Interactive Dynamic Simulation using Haptic Interaction*

Wookho Son
Texas A&M
wooks@cs.tamu.edu

Kyunghwan Kim
Korea Inst.of Sci&Tech
kimk@kist.re.kr

Nancy M. Amato
Texas A&M
amato@cs.tamu.edu

Jeffrey C. Trinkle
Sandia Natl Lab
trink@sandia.gov

Abstract

This paper describes an interactive dynamic simulator for virtual environments which allows user interaction via a haptic interface. The interactive simulation is performed in our testbed dynamic simulator I-GMS, which has been developed in an object-oriented framework for simulating motions of free bodies and complex linkages such as those needed for robotic systems or human body simulation. User-interaction is achieved by performing push and pull operations via the PHANToM haptic device which runs as an integrated part of I-GMS. We demonstrate the user-interaction capability of I-GMS through on-line editing of trajectories for a 6-dof robot manipulator.

Keywords: Dynamic Simulation, Haptic Interaction, Human Body Model, Object-Oriented Design, Interactive Simulation.

1 Introduction

Dynamic motion simulation arises in many engineering application domains such as virtual reality (VR), graphics, robot motion simulators, and computer games. Interaction with the virtual environment by the user in real-time is becoming increasingly important in computer games. This interaction is achieved not only via the user's textual input, but also via direct touch of an object in the virtual scene. The addition of force and touch (haptic feedback) to dynamic simulation increases the simulation realism when the virtual objects are manipulated by the user during a simulation. In particular, this sensory modality is highly desirable when the graphics are corrupted due to partial occlusion of manipulated objects during simulation or when the environment is dark.

In general, a simulation environment contains multi-rigid-body systems, each of which consists of a number of passive bodies, called *free bodies*, that move in response to external forces or forces arising from contacts, and a number of *active* bodies which are actuated. Dynamic simulation predicts the accelerations (and contact forces of rigid bodies in contact) of the multi-rigid-body systems in the environment.

Adding haptic interaction to the dynamic simulation has the effect of exerting user-applied external forces to the active bodies in the scene to change their dynamic behavior. In other words, it changes the course of simulation trajectories by keeping track of changes in dynamics due to outside disturbances such as contact. This has many applications such as teleoperation of robots for remote inspection, virtual training, etc.

In this paper, we present two methods of realizing interactive dynamic simulation via haptic feedback, *push* and *pull* modes. We performed the interactive simulation in our testbed dynamic simulator I-GMS [10], which can simulate dynamic motions of multi-rigid-bodies in virtual environments. In particular, the use of user interaction is focused on on-line editing and modification of trajectories of articulated robots.

2 Related Work

Early work on haptic interaction has focused on haptic rendering of graphical environments, and has used force feedback coupled with the visual display to realize surface shading, friction and texture [8] [12].

There are some dynamic simulators with haptic interaction capability. The use of an impulse-based simulation as a general purpose multi-body simulator for haptic display is presented in [2]. Haptic interaction for a point contact for rigid body dynamics is studied in [11]. Berkelman *et al.*[1] provides a tool-based haptic interaction where the user feels and interacts with the simulation environment through a rigid tool of a given shape rather than directly with the hand or fingers. A haptic interaction method for a virtual hand was presented for grasping dynamic objects and physical modeling of plasticity [7].

More sophisticated work which uses haptics to browse and edit abstract representations of animation

*This research supported in part by NSF CAREER Award CCR-9624315, NSF Grants IIS-9619850, EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017.

trajectories is given in [4]. This approach uses a vector field method to allow the user to manipulate motion-captured data.

3 I-GMS System Overview

3.1 Object-Oriented Framework

I-GMS is implemented in the C++ programming language and is comprised of object classes representing geometric entities in the virtual environment: *Environment*, *MultiBody*, *Body*, *FixedBody*, and *FreeBody*. It also contains abstract classes: *Transformation*, *Orientation*, *DHparameters*, *Connection*, etc. Each geometric class contains its own kinematic and dynamic functions as core member functions. Abstract classes support the geometric classes in that they characterize the connections among the component bodies and determine their positions and orientations via appropriate kinematic linkages.

