CEIG'09, San Sebastián, Sept. 9-11 (2009)
C. Andújar and J. LLuch (Editors)

Toward Haptic Rendering of Full-Hand Touch

Carlos Garre and Miguel A. Otaduy

GMRV, Universidad Rey Juan Carlos, Madrid
http://www.gmrv.es/Publications/2009/GO09a/

Abstract

Most of the current haptic rendering techniques model either force-interaction through a pen-like tool or vibration-interaction on the finger tip. Such techniques are not able, nowadays, to provide force-feedback of the interaction through the human hand. In this paper, we address some of the computational challenges in computing haptic feedback forces for hand-based interaction. We describe a haptic rendering algorithm that enables interactive contact between deformable surfaces, even with self-collisions and friction. This algorithm relies on a virtual hand model that combines rigid and deformable components, and we present the efficient simulation of such model under robust implicit integration.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual Reality—

1. Introduction

Haptic rendering is the computational technology that allows us to interact with virtual worlds through the sense of touch. It relies on an algorithm that simulates a virtual world in a physically-based manner and computes interaction forces, and a robotic device that transmits those interaction forces to the user. Haptics science is a multidisciplinary field that brings together psychophysics research for the understanding of tactile cues and human perception, mechanical engineering for the design of robotic devices, control theory for the analysis of the coupling between the real and virtual worlds, and computer science, in particular computer graphics, for the simulation of the virtual world and the design of the haptic rendering algorithm [LO08].

In the attempt to provide realistic haptic interaction with a virtual world, rendering direct hand-based touch plays a major role, as the hand is the main tactile sensor used by humans for capturing information from the world [KL03]. But, most importantly, the hand allows us to *manipulate* the world, and provides us with two-way interaction with our environment.

Despite the importance of the human hand for haptic interaction, current haptic devices and haptic rendering techniques suffer important limitations that have not allowed hand-based virtual touch to reach its full potential. Haptic rendering is typically carried out either through a pen-like robotic device, or through vibrotactile devices. But, besides hardware limitations, most of the haptic rendering algorithms are limited to point-based interaction between the user and the objects in the environment. There are some notable exceptions, both in terms of devices (e.g., exoskeleton structures [KMO*05, BBPB04, MOF*08]) and object-object rendering algorithms, also called 6-Degree-of-Freedom (6-DoF) haptic rendering [MPT99, OL06, BJ08].

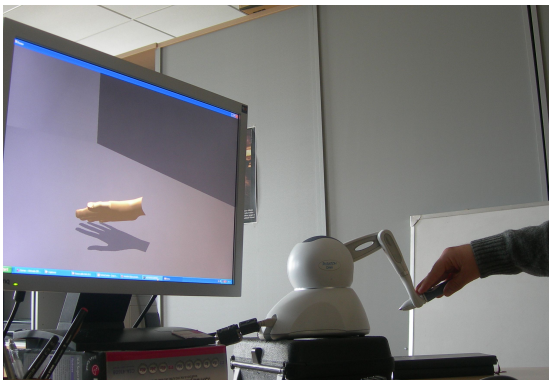

Figure 1: *Haptic manipulation of a virtual hand.*

In this paper, we take a step forward toward haptic rendering algorithms that will allow full-hand interaction with

a virtual environment, as shown in Fig. 1. Specifically, we present the following contributions:

- The adaptation of a 6-DoF haptic rendering algorithm for deformable objects (summarized in Section 3), in order to model multi-point contact between the hand and other objects. This algorithm does not suffer from the limitations of traditional single-point contact models for capturing the interaction between soft fingers and the environment.
- A virtual hand model composed of connected rigid and deformable components that can be actuated through standard haptic devices (See Section 4). We exploit the position and orientation tracked by a haptic device to guide the rigid components of the virtual hand, and model the coupling between the rigid and deformable components of the hand using stiff connections and implicit integration. As a first step toward full-hand haptic rendering, we consider the hand to be a shape formed by one rigid component and elastic flesh, and we do not take into account the articulations of the fingers.
- A computational algorithm, described in Section 5, and its corresponding object-oriented implementation, described in Section 6, that allow us to simulate a virtual hand using as black boxes existing implementations of rigid-body and deformable-body simulations. The use of implicit integration couples the degrees of freedom of the rigid and deformable components of the virtual hand, but our algorithm allows us to work around these couplings and still use existing implementations as black boxes.

We show our haptic rendering algorithm applied to the interaction between a hand model and other complex deformable objects with friction (see Fig. 5), with several moving deformable objects (see Fig. 6), or with self-collisions between the fingers (see Fig. 3).

2. Related Work

Hand-based haptic interaction with virtual environments has been researched from many different perspectives, such as the biomechanics of grasping, human grasping control, the development of robotic hands, hand-object contact models, or force-feedback devices for the fingers. We refer the reader to the book by Barbagli et al. [BPS05] for an excellent survey on the topic. Classical approaches for haptic rendering of contact between the hand and objects in the virtual environment have modeled finger-object contact as a single contact point. The fingertip is modeled as a soft tissue, and contact forces are computed using Hertz contact theory [dPSP05]. This approach exploits point-based haptic rendering methods, also called 3-DoF haptic rendering [ZS95, RKK97].

