Interactive Dynamic Simulation Schemes for Articulated Bodies through Haptic Interface

Wookho Son, Kyunghwan Kim, Byungtae Jang, and Byungtae Choi

This paper describes interactive dynamic simulation schemes for articulated bodies in virtual environments, where user interaction is allowed through a haptic interface. We incorporated these schemes into our dynamic simulator I-GMS, which was 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 with the PHANToM haptic device, which runs as an integrated part of I-GMS. We use both forward and inverse dynamics of articulated bodies for the haptic interaction by the push and pull operations, respectively. We demonstrate the user-interaction capability of I-GMS through on-line editing of trajectories for 6-dof (degrees of freedom) articulated bodies.

Manuscript received Mar. 12, 2002; revised Oct. 28, 2002.

Wookho Son (phone: +82 42 860 5031, e-mail: whson@etri.re.kr) and Byungtae Jang (e-mail: jbt(@etri.re.kr) are with Augmented Reality Research Team, ETRI, Daejeon, Korea.

Byungtae Choi (e-mail: btchoi@etri.re.kr) is with 3D Graphics Research Team, ETRI, Daejeon, Korea.

Kyunghwan Kim (e-mail: kimk@wooshin-m.com) is with Wooshin Mechatronics Co., Ltd., Seoul, Korea.

I. INTRODUCTION

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 that are actuated. Dynamic simulation advances over time steps by successively predicting the accelerations (and contact forces of any rigid bodies that are in contact) of the multi-rigid-body systems in the environment.

Dynamic motion simulation arises in many engineering application domains, such as virtual reality, graphics, robot motion simulators, and computer games. For example, interaction with the virtual environment by the user in real time is becoming increasingly important in computer games. This interaction is achieved not only through the user's textual input, but also by 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 from partial occlusion of manipulated objects during simulation or when the environment is dark.

Adding haptic interaction to the dynamic simulation has the effect of exerting user-applied external forces to the active bodies in the virtual 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 and virtual training.

In this work, we present two methods of realizing interactive dynamic simulation through haptic feedback: *push* and *pull* modes. We performed the interactive simulation in our test-bed

dynamic simulator I-GMS [1], which can simulate dynamic motions of multi-rigid-bodies in virtual environments. In particular, the user interaction is demonstrated by on-line editing and modification of trajectories of articulated bodies in applications, such as robot manipulators.

This paper is organized as follows. In section II, we introduce related works. The dynamics simulation system that incorporates our interactive schemes is described in section III. The dynamic models we used for the haptic interaction are discussed in section IV, followed by simulation examples that demonstrate our interactive schemes in section V. Finally, we give concluding remarks in section VI.

II. RELATED WORK

Early work on haptic interaction focused on haptic rendering of graphical environments and used force feedback coupled with the visual display to realize surface shading, friction, and texture [2]-[4]. Some proposed dynamic simulators have haptic interaction capability. One study used an impulse-based simulation as a general purpose multi-body simulator for haptic display [5], and another investigated the haptic interaction for a point contact for rigid body dynamics [6]. Berkelman et al. [7] provided 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. One haptic interaction method used a virtual hand for grasping dynamic objects and physical modeling of plasticity [8]. All these studies dealt with haptic interaction for non-linked free bodies.

Mataric [9] used a control method for a haptic pull operation similar to ours for performing a continuous sequence of smooth movements: a physics-based humanoid torso dancing the Macarena. However, that method implemented conventional user interaction rather than haptics.

Another scheme similar to our push operation used a simplified forward dynamics of multi-chained articulated figures, but it did not present any real-time simulation example that used haptic interaction [10]. Our work not only provides an interactive simulation scheme using a full forward dynamics but also provides a haptic push operation by using the inverse dynamics of multi-chained articulated bodies.

Donald and Henle [11] offered a more sophisticated model that used haptics to browse and edit abstract representations of animation trajectories. This approach used a vector field method to allow the user to manipulate motion-captured data. It was innovative in that it tried to manipulate high degrees of freedom (dof) animation characters through a low dof control space by using the PHANToM haptic device. While it provided manipulation of animation characters, it was through unintuitive indirect touch rather than direct touch on the animation character's body.