Common functions such as kinematics and dynamics for different multi-rigid-body systems are handled internally through virtual functions. An application programmer needs to specify only high-level functions such as *ComputeKinematics* and *ComputeDynamics* in the driver routine. The underlying *Body* class propagates its basic properties to its derived classes (e.g., *FreeBody* and *FixedBody*). This is illustrated in Figure 1. Other classes can be derived from each of these, such as *attFixedBody* and *attFreeBody*, respectively, where *att* is shorthand for 'attributed'. For our dynamic simulation purposes, they are named *RigidDynFixedBody* and *RigidDynFreeBody* to indicate that all the bodies are used for rigid-body dynamic simulation. A more detailed description of the design of I-GMS can be found in [10].

3.2 Interactive Simulation via Haptics

I-GMS supports interactive simulation via haptic interaction. Through real-time user interaction, we are able to modify an existing path or generate an arbitrary trajectory during simulation. Generating a trajectory can be a tedious off-line job if the code must be modified every time we need a modified (or new) trajectory for a robot to follow. With interactive simulation, we can adjust or create trajectories during the simulation. The PHANToM haptic device [6] is used as a means for achieving interactive simulation in two modes in I-GMS: push and pull operations.

A push operation occurs at the point of contact between the PHANToM and the virtual object, which triggers the contact force at the contact point and is incorporated as an external disturbance into the forward dynamics (see Equation (2) in Section 4.2). In

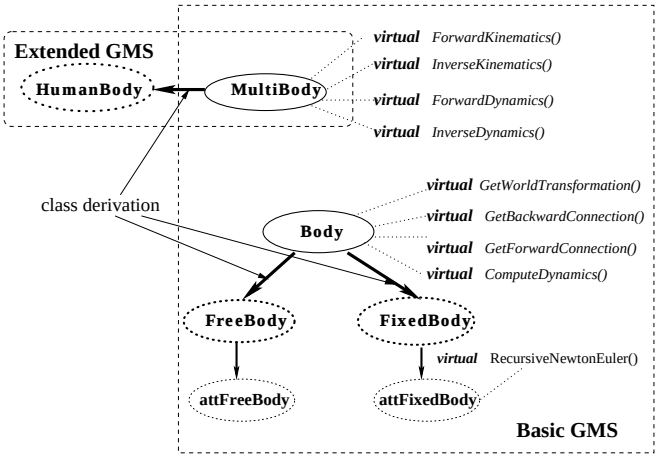

Figure 1: Class hierarchy within the I-GMS

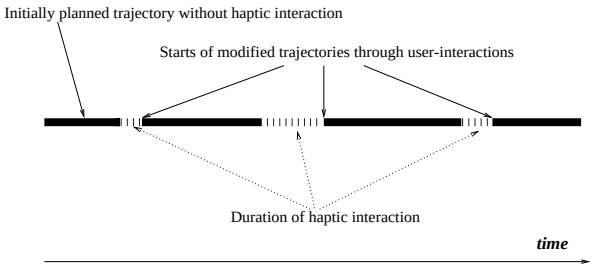

Figure 2: Interactive simulation as a sequence of interleaved operations with GMS

this way, a new acceleration is computed whenever haptic interaction occurs. This new acceleration determines the new starting state of the system from which trajectory generation is resumed (or integration is performed using the new acceleration) and continued until the occurrence of the next haptic interaction event. The change in the trajectory after the haptic touch occurs in real-time.

A pull operation occurs by attaching the PHANToM to the body and allowing the user to drag it around the workspace. For example, the PHANToM can be attached to the end-effector of an articulated structure in the workspace so that the joint motion can be followed dynamically as the user intends. Since we attach the PHANToM to the end-effector, the user is also able to feel the dragging force which corresponds to the dynamic motion of the robot. Since the operation occurs in the Cartesian space, this operation allows a more intuitive interaction for the user. This is explained in detail in Section 4.2.