As opposed to point-based rendering methods, in this work we advocate for simulating a virtual model of the complete hand, compute contact between the virtual hand and the objects in the scene, and provide force-feedback to the user exploiting the concept of virtual coupling [CSB95]. Six-DoF haptic rendering refers precisely to the computation of force-feedback due to the interaction of a manipulated virtual object with other objects in the scene. Initially, 6-DoF haptic rendering algorithms addressed the interaction between a rigid manipulated tool and a rigid environment [MPT99, JWC05, OL06], although several more recent approaches also address contact involving a deformable tool and other deformable objects [DDKA06, OG07, BJ08, GO09]. One basic difference in our haptic rendering approach is that we simulate the hand itself as tool. In principle, our algorithm could rely on any of the existing approaches for haptic rendering of the interaction between deformable objects, but we build on a recent multi-rate approach that employs accurate constraint-based simulation models for providing rich visual feedback [OTSG09], computes a linearized version of the coupling force between tool and handle in handle-space [GO09], and then uses this linearized force model for fast computation of feedback forces.

The focus of our work is on the ability to haptically interact through a virtual hand model, but the hand model itself is rather approximate. We have employed a soft tissue model based on linear co-rotational finite elements [MG04], coupled to rigid components. More sophisticated hand models exist in computer graphics. One possibility is to employ several captured poses of the hand, and follow a pose-space deformation approach for synthesizing arbitrary hand poses based on internal joint angles [KM04]. Another possibility is to employ biomechanical models that account for internal bone, tendon, and tissue interaction [SKP08].

3. Overview of the Haptic Rendering Algorithm

Haptic rendering typically distinguishes the computation of interaction through a tool object from direct interaction with the hand. Tool-based rendering algorithms simulate a virtual model of the tool and compute contact between the tool and other objects in the environment using robust contact modeling algorithms. Direct-hand rendering algorithms, on the other hand, track points on the finger tips and model contact forces based on single-point penalty-based models (See [dPSP05] for an example). Here we present a haptic rendering algorithm that adapts classic tool-based rendering methods to compute direct-hand interaction. The key element in our algorithm is to simulate a virtual hand model in an analogous way to the simulation of a virtual tool, and employ robust contact modeling methods to compute the interaction between the virtual hand and other virtual objects. The benefits of our algorithm include the possibility to model rich and complex frictional contact, or even handle self-collisions between fingers (as shown in Fig. 3).

We assume that the haptic device operates in impedance mode, i.e., it tracks the configuration of the hand and controls the applied force. We define the configuration of the hand as a set of rigid frames, i.e., position and orientation.

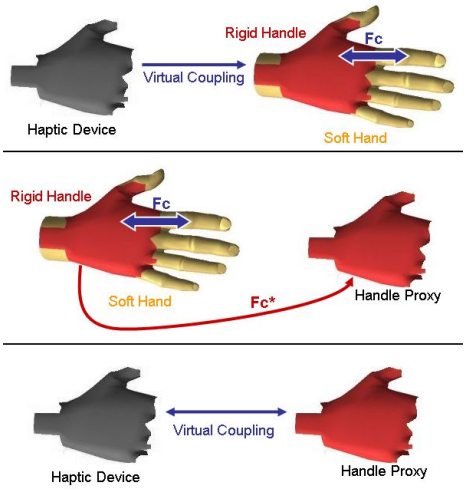

Figure 2: *Basic steps of haptic rendering based on handle-space force linerization [GO09]. Top: The haptic device is connected to a rigid handle on the hand model, which is in turn coupled through stiff springs to the rest of the soft hand. Middle: A proxy of the handle receives a linearized approximation $\mathbf{F}_c^*$ of the actual coupling forces $\mathbf{F}_c$ between the handle and the rest of the hand. Bottom: The feedback force is computed through a two-way virtual coupling between the haptic device and the proxy handle.*

This definition is applicable, for example, to pen-like haptic devices, which track one single rigid frame that corresponds to the global configuration of the hand, or to glove-like devices that track the rigid configurations of several phalanxes.

For each tracked rigid frame, we associate one rigid component in the virtual hand model, and we refer to this rigid component as *handle*. In the case of a pen-like device, the handle is placed at the palm of the hand, as shown in Fig. 2-top; and in glove-like devices the handles are placed, e.g., at the phalanxes of the virtual hand model. For the rest of this section, we will describe a situation with one single handle, as the one in Fig. 2. As shown also in Fig. 2-top, we set a one-directional viscoelastic virtual coupling between the tracked rigid frame that represents the configuration of the haptic device and the rigid handle in the virtual hand. The virtual coupling ensures that the handle follows the position and orientation tracked in the real world.

The majority of the hand is made of soft tissue; hence, we model the hand as a deformable object. Following the terminology in haptic rendering, the virtual hand model is denoted as *tool*, as it is the virtual object commanded by the haptic device. In our model, we connect the deformable hand or tool and the rigid handle with stiff springs. In Fig. 2-top, $\mathbf{F}_c$ denotes the total coupling force between handle and tool. Note that the inertial effects of the hand or the contact between the hand and other objects in the virtual environment are transmitted to the handle through the coupling force $\mathbf{F}_c$.