III. SYSTEM OVERVIEW

Our interactive simulation schemes were systematically incorporated into our dynamic simulator, called the I-GMS, to include the haptic interaction as an external disturbance into the dynamics of the articulated structures. In fact, the incorporation process itself self-verifies that our way of including the interactive schemes into the pre-developed dynamics is theoretically sound and correct. Thus, it is important to understand the underlying dynamics of our backbone simulator to understand subsequent development of haptic interaction.

1. Object-Oriented Framework of the I-GMS

The 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 auxiliary classes: *Transformation, Orientation, DHparameters, Connection*, etc. Each geometric class contains its own kinematic and dynamic functions as core member functions. Auxiliary classes support the geometric classes by characterizing the connections among the component bodies and determining their positions and orientations via appropriate kinematic propagations.

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 to specify the motions of a multi-rigid-body. The underlying *Body* class propagates its basic properties to its derived classes (e.g., *FreeBody* and *FixedBody*). This is illustrated in Fig. 1. Other classes can be derived from each of these, for example, *attFixedBody* and *attFreeBody*, where *att* is a shorthand for 'attributed.' For our simulation, we used *RigidDynFixedBody* and *RigidDynFreeBody* to indicate that all the bodies are used for rigid-body dynamic simulation.

In the I-GMS, a geometric class *MultiBody* is instantiated to represent any articulated structure, such as a robot manipulator or human body model. Thus, it is easy to incorporate the haptic interaction into the dynamic simulation, since we only need to modify the dynamics that are implemented as a member function of the *MultiBody* class. A more detailed description of the design of the I-GMS can be found in [1].

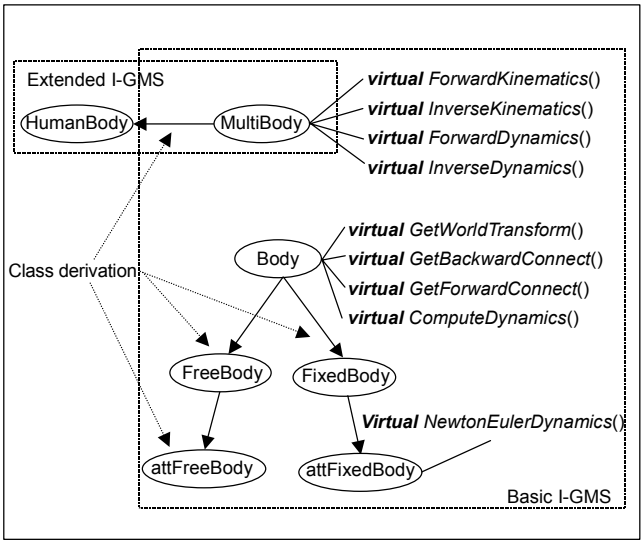


Fig. 1. Class hierarchy within the I-GMS.

2. Interactive Simulation through Haptics Interaction

We support interactive simulation through 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 manipulator to follow. With interactive simulation, we can adjust or create trajectories during the simulation. In particular, the user interaction is focused on the on-line editing and creation of trajectories for articulated manipulators. The PHANToM haptic device [12] achieves interactive simulation in two modes in our simulation: push and pull operations.

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

In a pull operation, the PHANToM is attached to an articulated object, and the user can 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 Cartesian space, this operation allows a more intuitive interaction for the user. This is explained in detail in section IV. 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 Fig. 2.

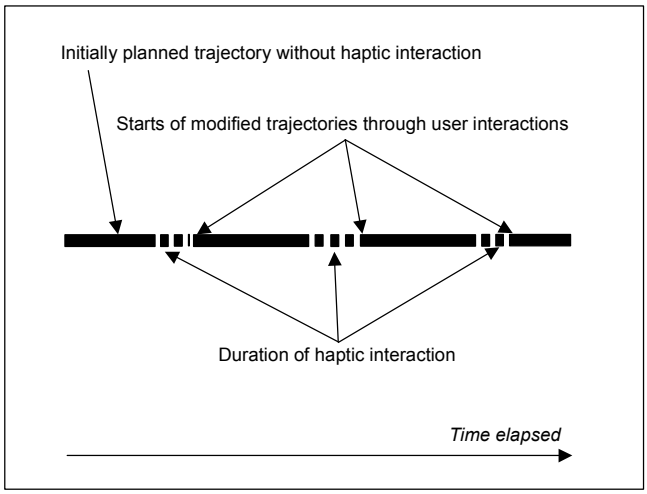


Fig. 2. Interactive simulation as a sequence of interleaved operations.