Since a user usually performs haptic interaction in a sporadic manner, computations of the new system state and the ensuing trajectory generation are repeated in an interleaved fashion during simulation. This situation is illustrated in Figure 2. This simple scheme of modifying a trajectory in real-time can be

incorporated into the usual simulation steps as follows:

Simulation Steps for Interactive Simulation

1. **WHILE** (*not stopped*) **DO**
2. **IF** *there is a haptic input* (push/pull)
3. **THEN**
4. Compute joint accelerations using equation (2) or equation (5);
5. **ELSE**
6. Compute joint trajectory to follow;
7. Compute the joint torques using inverse dynamics;
8. Compute joint accelerations using forward dynamics;
9. Update system's state;

4 Dynamic Models in I-GMS

Our objective is to deal with haptic interaction for articulated structures. We achieve this by incorporating the point contact as an external force into the exact dynamics of the articulated structures.

4.1 Multi-Branch Articulated Structure with a Floating Base

An articulated structure with a multi-branch linkage and a floating-base can be used to model very complex structures such as a human body model. The recursive Newton-Euler dynamics algorithm [3] (used for the fixed-base case) has been extended to describe the dynamics.

Here, the base is considered as a free-falling body in deriving the equations. Thus, we have attached a moving frame to the base, which results in an additional 6-dof for representing its position and orientation. Thus, we add the following equations to the *outward iteration* of the fixed-base case, so that the positional and angular acceleration of every link is propagated starting from the moving base link. (The notation is adopted from [3].)

$$\begin{aligned} {}^1\dot{v}_{C_1} &= {}^1R_0(\dot{v}_{C_0} + g) \\ {}^1\omega_1 &= {}^1R_0\,\omega_{C_0} \\ {}^1\dot{\omega}_1 &= {}^1R_0\,\dot{\omega}_{C_0} \end{aligned}$$

Here, $\dot{v}_{C_0}$, ω_{C_0}, and $\dot{\omega}_{C_0}$ are the linear acceleration, angular velocity, and angular acceleration of the center of mass of the base body, respectively.

Outward iterations: $i : 1 \rightarrow n-1$

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}R_i\,{}^{i+1}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\ {}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}R_i\,{}^i\dot{\omega}_i + {}^{i+1}R_i\,{}^i\dot{\omega}_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \\ &\quad \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\ {}^{i+1}\dot{v}_{i+1} &= {}^{i+1}R_i({}^i\dot{\omega}_i \times {}^iP_{i+1} + \\ &\quad {}^i\omega_i \times ({}^i\omega_i \times {}^iP_{i+1}) + {}^i\dot{v}_i) \\ {}^{i+1}\dot{v}_{C_{i+1}} &= {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} + \\ &\quad {}^{i+1}\omega_i \times ({}^i\omega_i \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1} \\ {}^{i+1}F_{i+1} &= m_{i+1}\,{}^{i+1}\dot{v}_{C_{i+1}} \\ {}^{i+1}N_{i+1} &= {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + \\ &\quad {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1} \end{aligned}$$

Multiple-branch linkage connections are taken into consideration during the *inward iteration* as follows:

