., Kolarov, K., & Khatib, O. (1997). Haptic interaction in virtual environments. *IEEE International Conference on Intelligent Robots and Systems,* 128–133.

Ullmann, T., & Sauer, J. (2000). Intuitive virtual grasping for non haptic environments. *Pacific Graphics,* 373–380.

Welch, R. B., & Warren, D. H. (1980). Immediate perceptual response to intersensory discrepancy. *Psychological Bulletin, 88*(3), 638–667.

Westenhofer, W., & Hahn, J. (1996). Using kinematic clones to control the dynamic simulation of articulated figures. *Computer Graphics International,* 26–37.

Zachmann, G., & Rettig, A. (2001). Natural and robust interaction in virtual assembly simulation. *Eighth ISPE International Conference on Concurrent Engineering: Research and Applications.*

Zilles, C. B., & Salisbury, J. K. (1995). A constraint-based god-object method for haptic display. *IEEE International Conference on Intelligent Robots and Systems,* 146–151.