This simple scheme of modifying a trajectory in real time can be incorporated into the usual simulation steps as follows:

Steps for Interactive Simulation

```
WHILE (not stopped) DO
  IF (there is a haptic input)   (push or pull)
    THEN
      Compute joint accelerations using forward or inverse dynamic equation;
    ELSE
      Compute joint trajectory to follow;
      Compute the joint torques using inverse dynamics;
      Compute joint accelerations using forward dynamics;
      Update system's state.
```

IV. DYNAMIC MODELS

Our objective is to deal with haptic interaction for articulated structures by considering this interaction as a change in the dynamic behavior of the underlying structures. We achieve this by incorporating the contact point as an external force into the exact dynamics of the articulated structures. Thus, it is crucial to describe the dynamics of the articulated structures in detail for further development of the haptic inclusion into the dynamic simulator. For our investigation, we describe the

dynamics supported in the I-GMS by motion equations for a generalized articulated structure, while explaining the inclusion of the haptic interaction as an external disturbance that changes the underlying dynamics.

1. Multi-Branch 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. To describe the dynamics, we extended the recursive *Newton-Euler* dynamics algorithm [13] (used for the fixed-base case).

Here, the base is considered as a free-falling body in deriving the equations. Thus, we attached a moving frame to the base, which resulted in an additional 6-dof for representing its position and orientation. To consider the moveable base, we added 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 was adopted from [13].)

Outward iterations

For base case,

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

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.

For i from 1 to $n-1$,

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

Inward iterations

For i from n down to 1

Multiple-branch linkage connections are taken into consideration during the *inward iteration.*

$$
\begin{aligned}
{}^{i}f_i &= \sum_{j\in O_i} {}^{i}R_j\,{}^{j}f_j + {}^{i}R_0 \sum_{j\in O_i} {}^{0}f_{E_j} + {}^{i}F_i\\
{}^{i}n_i &= \sum \left[{}^{i}R_j\,{}^{j}n_j + {}^{i}P_{i+1} \times {}^{i}R_j\,{}^{j}f_j\right]\\
&\quad - \sum {}^{j}c_j \times {}^{0}f_{E_j} + {}^{i}N_i + {}^{i}P_{C_i} \times {}^{i}F_i\\
\tau_i &= {}^{i}n_i^T\,{}^{i}\hat{z}_i.
\end{aligned}
$$

Here, ${}^{i}P_j$, ${}^{i}R_j$, and ${}^{i}P_{C_i}$ 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. O_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. Note that these equations account for the effects that are due to both multi-branch links on the incident links and external forces acting on a link.

The force and moment acting at the floating base of the multi-linkage structure are defined as:

$$
\begin{aligned}
f_B &= {}^{o}R_B\,{}^{1}f_1\\
n_B &= {}^{o}R_B\,{}^{1}n_1,
\end{aligned}
$$

where ${}^{0}R_B$ is the orientation of the base in the inertial frame, and ${}^{1}f_1$ and ${}^{1}n_1$ are the force and moment at the base in the bodyfixed local frame.

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

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

where

p_B	(3×1) vector specifying base link position
A_B	$(3\times3)\times1$ matrix specifying base link attitude
Θ	$(N\times1)$ 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	$(N\times1)$ vector specifying angular joint velocity
f_B	(3×1) force vector acting on base link
n_B	(3×1) torque vector acting on base link
τ	$(N\times1)$ torque vector acting on joints
N	number of joints in the system.

Recall that we 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:

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

where

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

$G(X)$ $(N+6)\times 1$ gravity effects

U_E $(N+6)\times 1$ *s*pecifying generalized force generated by external forces.

2. Dynamics for Haptic Interaction

A. Push operation

For the push operation, we consider 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 (1)):

$$\dot{V} = H^{-1}(X)(U_E - C(X,V) - G(X)). \quad (2)$$

U_E denotes 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.

B. Pull operation

For the pull operation, we used the impedance controller approach introduced by [14]. 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 (4).