Inward iterations: $i : n \rightarrow 1$

$$\begin{aligned} {}^if_i &= \sum_{\mathbf{j}\in\mathbf{O_i}} {}^{\mathbf{i}}\mathbf{R_j}\,{}^{\mathbf{j}}\mathbf{f_j} + {}^{\mathbf{i}}\mathbf{R_o}\sum_{\mathbf{j}\in\mathbf{O_i}} {}^{\mathbf{o}}\mathbf{f_{E_j}} + {}^iF_i \\ {}^in_i &= \sum_{\mathbf{j}\in\mathbf{O_i}} {}^{\mathbf{i}}\mathbf{R_j}\,{}^{\mathbf{j}}\mathbf{n_j} + {}^iP_{i+1} \times {}^iR_j\,{}^jf_j \\ &\quad - \sum_{\mathbf{j}\in\mathbf{O_i}} {}^{\mathbf{j}}\mathbf{c_j} \times {}^{\mathbf{o}}\mathbf{f_{E_j}} + {}^iN_i + {}^iP_{C_i} \times {}^iF_i \\ \tau_i &= {}^in_i^T\,{}^i\hat{Z}_i \end{aligned}$$

Here, ${}^iP_j$, ${}^iR_j$, and ${}^iP_{C_i}$ refer to the position, orientation of the j^{th} body, and the center of mass of the i^{th} body in the i^{th} body frame, and the center of mass of the i^{th} body respectively, i refers to the set of all indices of branching links of the i^{th} link body, f_{E_j} refers to j^{th} external force in the set of all external forces (M_j) acting on the link indexed by i, and O_i refers to the set of indices corresponding to all the branch-outs from the i^{th} body. The boldfaced terms account for the effects due to multi-branch links on the incident links and external forces acting on a link, respectively.

To write all these equations in a state-space representation, we introduce the following notation:

$$\begin{aligned} X^T &= [p_B^T, A_B^T, \Theta^T] \in R^3 \times SO(3) \times R^N \\ V^T &= [v_B^T, \omega_B^T, \omega^T] \in R^3 \times R^3 \times R^N \\ U^T &= [f_B^T, n_B^T, \tau^T] \in R^3 \times R^3 \times R^N \end{aligned}$$

where

p_B	:	(3×1) vector specifying base-link position
A_B	:	(3×3) matrix specifying base-link attitude
Θ	:	$(N \times 1)$ vector specifying joint angle
v_B	:	(3×1) vector specifying base-link velocity
ω_B	:	(3×1) vector specifying angular velocity of base-link
ω	:	$(N \times 1)$ vector specifying joint angular velocity
f_B	:	(3×1) force vector acting on base-link
n_B	:	(3×1) torque vector acting on base-link
τ	:	$(N \times 1)$ torque vector acting on joints
N	:	number of joints in the system

Recall that we have extended the system's state vector with an additional 6-dof for representing the position and orientation of the base. Thus, the state-space representation of the dynamics is:

$$H(X)\dot{V} - C(X,V)V + G(X) = U - U_E, \qquad (1)$$

where

$H(X)$: $(N+6) \times (N+6)$ inertia matrix

$C(X,V)$: $(N+6)\times(N+6)$ matrix specifying centrifugal and Coriolis's effects

$G(X)$: $(N+6)\times 1$ vector specifying gravity effect

U_E : $(N+6)\times 1$ vector specifying generalized force generated by external forces

Note that the above extended recursive Newton-Euler equations naturally accommodate the traditional single-branch articulated bodies such as robot manipulators.

4.2 Dynamics with Haptic Interaction

Push operation:
For the case of push operation, I-GMS considers the haptic interaction on a multi-rigid-body system as an external force applied to it by the user, acting at a contact point on the body surface. For instance, haptic touch on a robot manipulator is regarded as an external contact force (by the haptic device) acting on it, which leads to a modification of the forward dynamic equation (derived from Equation (1)):

$$\dot{\mathbf{V}} = \mathbf{H^{-1}}(\mathbf{X})[\mathbf{U} - \mathbf{U_E} + \mathbf{C}(\mathbf{X},\mathbf{V}) - \mathbf{G}(\mathbf{X})] \quad (2)$$

U_E accounts for the joint torque vector corresponding to the contact force, c, due to collision as follows:

$$U_E = J^T c, \quad c = k_p\, d_{penetration} \quad (3)$$

This induces accelerations on the system in response to the haptic touch. The contact force at the contact point due to the haptic interaction is computed by a lumped spring model, where k_p is the position gain and $d_{penetration}$ is the penetration distance between the haptic device and the virtual object.