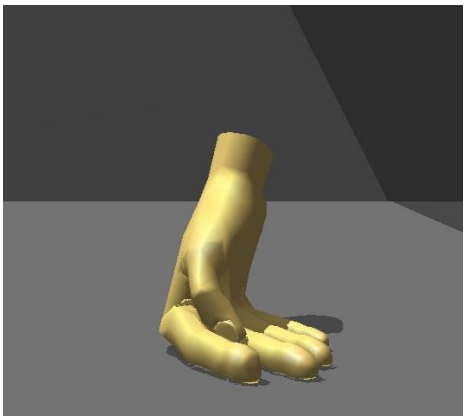

Figure 3: *Self-collision between thumb and other fingers.*

If we want a realistic and high-impedance feedback, we need a high update rate of feedback forces (e.g., 1 kHz). At such a high rate, it is not possible to compute all the tasks involved in haptic rendering: collision detection and response, simulation of the virtual environment, and elastic deformations. Hence, we use a multi-rate rendering algorithm based on linearization of the coupling force in handle space [GO09]. This algorithm simulates in a visual loop the motion of the hand model, and computes in a haptic loop the forces to be transmitted to the user. Specifically, in the visual loop we compute a constraint-based simulation of the virtual hand and other objects in the environment [OTSG09]. Constraint-based simulation of dynamic deformations with contact provides visually realistic modeling of the interaction between virtual objects.

But, besides computing the motion of the virtual environment, in the visual loop we also compute a linearized version of the coupling forces between the handle and the rest of the hand or tool. In Fig. 2-center, $\mathbf{F}_c^*$ denotes the linearized coupling force [GO09].

In the haptic loop, we simulate a proxy of the rigid handle. This proxy handle is connected through a two-way viscoelastic virtual coupling to the input rigid frame of the haptic device. But, instead of simulating the complete interaction between the handle and the rest of the virtual hand, we approximate it using the linearized version of the coupling. This linearized coupling encapsulates effects such as contact between the hand and other objects in the environment, therefore there is no need for expensive collision detection or simulation of deformations and contact in the haptic loop. Finally, the feedback force sent to the user is computed thanks to the virtual coupling between the input haptic device frame and the proxy handle, as shown in Fig. 2-bottom. Appropriate force feedback requires that the haptic device contains a force actuator associated to each tracked

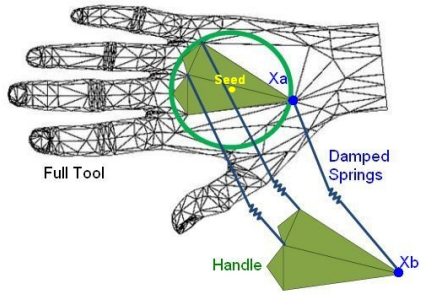

Figure 4: *Given the mesh of the deformable hand, we construct the rigid handle (in green) by picking a seed point in the palm and collecting all hand nodes inside a sphere centered at the seed. We couple the hand and the handle using stiff damped springs (in blue). Note that the hand and the handle are collocated when they are created, hence the coupling springs have zero rest length. In the figure the hand and handle are artificially separated for clarity.*

rigid frame. In other words, a pen-like device should provide one global force, and a glove-like device should provide a force for each tracked phalanx.

4. Hand Model

We have opted for modeling the hand as a deformable tool. Specifically, we use a linear co-rotational finite element model [MG04]. In order to apply the haptic rendering algorithm described in the previous section, we need to identify a rigid handle for each rigid frame tracked by the haptic device. In our case, using the Phantom Omni as haptic device, we need one rigid handle. We opt for locating the rigid handle at the palm of the hand model. We need to define the rigid-body properties of the handle, but we do not want to sculpt a separate 3D model. Therefore, given the tetrahedral mesh of the whole hand, we define the rigid handle by selecting a seed point in the palm and collecting all hand nodes inside a sphere centered at the seed. Fig. 4 depicts a 2D example of hand and handle design.

The state $\mathbf{x}_t \in \mathbb{R}^{3n}$ and velocity $\mathbf{v}_t \in \mathbb{R}^{3n}$ of the deformable hand (a.k.a. tool) are defined by the positions and velocities of all the n nodes that form the tetrahedral mesh used for the dynamic simulation. The state $\mathbf{x}_h \in \mathbb{R}^7$ of the handle, on the other hand, comprises the position of its center of mass $\mathbf{x}_{com}$ and its orientation, defined as a quaternion $\mathbf{q}_h$. The velocity $\mathbf{v}_h \in \mathbb{R}^6$ of the handle comprises the velocity of the center of mass $\mathbf{v}_{com}$ and the angular velocity ω_h. For clarity, here we summarize all state and velocity vectors as column vectors:

$$\begin{aligned} \mathbf{x}_t &= (\mathbf{x}_1^T, \mathbf{x}_2^T, \dots \mathbf{x}_n^T)^T, & \mathbf{x}_h &= (\mathbf{x}_{com}^T, \mathbf{q}_h^T)^T, \\ \mathbf{v}_t &= (\mathbf{v}_1^T, \mathbf{v}_2^T, \dots \mathbf{v}_n^T)^T, & \mathbf{v}_h &= (\mathbf{v}_{com}^T, \omega_h^T)^T. \end{aligned} \quad (1)$$

The mass matrix of the tool hand $\mathbf{M}_t$ is defined by lumping masses at the nodes, i.e.,

$$\mathbf{M}_t = \begin{pmatrix} m_1\mathbf{I}_{3\times3} & 0 & \dots & 0 \\ 0 & m_2\mathbf{I}_{3\times3} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & m_n\mathbf{I}_{3\times3} \end{pmatrix}. \quad (2)$$

The mass matrix of the handle is defined as

$$\mathbf{M}_h = \begin{pmatrix} m_h\mathbf{I}_{3\times3} & 0 \\ 0 & \mathbf{M}_q \end{pmatrix}, \quad (3)$$

where m_h is the total mass of the rigid handle and $\mathbf{M}_q$ is its inertia matrix. We compute the mass and inertia properties of the handle from the lumped masses of the selected hand nodes [Mir96].