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

where X_{endeff} and X_{ph} are 6-D vectors defining the actual and desired position/orientation of the end-effector in Cartesian space, and $\dot{X}_{endeff}$ and $\dot{X}_{ph}$ are 6-D vectors representing the actual and desired velocities of the positions/orientations of the end-effector, respectively. In addition, *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:

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

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

3. Integration of I-GMS with PHANToM

Our prototype hardware system for performing haptic interaction consists of a 3-dof PHANToM haptic device [12], 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.We integrated I-GMS's manipulator dynamics into our haptic-interaction application which was developed using the C++ *General Haptic Open Software Toolkit* (*GHOST SDK*) [15]. 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 Fig. 3.

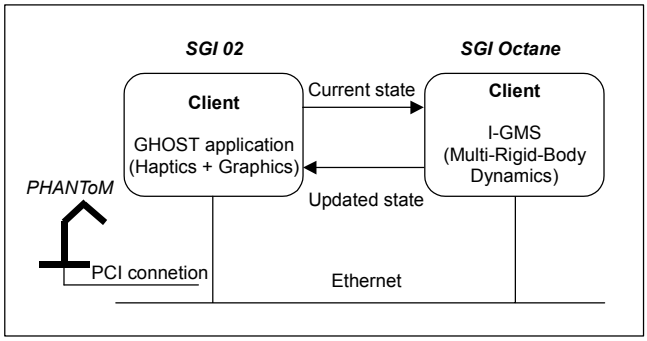


Fig. 3. Overall system architecture for our interactive dynamic simulator.

To achieve realistic feedback of approximately 1 kHz frequency, we 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 divided the two major tasks (haptics and dynamics) into separate processors using socket programming over the UDP/IP layer on the Ethernet. The UDP is a connectionless client/server communication mechanism that transmits data faster than the TCP, but this results in less reliable transmission of data packets. However, for us, transmitting data at a faster servo rate is more crucial than a 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 preset time. The preset time interval is adjustable within I-GMS; setting it close to 1 millisecond gave reasonable haptic interaction in our experiments.

V. SIMULATION RESULT

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

1. Push for 6-dof Robot Manipulator

We consider a simple scenario where a 6-dof robot manipulator (Fig. 4) is supposed to follow a straight-line trajectory from its starting point until it reaches a wall. When there is no obstacle between the robot and the wall, as in Fig. 4(a), it is not very hard to plan the trajectory and have the robot follow it. A straight-line trajectory is given in Fig. 5. However, when an obstacle is introduced in the way of the pre-planned trajectory, as shown in Fig. 4(b), it is hard to find a collision-free trajectory for the robot. The real problem is that this change in the environment could happen dynamically, thus requiring replanning every time there is a change.

An interactive way of modifying an existing trajectory is an efficient way of avoiding costly preprocessing. In our scheme for modifying the trajectory, the user uses visual cues to edit the pre-planned trajectory using the haptic interaction push mode to avoid a collision. The resulting trajectory is a path modified by the change in dynamic motion of the manipulator using haptic touch. Here, we pushed 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 time steps in Fig. 6(a) show a 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

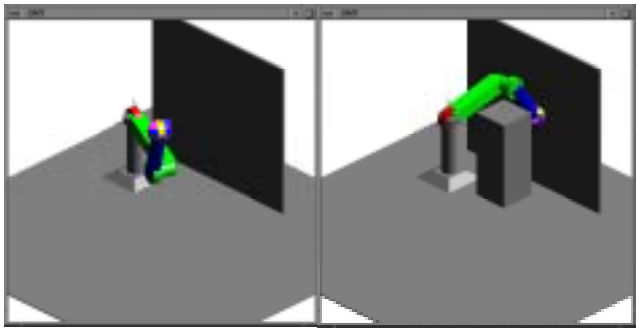


Fig. 4. (a) 6-dof robot manipulator and a wall, (b) 6-dof robot with an additional obstacle.