Pull operation:
For the pull operation, I-GMS uses an impedance controller approach introduced by [5]. The impedance controller calculates the force f from the virtual spring and damper. In particular, the virtual force f is computed by attaching a virtual spring and damper from the end-effector position (X_{endeff}) to the PHANToM position (X_{ph}), as in Equation (4).

$$F = k(X_{ph} - X_{endeff}) - b(\dot{X}_{ph} - \dot{X}_{endeff}) \quad (4)$$

where X_{endeff} and X_{ph} are 6-D vectors defining the actual and desired position/orientation of the end-effector in the cartesian space, and $\dot{X}_{endeff}$ and $\dot{X}_{endeff}$ are 6-D vectors representing the actual and desired velocities of the positions/orientations of the end-effector, respectively. Also, k and b are stiffness and damping matrices, respectively. These last two tunable parameters affect the sense of contact the operator feels through the haptic device. Then, the desired force is produced by applying torque τ at the joints, which are calculated using the Jacobian $J(\theta_{endeff})$ as in the following relation:

$$\tau = J(\theta_{endeff})^T F \quad (5)$$

This τ in turn is fed into Equation (1) to compute the corresponding joint motions for the articulated structure.

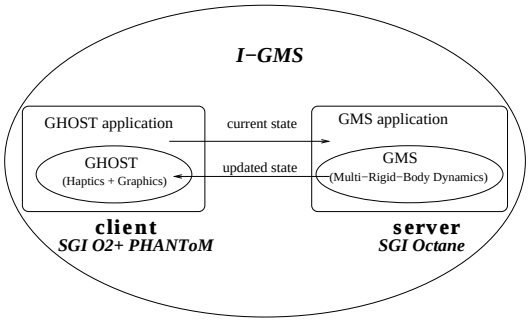

Figure 3: Overall system architecture of I-GMS

5 Integration of I-GMS with PHANToM

Our prototype hardware system for performing haptic interaction consists of a 3-dof PHANTOM haptic device [6], an SGI O2 graphics workstation (graphics display) and an SGI Octane (dynamic computation server). The graphics keeps track of the position updates of the PHANTOM finger tip. The PHANTOM generates force-feedback using collision/penetration information between the finger tip and the body. The operator can use the PHANTOM to touch a rigid object in the virtual scene. Currently, we have integrated I-GMS's manipulator dynamics into our haptic-interaction application which was developed using the C++ *General Haptic Open Software Toolkit* (GHOST SDK) [9]. Both haptic and dynamic computations occur in the same servo cycle to enable us to reflect the appropriate I-GMS state change within the GHOST application. The overall system architecture is depicted in Figure 3.

To achieve realistic feedback of approximately 1 kHz frequency, we have used two techniques: one is the distribution of computations over the network and the other is the use of an interpolation of the system's state between network relays. For the distribution of computations, we have divided the two major tasks (haptics and dynamics) into separate processors using socket programming over the UDP/IP layer on the Ethernet. The UDP protocol is a connectionless client/server communication mechanism, which facilitates faster transmission of data than TCP. But this is

Figure 4: 6-dof robot manipulator and a wall

with less reliable transmission of data packets. However, for us, transmitting data at a faster servo rate is more crucial than the possible minimal loss of data. To maintain a high servo rate, the client (haptic computation) uses the results computed at a previous time cycle if the remote server does not return the dynamics results after a certain pre-set time. The preset time interval is adjustable within I-GMS; setting it close to 1 *ms* gave reasonable haptic interaction in our experiments.

6 Simulation Examples

We have demonstrated interactive simulations on a 6-dof robot manipulator through on-line editing of pre-planned trajectories.