In order to couple the hand and the handle, we set stiff damped springs between pairs of corresponding nodes of both models. The springs model two-way coupling as a vector of internal forces F_c. Given that the hand and handle are collocated when they are created, these springs have zero rest length. This fact largely simplifies the formulation of the spring force, which becomes linear in the positions of the end points. Let us consider a spring between a node $\mathbf{x}_A$ of the hand and a point $\mathbf{x}_B$ on the handle (as shown in Fig. 4), with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$. Then, the force $\mathbf{F}_A$ acting on the node of the hand can be written as:

$$\begin{aligned} &\mathbf{F}_A = -k(\mathbf{x}_A - \mathbf{x}_B) - d(\mathbf{v}_A - \mathbf{v}_B), \\ &\text{with } \mathbf{x}_B = \mathbf{x}_{com} + \mathbf{R}_h\mathbf{r}_B, \\ &\text{and } \mathbf{v}_B = \mathbf{v}_{com} + \omega_h \times (\mathbf{R}_h\mathbf{r}_B). \end{aligned} \quad (4)$$

Note that the position and velocity of the point $\mathbf{x}_B$ are expressed in terms of the handle state and velocity vectors, with $\mathbf{R}_h$ a rotation matrix that represents the orientation of the handle, and $\mathbf{r}_B$ the position of $\mathbf{x}_B$ in the local reference frame of the handle. The coefficients k and d are the stiffness and damping parameters of the spring.

Likewise, the opposite force acting on the handle is $\mathbf{F}_B = -\mathbf{F}_A$, while the torque can be written as

$$\mathbf{T}_B = (\mathbf{R}_h\mathbf{r}_B) \times \mathbf{F}_B. \quad (5)$$

In order to ensure stiff yet stable coupling between the hand and the handle, we integrate the dynamics equations using implicit integration methods. Without coupling springs, the dynamics equations of the hand and the handle can be solved separately, but the coupling also couples the systems of equations. In the next section we present an algorithm for solving the coupled hand and handle dynamics.

5. Implicit Handle-Hand Coupling

As part of the haptic rendering algorithm, we need to solve a constrained dynamics problem involving the deformable

hand, the rigid handle, and possibly other dynamic objects in the virtual environment. Then, the coupling force between the tool-hand and the handle must be linearized w.r.t. the handle state, accounting for contact constraints and inertial effects. Both the constrained dynamics solver and the linearization require solving in an efficient manner the coupled dynamics of the tool-hand and the handle. The big picture about the constrained dynamics solver and the handle-space linearization has been published elsewhere [OTSG09, GO09], but here we describe in detail our novel algorithm for efficiently solving the coupled dynamics of the complete hand model. We first discuss the formulation of the coupling terms, and then their efficient solution. Note that here, for clarity reasons, we discuss in detail the constraint-free setting, but the same algorithm constitutes a building block of the constrained setting.

5.1. Formulation of the Coupling Terms

Grouping all forces acting on the hand tool (including contact constraint forces) in a vector $\mathbf{F}_t \in \mathbb{R}^{3n}$, and the forces and torques acting on the handle in a vector $\mathbf{F}_h \in \mathbb{R}^6$, the dynamics equations of motion can be written as:

$$\begin{aligned} \mathbf{F}_h &= \mathbf{M}_h \dot{\mathbf{v}}_h, \\ \mathbf{F}_t &= \mathbf{M}_t \dot{\mathbf{v}}_t. \end{aligned} \tag{6}$$

We discretize the dynamics equations using implicit integration [BW98], which allows the use of large time steps even with stiff forces. In our case we have used the semi-implicit Euler method with linearized forces. Implicit integration yields a linear system of equations of the form:

$$\begin{pmatrix} \mathbf{A}_h & \mathbf{A}_{ht} \\ \mathbf{A}_{th} & \mathbf{A}_t \end{pmatrix} \begin{pmatrix} \mathbf{v}_h \\ \mathbf{v}_t \end{pmatrix} = \begin{pmatrix} \mathbf{b}_h \\ \mathbf{b}_t \end{pmatrix}. \tag{7}$$

In this system, the matrices $\mathbf{A}_h$ and $\mathbf{A}_t$ are related to the uncoupled dynamics of the hand and the handle. Just for completeness, $\mathbf{A}_h$ is a dense 6×6 matrix, while $\mathbf{A}_t$ is a sparse, symmetric, positive definite, $3n \times 3n$ matrix. We will focus here on the terms $\mathbf{A}_{ht}$ and $\mathbf{A}_{th}$, which arise due to the implicit integration of the coupling between hand and handle.

With implicit Euler, the matrix $\mathbf{A}_{ht}$ is formulated as:

$$\mathbf{A}_{ht} = \mathbf{M}_{ht} - \Delta t \frac{\partial \mathbf{F}_h}{\partial \mathbf{v}_t} - \Delta t^2 \frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_t}. \tag{8}$$

Here, the term $\mathbf{M}_{ht} = 0$ because there is no mass coupling between handle and hand. The term $\frac{\partial \mathbf{F}_h}{\partial \mathbf{v}_t}$ represents the Jacobian of the coupling forces (and torques) acting on the handle w.r.t. hand velocities. We leave this term to the reader, as it is less complex than the term $\frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_t}$, which represents the Jacobian of coupling forces acting on the handle w.r.t. the hand state. This last term can be decomposed in a block representation as follows:

$$\frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_t} = \begin{pmatrix} \frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_n} \end{pmatrix}. \tag{9}$$

Each 6×3 submatrix captures the Jacobian of coupling force and torque acting on the handle w.r.t. the position of one node of the tool hand. Then, the terms for hand nodes $\{\mathbf{x}_i\}$ that are not connected by springs to the handle are $\frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_i} = 0$, and the non-zero terms capture the Jacobian of one and only one spring.