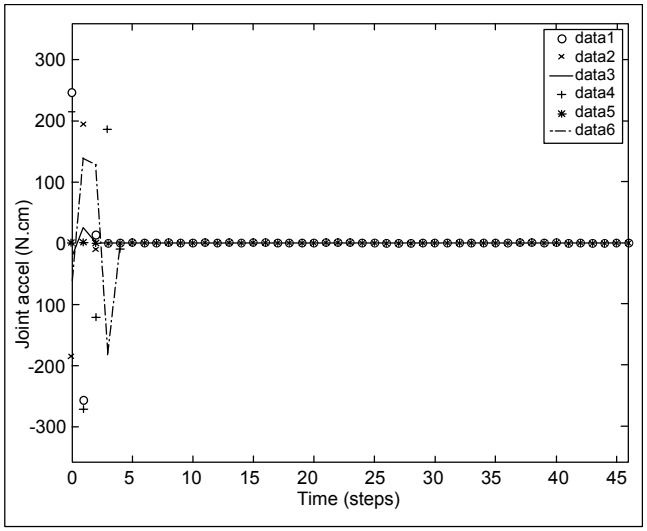


Fig. 5. The straight line trajectory.

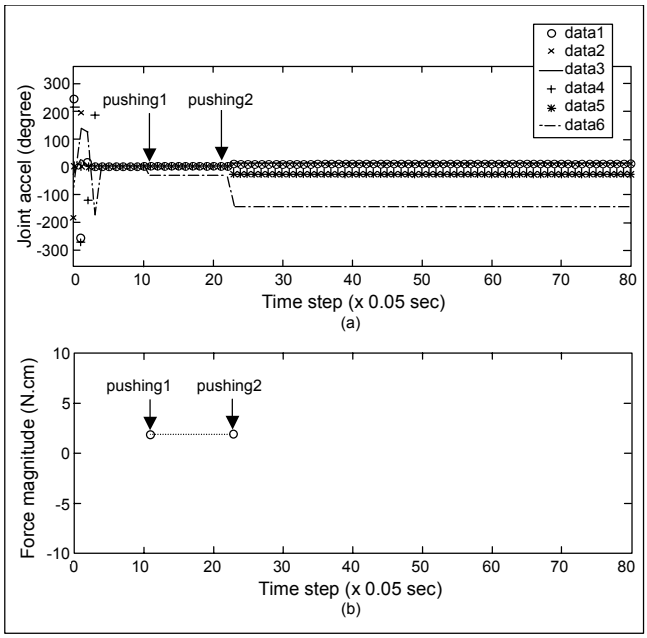


Fig. 6. (a) The trajectory disturbed by a push operation, (b) Forces computed by haptic touches

calculation at time step 11 in Fig. 6(b). Modified trajectories resulting from real-time haptic interaction by the user are then followed. The forces computed by haptic touches are also given in Fig. 6 (b). Note that there is 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 Fig. 2) during the interleaved operations. In fact, the first such change ('pushing1') is for the robot to avoid the first collision with the obstacle, while the second ('pushing2') is to direct the robot towards the wall as the original intention of the trajectory does. Once 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 (3) at time steps 11 and 23 (the magnitude unit is N.cm). Also, here joint1, jount2, etc. indicate the usual joints in the robot manipulator (Fig. 7). In fact, there are six joints available in the robot manipulator provided as an example.

The simulation snapshots of the modified trajectory are given in Fig. 7. This shows the 6-dof manipulator taking a detour around the obstacle to avoid it rather than taking the original straight-line trajectory. This example shows that just a few haptic pushes at appropriate points on the manipulator bodies can change the pre-planned trajectory to avoid colliding with an obstacle.

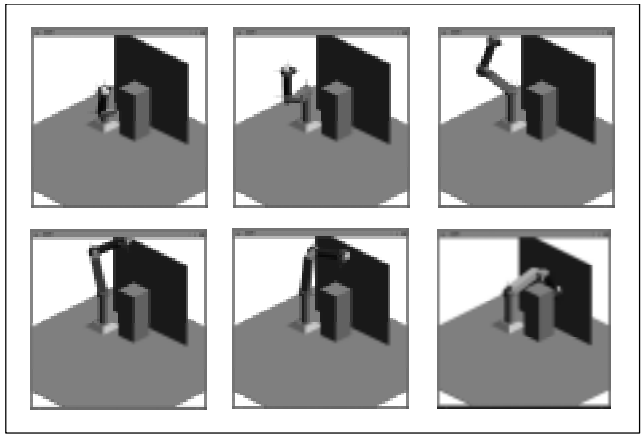


Fig. 7. Simulation snapshots of the 6-dof robot manipulator following the modified trajectory.

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 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 would not encounter a collision.