6.1 Push for 6-dof robot manipulator

We consider a simple scenario where a 6-dof robot manipulator (see Figure 4) is supposed to follow a straight-line trajectory from its starting point until it reaches a wall. An obstacle is introduced in the way of the pre-planned trajectory, which is shown in Figure 5. The user is supposed to use visual cues to edit the pre-planned trajectory using the haptic interaction push mode to avoid the collision. The resulting trajectory is a path modified by the change in dynamic motion of the manipulator via haptic touch. Here, we have tried to push the second link of the manipulator away from the obstacle, since it was touching the obstacle while nominally following the pre-planned (straight-line) trajectory.

The initial steps in Figure 6 (top) show the portion of the original pre-planned trajectory. This lasts until there is the first haptic touch by the user, which is indicated by the force calculation at step 11 in Figure 6 (bottom). Then modified trajectories resulting from real-time haptic interaction by the user are followed. The forces computed by haptic touches are also given in Figure 6 (bottom). Note that there is

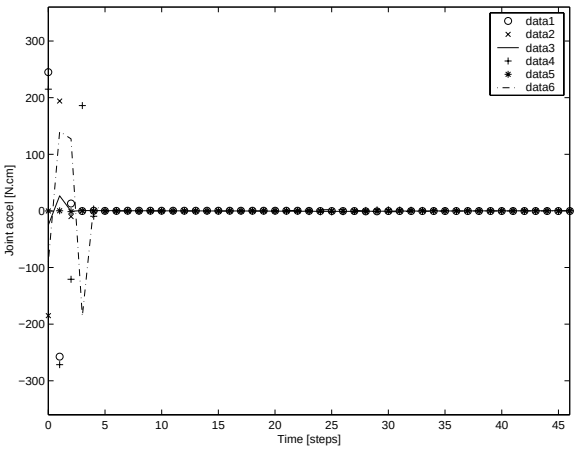

Figure 5: The nominal trajectory

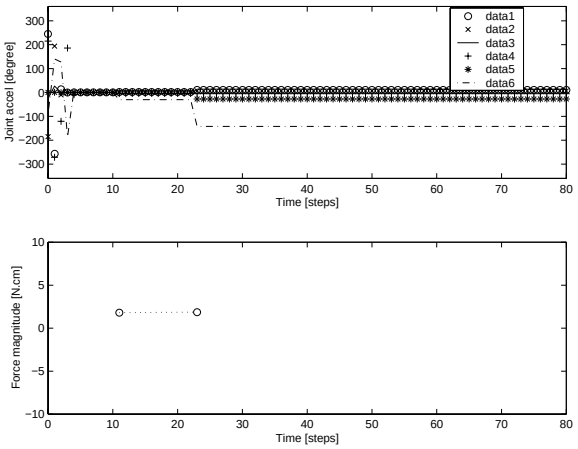

Figure 6: The trajectory disturbed by push operation

some instantaneous change of accelerations due to the user's haptic interaction. These changes in accelerations correspond to the starts of new states to be used for subsequent dynamic update (refer to Section 3.2) during the interleaved operations. Once a haptic interaction occurs, the joint accelerations are maintained until subsequent user interaction, which is evident in the plot. The external forces acting at the contact points are computed by Equation (3) at time steps 11 and 23 (the magnitude unit is $N.cm$).

This example shows that just a few haptic pushes at appropriate points on the manipulator bodies could change the pre-planned trajectory to avoid colliding with an obstacle.

6.2 Pull for 6-dof robot manipulator

We also performed pull operations on a 6-dof robot manipulator. We used the same scenario as in the

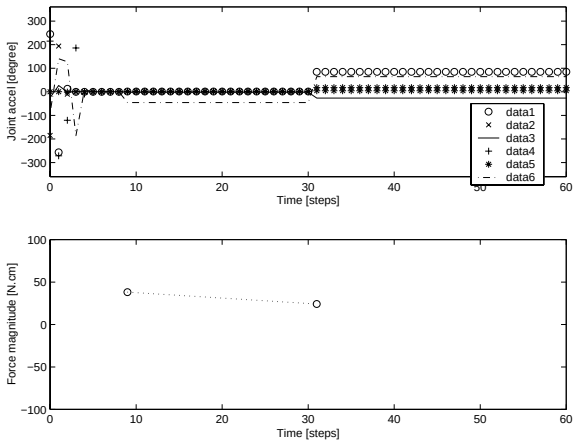

Figure 7: The trajectory disturbed by pulling operation

push mode example. This time, to avoid the collision, the user dragged the end-effector around the obstacle, which required some care to ensure that the second link was not placed in collision.