The matrix $\mathbf{A}_{th}$ in Eq. (7) is computed similarly as:

$$\mathbf{A}_{th} = \mathbf{M}_{th} - \Delta t \frac{\partial \mathbf{F}_t}{\partial \mathbf{v}_h} - \Delta t^2 \mathbf{G} \frac{\partial \mathbf{F}_t}{\partial \mathbf{x}_h}. \tag{10}$$

Here, the matrix $\mathbf{G}$ relates the angular velocity of the handle to the derivative of its orientation (represented as a quaternion in our case) [Sha89]. The term $\frac{\partial \mathbf{F}_t}{\partial \mathbf{x}_h}$ is formed by rows $\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_h}$ that represent the Jacobian of the coupling force acting on nodes $\mathbf{x}_i$ of the hand w.r.t. the velocity vector of the handle. Similar to the observation discussed earlier, a non-zero term $\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_h}$ refers to one and only one coupling spring between the hand and the handle.

From the derivation above, the assembly of the matrices $\mathbf{A}_{ht}$ and $\mathbf{A}_{th}$ is reduced to computing the Jacobians of the coupling spring forces described in Section 4. Recall that $\mathbf{F}_A$ in Eq. (4) represents a spring force acting on the node $\mathbf{x}_A$ of the tool hand, $\mathbf{F}_B = -\mathbf{F}_A$ represents the force acting on the handle, and $\mathbf{T}_B$ in Eq. (5) represents the spring torque acting on the handle. The Jacobians in $\frac{\partial \mathbf{F}_h}{\partial \mathbf{x}_A}$ and $\frac{\partial \mathbf{F}_A}{\partial \mathbf{x}_h}$ can then be computed as:

$$\begin{aligned} \frac{\partial \mathbf{F}_B}{\partial \mathbf{x}_A} &= k\mathbf{I}, \\ \frac{\partial \mathbf{T}_B}{\partial \mathbf{x}_A} &= (\mathbf{R}_h \mathbf{r}_B) \times \frac{\partial \mathbf{F}_B}{\partial \mathbf{x}_A}, \\ \frac{\partial \mathbf{F}_A}{\partial \mathbf{x}_{com}} &= k\mathbf{I}, \\ \frac{\partial \mathbf{F}_A}{\partial \mathbf{q}_h} &= k\frac{\partial (\mathbf{R}_h \mathbf{r}_B)}{\partial \mathbf{q}_h} + d \cdot \omega_h \times \frac{\partial (\mathbf{R}_h \mathbf{r}_B)}{\partial \mathbf{q}_h}. \end{aligned} \tag{11}$$

The detailed definition of the term $\frac{\partial (\mathbf{R}_h \mathbf{r}_B)}{\partial \mathbf{q}_h}$ is given in Appendix A.

5.2. Resolution of the Coupled System

At every time step of the dynamics simulation, the computation of the handle and tool-hand velocities requires the solution to the linear system in Eq. (7). One possibility for doing so could be to formulate one large system containing all equations and the use of a state-of-the-art linear system solver for computing all velocities at once. However, this possibility suffers from two disadvantages: (i) the matrices $\mathbf{A}_{ht}$ and $\mathbf{A}_{th}$ might be rather dense if the handle is large (i.e., it contains many nodes of the hand), thereby complicating the use of solvers for sparse systems, and (ii) the choice of solver would require knowing the particular characteristics of the matrices $\mathbf{A}_h$ and $\mathbf{A}_t$, hence the implementations of rigid body simulation (for the handle) and deformable body

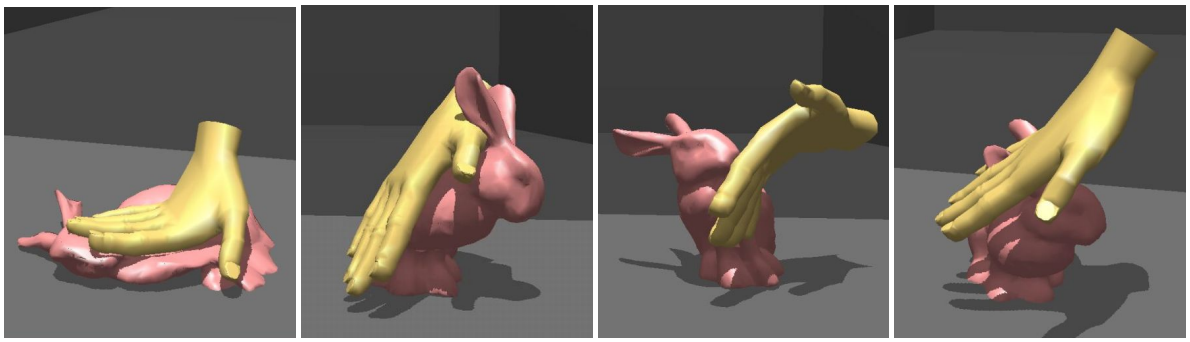

Figure 5: *Virtual hand touching a bunny model. Notice the large deformations (in particular in the leftmost figure), large-area contact, intricate contact (see the ear sticking out between the fingers in the rightmost figure), and friction effects, all computed interactively with our algorithm.*

simulation (for the hand) could not be interpreted as black boxes in the coupled implementation, thereby complicating the job of the programmer.