The plots in Fig. 8 show the same information as in the push mode example. In other words, the first pulling ('pulling1') is for the robot to avoid the first collision with the obstacle, while the second ('pulling2') is to direct the robot towards the wall as the original intention of the trajectory does. The actual 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.

Generally, we observed that it was easier for the user to use the pulling mode than the pushing mode to change trajectories. However, there are some trade-offs in terms of advantages and disadvantages in using these two modes of operation for trajectory modifications. We found that relatively greater forces are required for the dragging operation than the push operation and this is natural. This is because the articulated object's end-effector is supposed to follow the user's finger tip position (which is basically the position of haptic touch). On the other hand, it is harder for the user to guide the articulated bodies towards a target position, since the pushing operation is an indirect way of achieving the desired motion. Thus, the pushing operation usually takes a more sophisticated effort for the user to be able to manipulate the articulated bodies haptically, but it requires less force to move the articulated bodies.

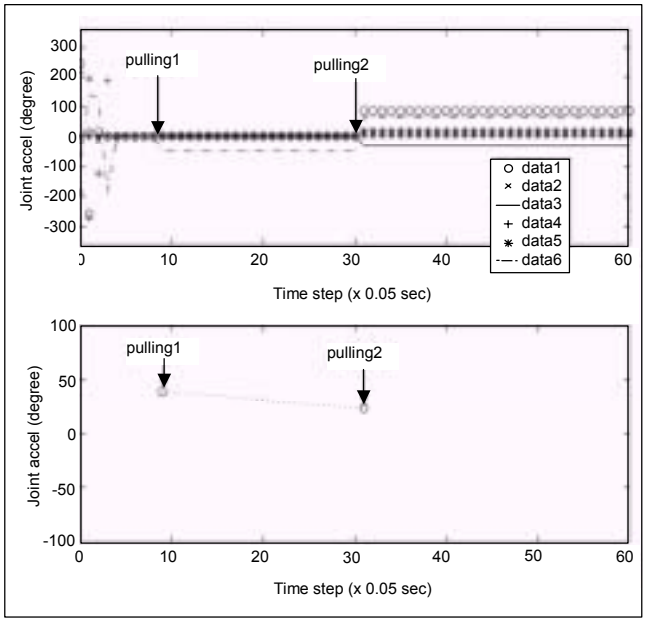


Fig. 8. The trajectory disturbed by a pull operation.

VI. DISCUSSION AND FUTURE WORK

Our interactive simulation schemes allow the user to adjust the behavior of articulated structures by using a haptic interface in real time. In particular, we developed both push and pull operations to use in the haptic interaction, using full forward and inverse dynamics of multi-linkage articulated structures, respectively. By using these schemes, the user is allowed to push or pull any part of the articulated structures and feel the force and impact created by the interaction. This shows promise for user interaction even of fairly complex articulated structures, such as legged robots, once performance issues are resolved to ensure stable haptic interaction.

Realizing a fast servo-update rate for complex articulated structures in a crowded virtual environment is a difficult problem to handle, unless some portions of the computationally expensive dynamics are relieved in one way

or another. We are devising a method to get around this problem by trying to simplify portions of the dynamic computation involved in the simulation.

We are also working on incorporating haptic interaction for free bodies in contact with an environment. For example, we can consider a ball rolling or 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 a 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 that include free bodies in contact as well as articulated structures.

REFERENCES

[1] W. Son, K.H. Kim, and N.M. Amato, "An Interactive Generalized Motion Simulator (I-GMS) in an Object-Oriented Framework," *Proc. Computer Animation*'00, 2000, pp. 176-181.

[2] C. Zilles and K. Salisbury, "A Constraint-Based God-Object Method for Haptic Display," *Proc. IEEE-Int'l Conf. on Intelligent Robots and Systems*, PA, 1995, pp. 146-151.

[3] D.C. Ruspini, K. Kolarov, and Oussama Khatib, "The Haptic Display of Complex Graphical Environments," *SIGGRAPH*, 1997, pp. 345-352.

[4] T.V. Thompson II, D.E. Johnson, and Elaine Cohen, "Direct Haptic Rendering of Sculptured Models," *Symp. on Interactive 3D Graphics*, 1997, pp. 167-176.

[5] B. Chang and J.E. Colgate, "Real-Time Impulse-Based Simulation of Rigid Body Systems for Haptic Display," *Proc. Symp. on Interactive 3D Graphics*, 1997, pp. 200-209.