The plots in Figure 7 show the same information as in the push mode example. The difference here is that the force is computed by a virtual spring connecting the PHANToM and the robot's end effector, as opposed to the contact effect for the push mode.

We note that relatively greater forces are required for the dragging operation than the push operation and this is considered reasonable. Generally, we observed that it was easier for the user to use pull mode than the push mode to change trajectories.

7 Discussion and Future Work

The user-interaction capability of I-GMS allows the user to adjust the behaviors of articulated structures in real-time. This shows promise for user-interaction of fairly complex articulated structures, too, once performance issues can be resolved to ensure stable haptic interaction.

Currently, we are working on incorporating haptic interaction for free bodies in contact with an environment. For example, we can consider a ball rolling and sliding on a flat surface. We regard the contact where the haptics occurs as the primary one and the one between the ball and the surface as secondary. Our goal is to implement correct haptic interaction even in the presence of the secondary contact. This is a complicated problem which requires exact contact mechanics to predict physically-correct contact mode at the secondary contact whose effect is in turn propagated to the primary contact for the appropriate haptic interaction. This exact haptic interaction will support more sophisticated user interaction in general simulation environments which include free bodies in contact as well as articulated structures.

References

[1] Peter J. Berkelman, Ralph L. Hollis, and David Baraff. Interaction with a realtime dynamic environment simulation using a magnetic levitation haptic interface device. In *Proceedings of IEEE International Conference on Robotics and Automation*, Detroit, Michigan, May 1999.

[2] B. Chang and J. E. Colgate. Real-time impulse-based simulation of rigid body system for haptic display. *Proc. Symp. on Interactive 3D Graphics*, pages 200–209, April 1997.

[3] J. J. Craig. Introduction to robotics, mechanics, and control. 1986.

[4] Bruce Randall Donald and Frederick Henle. Using haptic vector fields for animation motion control. In *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, CA, May 2000.

[5] N. Hogan. Impedance control: An approach to manipulation. *Journal of Dynamics Systems, Measurement, and Control*, 107:1–24, 1985.

[6] T. H. Massie and J. K. Salisbury. The PHANToM haptic interface: A device for probing virtual objects. In *Int. Mechanical Engineering Exposition and Congress, DSC 55-1*, pages 295–302, Chicago, 1994. C. J. Radcliffe, ed., ASME.

[7] V. Popescu, G. Burdea, and M. Bouzit. Virtual reality simulation modeling for a haptic glove. In *Proceedings of Computer Animation 1999*, Geneva, Switzland, May 1999.

[8] Diego C. Ruspini, Krasinir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *Computer Graphics Proceedings*, Los Angeles, CA, July 1997.

[9] Sensable Technologies, Inc. *GHOST Software Developer's Toolkit Programmer's Guide Version 1.21.*

[10] Wookho Son, Kyunghwan Kim, and Nancy M. Amato. An interactive generalized motion simulator (i-gms) in an object-oriented framework. In *Proceedings of Computer Animation 2000*, Philadelphia, Pennsylvania, May 2000.

[11] S. Vedula and D. Baraff. Force feedback in interactive dynamic simulation. In *Proceedings of the First PHANToM User's Group Workshop*, Deaham, MA., September 1996.

[12] C. Zilles and K. Salisbury. A constraint-based god-object method for haptic display. In *IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 146–151, Pittsburgh, PA, 1985.