Instead, we propose a solution to Eq. (7) that can treat the rigid-body simulation and the deformable-body simulation as black boxes, and does not require implementing additional linear-system solvers. Our solution is based on a sequential solution of $\mathbf{v}_t$ and $\mathbf{v}_h$ based on the computation of a Schur complement [GV96] in Eq. (7). We first single out $\mathbf{v}_t$ in the second row of Eq. (7) as:

$$\mathbf{v}_t = \mathbf{A}_t^{-1}(\mathbf{b}_t - \mathbf{A}_{th}\mathbf{v}_h). \tag{12}$$

We then substitute the result in the first row, which yields the following linear system:

$$(\mathbf{A}_h - \mathbf{A}_{ht}\mathbf{A}_t^{-1}\mathbf{A}_{th})\mathbf{v}_h = \mathbf{b}_h - \mathbf{A}_{ht}\mathbf{A}_t^{-1}\mathbf{b}_t. \tag{13}$$

Here, $\mathbf{A}_h - \mathbf{A}_{ht}\mathbf{A}_t^{-1}\mathbf{A}_{th}$ is the Schur complement of $\mathbf{A}_t$. The velocity of the handle, $\mathbf{v}_h$, can be solved through Gaussian elimination in Eq. (13). Then, the solution is plugged in Eq. (12), and we solve for the velocity of the tool, $\mathbf{v}_t$. Note that we do not invert $\mathbf{A}_t$ in Eq. (12), and solve it instead using the Conjugate Gradient method.

The reader may observe that there are two options for solving Eq. (7) in a sequential manner using Schur complements. Our choice, which formulates the Schur complement of $\mathbf{A}_t$, requires the solution to 8 large linear systems of the form $\mathbf{A}_t\mathbf{s} = \mathbf{r}$: 1 in the computation of the right-hand side in Eq. (13), 6 in the computation of $\mathbf{A}_t^{-1}\mathbf{A}_{th}$ (one per column of $\mathbf{A}_{th}$), and 1 in the final computation of $\mathbf{v}_t$ in Eq. (12). These large linear systems are efficiently solved with the Conjugate Gradient method. Another choice, which formulates the Schur complement of $\mathbf{A}_h$, may suffer from dense matrices. The term $\mathbf{A}_{th}\mathbf{A}_h^{-1}\mathbf{A}_{ht}$ would contain a non-zero block for each pair of nodes of the hand connected to the handle, and its density would grow with the size of the handle. Our solution, on the other hand, scales linearly (as desired) with the number of nodes in the handle.

6. Object-Oriented Implementation

We assume that someone willing to program an object-oriented hand-based haptic rendering system may have at hand implementations of rigid-body and deformable-body simulations. When using an object-oriented programming paradigm, it is very important to reuse the code encapsulated in the existing classes by using them as "black boxes". As discussed earlier, implicit integration, although very robust for solving stiff forces, causes a coupling between the handle and the tool-hand that complicates the reuse of the rigid-body and deformable-body implementations at hand. The solution that we have presented in Section 5.2 allows the reuse of those implementations, and now we will outline pseudocode for implementing our proposed solution.

All the dynamic bodies in the simulation will be able to formulate and solve a linear system for implicit integration of the form $\mathbf{As} = \mathbf{r}$. They will also share a very simple common class interface consisting of methods for formulating the linear system, setting and reading the linear-system matrix $\mathbf{A}$ and the right-hand side $\mathbf{r}$, reading the solution $\mathbf{s}$, and solving the linear system. Then, we assume that we are given black-box implementations of a rigid-body handle and a deformable-body tool-hand that agree with the specified class interface.

Our object-oriented implementation of the haptic rendering algorithm implements the complete hand model, comprising both the deformable tool and the rigid handle, using yet the same class interface. The complete haptic hand contains as data members the rigid handle and the deformable tool, and implements the solution to its own linear system by issuing calls to the internal methods of the deformable- and rigid-body classes. It also contains as data members the coupling matrices $\mathbf{A}_{th}$ and $\mathbf{A}_{ht}$. Then, the following pseudocode, based on the algorithm from Section 5.2, deals with the implementation of the method for solving the linear system of the complete haptic hand, i.e., Eq. (7):

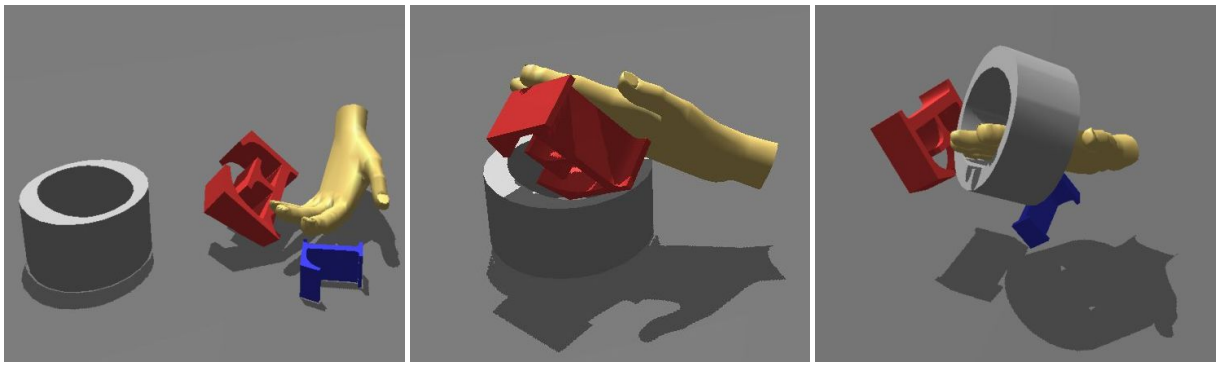

Figure 6: *Our haptic rendering algorithm also serves to touch moving objects (such as the letters) with a virtual hand.*