[6] S. Vedula and D. Baraff, "Force Feedback in Interactive Dynamic Simulation," *Proc. the First PHANToM User's Group Workshop*'96, 1996, pp. 25-31.

[7] P.J. Berkelman, R.L. Hollis, and D. Baraff, "Interaction with a Real-Time Dynamic Environment Simulation Using a Magnetic Levitation Haptic Interface Device," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1999. pp. 3261-3266.

[8] V. Popescu, G. Burdea, and M. Bouzit, "Virtual Reality Simulation Modeling for a Haptic Glove," *Proc. Computer Animation*'99, 1999, pp. 41-47.

[9] M.J. Mataric, "Making Complex Articulated Agents Dance: An Analysis of Control Methods Drawn from Robotics, Animation, and Biology," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 1, July, 1999, pp. 23-44.

[10] Ruspini Diego and Oussama Khatib, "Dynamic Models for Haptic Rendering Systems," *Advances in Robot Kinematics: ARK*'98, Strobl/Salzburg, Austria, June 1998, pp. 523-532.

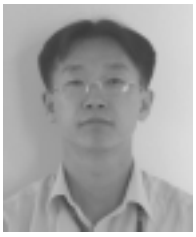
[11] B.R. Donald and F. Henle, "Using Haptic Vector Fields for Animation Motion Control," *Proc. IEEE-Int'l Conf. on Robotics and Automation*, 2000, pp. 3435-3442.

[12] T.H. Massie and J.K. Salisbury, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *Proc. of the ASME Int'l Mechanical Engineering Congress and Exhibition*, vol. DSC 55-1, IL, 1994, pp. 295-302.

[13] J.J. Craig, *Introduction to Robotics, Mechanics, and Control*, Addison-Wesley, 1986.

[14] N. Hogan, "Impedance Control," *Journal of Dynamics Systems, Measurement, and Control*, vol. 107, no. 3, Sept. 1985, pp. 1-24.

[15] SensAble Technologies Inc., *GHOST Software Developer's Toolkit: Programmer's Guide Version 2.1*, MA, 1999.



Wookho Son received the BS degree in computer science from Yonsei University in Korea in 1987 and the MS and PhD degrees in computer science from Texas A&M University in the US in 1996 and 2001. He is currently working as a Senior Research Scientist at the Virtual Reality Research Center in Electronics and Telecommunications Research Institute (ETRI) in Daejeon, Korea. His research interests include virtual reality (especially haptic interaction), robotics, and physically-based dynamic simulation.

Kyunghwan Kim received the BE degree in electrical engineering from Yonsei University in Korea in 1992 and the ME and PhD degrees in electrical engineering from the University of Tokyo in Japan in 1994 and 1997. From Sept. 1997 to Aug. 1999, he was a Postdoctoral Research Associate at the University of Wisconsin-Madison and Texas A & M University. From Sept. 1999 to June 2002, he was a Senior Research Scientist in the Korea Institute of Science and Technology (KIST) in Seoul, Korea. He is currently a Director of Wooshin Mechatronics Co., Ltd. His research interests include robotics (especially, human-computer interface), micro and nano robotics, and semiconductor handling.

Byungtae Jang received the BS degree in atmospheric science from Seoul National University in Korea in 1989 and the MS and the PhD degrees in computer science from Chungnam National University in 1994 and 2001. From 1989 to 1996, he was a Senior Research Member at Software Engineering Research Institute (SERI) in Daejeon, Korea. Since joining Electronics and Telecommunications Research Institute (ETRI) in Daejeon, Korea, in 1997, he has been working as a Principal Researcher at the Virtual Reality Research Center leading the Augmented Reality Research Group. His research interests include virtual reality (especially, augmented reality), image processing, and human-computer interaction.

Byungtae Choi received the BS degree in electronic engineering from Kyungbook National University in Korea in 1989 and the MS degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in Daejeon, Korea. From Jan. 1986 to Sept. 1988, he worked at Samsung Electronics Co., Ltd., Seoul. Since joining Electronics and Telecommunications Research Institute (ETRI), Daejeon, in 1989, he has been working as a Principal Researcher at the Virtual Reality Research Center leading the 3D Graphics Research Group. His research interests include in virtual reality, computer graphics, computer vision, and robotics.