\\Formulate the right-hand side of Eq. (13)
  Tool.`SolveLinearSystem()`
  Handle.$\mathbf{r} = \mathbf{b}_t$ - $\mathbf{A}_{ht} \cdot$ Tool.$\mathbf{s}$

\\Formulate the Schur complement in Eq. (13).
  For i = 1 to 6 do \\Solve each column
    Tool.$\mathbf{r} = \mathbf{A}_{th}$(i)
    Tool.`SolveLinearSystem()`
    temp(i) = Tool.$\mathbf{s}$
  Handle.$\mathbf{A} = \mathbf{A}_h$ - $\mathbf{A}_{ht} \cdot$ temp

\\Solve for the handle velocity $\mathbf{v}_h$
  Handle.`SolveLinearSystem()`
  $\mathbf{v}_h$ = Handle.$\mathbf{s}$

\\Formulate the right-hand side of Eq. (12)
  Tool.$\mathbf{r} = \mathbf{b}_t$ - $\mathbf{A}_{th} \cdot \mathbf{v}_h$

\\Solve for the tool-hand velocity $\mathbf{v}_t$
  Tool.`SolveLinearSystem()`
  $\mathbf{v}_t$ = Tool.$\mathbf{s}$

7. Experiments

We have executed our experiments on a quad-core 2.4 GHz PC with 3 GB of memory (although we have only used two processors, for the visual and haptic loops) and a GeForce 8800 GTS. We manipulated the models using a Phantom Omni haptic device from SensAble Technologies.

Our haptic rendering algorithm successfully models interaction through a virtual hand. The surface of the hand model is modeled with 1733 triangles, and we embed the surface in a regular tetrahedral mesh to create the elastic finite element model as described in Section 4. We have tested meshes ranging from 350 (in Fig. 5) to 1700 tetrahedra (in Fig. 3). The rigid handle is located in the palm of the hand, and covers, e.g., 79 nodes for the mesh with 350 tetrahedra.

Fig. 5 shows situations with large-area contact, intricate contact, or large deformation when touching a bunny model. In the bunny, we use a surface mesh with 4000 triangles for collision detection, and a tetrahedral mesh with 271 tetrahedra for the deformation. Fig. 6 depicts several snapshots of a scene where the virtual hand interacts with moving letters. The letters are modeled with meshes with 175 triangles and 65 tetrahedra on average. Fig. 3 on the other hand, shows that we can also handle self-contact among fingers of the hand itself.

The visual simulation runs at an average of 30 fps in situations with contact, while feedback forces are computed at a frequency of 1 kHz. To a large extent, the high efficiency of the constrained dynamics simulation is obtained thanks to our efficient handling of the implicit coupling between the rigid and deformable components of the hand model.

8. Discussion and Future Work

We have presented a haptic rendering algorithm for computing force interaction between the hand and virtual models. Our algorithm differs from previous rendering algorithms as it employs a virtual hand model and the concept of virtual coupling for modeling the interaction. This hand model couples rigid and deformable components, and we have presented a computational algorithm that solves efficiently implicit integration under such coupling, thus allowing effective haptic rendering.

Our proposed algorithm constitutes a step forward toward full-hand haptic interaction, but there are still unsolved problems. In the future we will explore hand models that combine soft tissue and an articulated model for the skeleton. The associated challenges deal mostly with the extension of virtual coupling approaches to the articulated skeleton. We consider solutions for modeling skeletal posture statistically from a few tracked phalanxes [CFSU*08], or control approaches for synthesizing posture from sparse data [Liu08].

We are also interested in the evaluation of our model in terms of perception, in aspects such as sensory substitution from full-hand touch to a stylus-type device, or the use of a 3-DoF device not capable of transmitting torque. We would also like to test our haptic rendering algorithm on glove-like

or exoskeleton haptic devices, which would enable tracking of and applying force-feedback to each finger independently.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, and the GMRV group at URJC. This work was funded in part by the URJC - Comunidad de Madrid project CCG08-URJC/DPI-3647.

References

[BBPB04] BOUZIT M., BURDEA G., POPESCU G., BOIAN R.: The rutger master ii - new design force-feedback glove. *IEEE Transactions on Mechatronics 7*, 2 (2004).

[BJ08] BARBIČ J., JAMES D. L.: Six-DoF haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Trans. on Haptics 1*, 1 (2008).

[BPS05] BARBAGLI F., PRATTICHIZZO D., SALISBURY K.: *Multi-Point Interaction with Real and Virtual Objects*. Springer, 2005.

[BW98] BARAFF D., WITKIN A. P.: Large steps in cloth simulation. *Proc. of ACM SIGGRAPH* (1998).

[CFSU*08] COBOS S., FERRE M., SANCHEZ URAN M., ORTEGO J., PENA C.: Efficient human hand kinematics for manipulation tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), pp. 2246–2251.

[CSB95] COLGATE J. E., STANLEY M. C., BROWN J. M.: Issues in the haptic display of tool use. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (1995), pp. 140–145.

[DDKA06] DURIEZ C., DUBOIS F., KHEDDAR A., ANDRIOT C.: Realistic haptic rendering of interacting deformable objects in virtual environments. *Proc. of IEEE TVCG 12*, 1 (2006).

[dPSP05] DE PASCALE M., SARCUNI G., PRATTICHIZZO D.: Real-time soft-finger grasping of physically based quasi-rigid objects. In *Proc. World Haptics Conference* (Pisa, Italy, March 18-20 2005), pp. 545–546.

[GO09] GARRE C., OTADUY M. A.: Haptic rendering of complex deformations through handle-space force linearization. In *Proc. of World Haptics Conference* (mar 2009), IEEE Robotics and Automation Society.

[GV96] GOLUB G. H., VAN LOAN C. F.: *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[JWC05] JOHNSON D. E., WILLEMSEN P., COHEN E.: 6-DOF haptic rendering using spatialized normal cone search. *IEEE TVCG 11*, 6 (2005).

[KL03] KLATZKY R. L., LEDERMAN S. J.: Touch. In *Experimental Psychology* (2003), pp. 147–176. Volume 4 in I.B. Weiner (Editor-in-Chief). Handbook of Psychology.

[KM04] KURIHARA T., MIYATA N.: Modeling deformable human hands from medical images. In *Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2004), pp. 357–366.

[KMO*05] KAWASAKI H., MOURI T., OSAMA M., SUGIHASHI Y., OHTUKA Y., IKENOHATA S., KIGAKU K., DANIULAITIS V., HAMADA K., SUZUKI T.: Development of five-fingered haptic intereface: Hiro ii. In *Proc. of ICAT* (2005).

[Liu08] LIU C. K.: Synthesis of interactive hand manipulation. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation)* (2008).

[LO08] LIN M. C., OTADUY M. A. (Eds.): *Haptic Rendering: Foundations, Algorithms, and Applications*. AK Peters, 2008.

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. *Proc. of Graphics Interface* (2004).

[Mir96] MIRTICH B.: Fast and accurate computation of polyhedral mass properties. *Journal of Graphical Tools 1*, 2 (1996).

[MOF*08] MONROY M., OYARZABAL M., FERRE M., CAMPOS A., BARRIO J.: Masterfinger: Multi-finger haptic interface for collaborative environments. In *Proc. of Eurohaptics* (2008).

[MPT99] MCNEELY W., PUTERBAUGH K., TROY J.: Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH* (1999), 401–408.

[OG07] OTADUY M. A., GROSS M.: Transparent rendering of tool contact with compliant environments. *Proc. of World Haptics Conference* (2007).

[OL06] OTADUY M. A., LIN M. C.: A modular haptic rendering algorithm for stable and transparent 6-DoF manipulation. *IEEE Trans. on Robotics 22*, 4 (2006).

[OTSG09] OTADUY M. A., TAMSTORF R., STEINEMANN D., GROSS M.: Implicit contact handling for deformable objects. *Proc. of Eurographics* (2009).

[RKK97] RUSPINI D., KOLAROV K., KHATIB O.: The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH* (1997), 345–352.

[Sha89] SHABANA A. A.: *Dynamics of Multibody Systems*. John Wiley and Sons, 1989.

[SKP08] SUEDA S., KAUFMAN A., PAI D. K.: Musculotendon simulation for hand animation. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008).

[ZS95] ZILLES C., SALISBURY K.: A constraint-based god object method for haptics display. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems* (1995).

Appendix A: Jacobians of Rotation Terms

The vector $\mathbf{R}_h\mathbf{r}_B$ represents the vector from the center of mass of the handle, $\mathbf{x}_{com}$, to point $\mathbf{x}_B$, expressed in the global reference frame. It can be computed based on the unit quaternion $\mathbf{q}_h$ of the handle as $\mathbf{R}_h\mathbf{r}_B = \mathbf{q}_h \circ \mathbf{q}(\mathbf{r}_B) \circ \mathbf{q}'_B$. Here, $\circ$ represents quaternion product, $\mathbf{q}'_B$ the conjugate of $\mathbf{q}_B$, and $\mathbf{q}(\mathbf{r}_B)$ a quaternion constructed with $\mathbf{r}_B$ as the vector part and zero as the scalar part. From this expression, we derive the Jacobian

$$\frac{\partial(\mathbf{R}_h\mathbf{r}_B)}{\partial\mathbf{q}_h} = \mathbf{Q}\mathbf{C} + \bar{\mathbf{Q}}, \quad \text{with} \tag{14}$$

$$\mathbf{Q} = \begin{pmatrix} s & -z & y & x \\ z & s & -x & y \\ -y & x & s & z \end{pmatrix}, \quad \mathbf{q} = (x,y,z,s)^T,$$

$$\bar{\mathbf{Q}} = \begin{pmatrix} \bar{s} & \bar{z} & -\bar{y} & \bar{x} \\ -\bar{z} & \bar{s} & \bar{x} & \bar{y} \\ \bar{y} & -\bar{x} & \bar{s} & \bar{z} \end{pmatrix}, \quad \mathbf{q}(\mathbf{r}_B) \circ \mathbf{q}'_B = (\bar{x},\bar{y},\bar{z},\bar{s})^T,$$

$$\mathbf{C} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$